CA AuthMinder™

Web Services Developer's Guide

r7.1

This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the “Documentation”) is for your informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and may not be disclosed by you or used for any purpose other than as may be permitted in (i) a separate agreement between you and CA governing your use of the CA software to which the Documentation relates; or (ii) a separate confidentiality agreement between you and CA.

Notwithstanding the foregoing, if you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all CA copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to CA that all copies and partial copies of the Documentation have been returned to CA or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is CA.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b)(3), as applicable, or their successors.

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.

Contents

Chapter 1: Getting Started 11

Introduction to the AuthMinder Web Services 12
AuthMinder Web Services Features 15
Before You Begin 16
Develop Client Applications Using AuthMinder WSDL Files 16
Quick Summary 18

Chapter 2: Understanding AuthMinder WorkFlows 21

Enrollment Workflows 21
    Enrolling New Users 22
    Migrating Existing Users 23
ArcotID PKI Authentication Workflow 28
ArcotID PKI Roaming Download Workflow 30
Forgot Your Password Workflow 32
Workflow Summary 33

Chapter 3: Managing Web Services Security 35

Authentication Header Elements 35
Authorization Header Elements 36
SOAP Header Namespace 36

Chapter 4: Managing Organizations 39

Creating Organizations 40
Updating Organizations 46
Updating Organization Status 48
Refreshing the Organization Cache 50
Fetching Default Organization Details 52
Fetching Organization Details 55
Searching Organizations 57
Fetching AuthMinder Database Attributes 59
Fetching Directory Service Attributes 62
Deleting Organizations 64

Chapter 5: Configuration Management Web Service 67

Managing Account Types 68

Creating Account Types 69
Updating Account Types 71
Fetching Account Types 73
Deleting Account Types....75
Fetching Email and Telephone Types 77
Fetching Email Types....77
Fetching Telephone Types 79
Fetching User Attributes Configured for Encryption....81

Chapter 6: Managing Users and Accounts 85

Before You Proceed....85
User States 86
Supported User State Transitions 86
User Operations and States 87
User Account Operations and States 88
Performing User Operations 89
Creating Users 89
Updating Users....94
Updating User Status 98
Fetching User Details 100
Searching Users by Using Pagination 105
Searching All Users....108
Fetching User Status 112
Updating User Status 114
Deleting Users 116
Performing User Account Operations 118
Adding User Accounts 119
Updating User Accounts....122
Fetching All Accounts of a User....124
Fetch a User Account Details 126
Fetching User Details Using Accounts....129
Deleting User Accounts....133
Setting the Personal Assurance Message....135
Fetching the Personal Assurance Message 137
Setting Custom User Attributes 139
Authenticating LDAP Users 141
Using the LDAP Password 141
Using Directory Service Attributes....143

Chapter 7: Managing AuthMinder Configurations 151

Creating Configurations 152

Preparing the Request Message 152
Invoking the Web Service 181
Interpreting the Response Message 182
Updating Configurations 183
Fetching Configurations 184
Assigning Default Configurations 187
Fetching Server Events 191
Checking Key Availability in HSM 193
Deleting Configurations 195

Chapter 8: Performing Credential Operations 199

Before You Proceed 201
Checking the User Status 201
Credential States and Supported Transitions 202
Credential Operations and States 204
Creating Credentials 206
Common Input Elements 206
ArcotID PKI Input Elements 209
One-Time Password (OTP) Input Elements 210
OATH OTP Input Elements 210
ArcotID OTP Input Elements 210
EMV OTP Input Elements 211
Questions and Answers (QnA) Input Elements 211
Password Input Elements 212
Disabling Credentials 215
Enabling Credentials 217
Resetting Credentials 219
Fetching Credential Details 221
Reissuing Credentials 223
Resetting Credential Validity 225
Resetting Custom Attributes 227
Fetching QnA Configuration 228
Setting Unsigned Attributes 231
Deleting Unsigned Attributes 234
Adding Elements to ArcotID PKI Key Bag 236
Fetching ArcotID PKI Key Bag Elements 239
Deleting ArcotID PKI Key Bag Elements 242
Downloading Credentials 245
Deleting Credentials 247

Chapter 9: Integrating ArcotID PKI Client with Your Application 249

ArcotID PKI Client Overview 249

Flash Client 249

Signed Java Applet 250

Copying ArcotID PKI Client Files 250

For Flash Client 250

For Java Signed Applet 251

ArcotID PKI Client APIs 251

Downloading ArcotID PKI 252

Signing the Challenge 252

Chapter 10: Authenticating Users 253

ArcotID PKI Authentication 254

Step 1: ArcotID PKI Download 254

Step 2: ArcotID PKI Authentication 258

Questions and Answers Authentication 264

Password Authentication 269

Complete Password Authentication 269

Partial Password Authentication 272

One-Time Password Authentication 277

OATH One-Time Password Authentication 280

OATH One-Time Password Synchronization 283

ArcotID OTP (ArcotID OTP-OATH) Authentication 286

ArcotID OTP (ArcotID OTP-OATH) Synchronization 289

EMV OTP (ArcotID OTP-EMV) Authentication 292

EMV OTP (ArcotID OTP-EMV) Synchronization 295

Verifying Password Type Credentials 298

Verifying the Authentication Tokens 301

Fetching the PAM 304

Chapter 11: Performing Bulk Operations 305

Assigning Credentials to Users 306

Uploading OATH Tokens 307

Fetching OATH Tokens 311

Appendix A: Input Data Validations 315

AuthMinder Validation Checks 315

User Attributes Validation Checks 331

Appendix B: AuthMinder Logging 335

About the Log Files 336
Installation Log File 337
AuthMinder Server Startup Log File 337
AuthMinder Server Log File 338
UDS Log File 339
Administration Console Log File 340
Format of the AuthMinder Log Files 341
Format of UDS and Administration Console Log Files 342
Supported Severity Levels 343
Server Log File Security Levels 343
Administration Console and UDS Log File Severity Levels 344
Sample Entries for Each Log Level 345

Appendix C: Enabling SSL for Web Services 347

Setting up SSL 347
One-Way SSL 349
Two-Way SSL 350

Appendix D: Error Codes 353

User Data Service Error Codes 353
AuthMinder Server Codes 373

Chapter 1: Getting Started

This chapter briefly discusses the Web services provided by CA AuthMinder (later referred to as AuthMinder), and the checks that you must perform before using the Web services. It covers the following topics:

- Introduction to the AuthMinder Web Services (see page 12)
- AuthMinder Web Services Features (see page 15)
- Before You Begin (see page 16)
- Develop Client Applications Using AuthMinder WSDL Files (see page 16)
- Quick Summary (see page 18)

This guide provides information on how to develop Web applications that use the strong and versatile modes of authentication provided by CA AuthMinder. This guide discusses Java classes and methods that you can use to programmatically integrate with AuthMinder SDK.

Introduction to the AuthMinder Web Services

The AuthMinder Web services provides a programmatic interface that you can use to integrate your application with AuthMinder. The AuthMinder Web services are broadly classified, as follows:

- Organization Management Web Service
- Configuration Management Web Service
- User Management Web Service
- Administration Web Service
- Issuance Web Service
- Authentication Web Service
- Bulk Upload Web Service

Organization Management Web Service

Organization is an AuthMinder unit that can either map to a complete enterprise (or a company) or a specific division, department, or other entities within the enterprise. The organization management Web service is used to create and manage these organizations. You can perform the following operations by using the organization management Web service:

- Create organizations
- Fetch organization information
- Fetch default organization

  An organization that is created by default when you install AuthMinder. If you have set any other organization as the default, then that organization is used.
- Update organization information
- Update organization status
- Refresh organization cache
- Fetch user attributes that AuthMinder supports
- Fetch user attributes that the directory service supports

See chapter, "Managing Organizations" (see page 39) for more information on how to use the organization management Web services.

Configuration Management Web Service

The configuration management Web service is used to perform the following operations:

- Create account types

- Update account types
- Fetch account types
- Fetch email ID and telephone number types configured at the global-level
- Fetch the user attributes that are configured to be stored in the encrypted format

See chapter, "Configuration Management Web Service" (see page 67) for more information on how to use the user management Web services.

User Management Web Service

The user management Web service is used to manage users, user accounts, user's *Personal Assurance Message* (PAM), and authentication operations for LDAP users. You can perform the following operations by using the user management Web service:

- Create users and user accounts
- Search users
- Fetch users and user accounts
- Update user information
- Update user account information
- Fetch and update user status
- Authenticate administrators (password and QnA authentication mechanisms *only*)
- Set and fetch PAM

See chapter, "Managing Users and Accounts" (see page 85) for more information on how to use the user management Web services.

Administration Web Service

The AuthMinder administration Web service is used to perform the following administration operations:

- Create credential profiles and policies
- Update credential profiles and policies
- Fetch credential profiles and policies
- Delete credential profiles and policies
- Assign default configurations
- Set up RADIUS configurations
- Set up other AuthMinder configurations such as, ASSP, RADIUS, and SAML
- Credential type resolution

- Key management

See chapter, "Managing AuthMinder Configurations" (see page 151) for more information on how to use the administration Web services.

Issuance Web Service

The AuthMinder Issuance Web service (also known as *Credential Management Web service*) interacts with AuthMinder Server to create, fetch, enable, and disable credentials. You can perform the following operations by using the Issuance Web service:

- Create credentials for the users.
- Perform credential lifecycle management operations such as, enable, disable, reset credential, reset credential validity, and delete.

See chapter, "Performing Credential Operations" (see page 199) for more information on how to use the Issuance Web services.

Authentication Web Service

The AuthMinder authentication Web service can be used to authenticate users using the out-of-the-box credentials supported by AuthMinder, and also the custom credentials that your system supports.

See chapter, "Authenticating Users" (see page 253) for more information on how to use the authentication Web service.

Bulk Upload Web Service

The AuthMinder Bulk Upload Web service can be used to assign credentials to users, assign and fetch OATH tokens in bulk.

See chapter, "Performing Bulk Operations" (see page 305) for more information on how to use the bulk upload Web service.

AuthMinder Web Services Features

This section discusses the salient features of AuthMinder Web services.

- **Web Services Authentication and Authorization**

  AuthMinder Web services are protected from rogue requests by enabling authentication and authorization for all incoming requests. As a result, all requests to AuthMinder Web services are authenticated for valid credentials. After successful authentication, all requests are then validated for appropriate privileges to access the Web services. See chapter, "Managing Web Services Security" (see page 35) for more information.

  This feature is enabled using the Enable Authentication and Authorization For Web Services page of Administration Console. See chapter, "Getting Started" in *CA AuthMinder Administration Guide* for more information.

- **Handling Multiple Operations Using Single Function**

  You can perform credential lifecycle operations on different credentials simultaneously. For example, you can create ArcotID PKI, Question and Answer, and One-Time Password credentials simultaneously using a single CreateCredential operation.

- **Support for Additional Parameters**

  In addition to the mandatory inputs, the Web services also accept additional input that can be passed as a name-value pair. This input can include information such as locale, calling application details, or profile.

Before You Begin

Before you integrate your application with AuthMinder, you must ensure that:

- The systems on which you plan to install AuthMinder meet the system requirements.

  Book: Refer to the section, "System Requirements" in the *CA AuthMinder Installation and Deployment Guide* for more information.

- You have completed the configuration and planning-related information:

  - You have installed and configured the required number of AuthMinder database instances.

  Book: See sections, "Configuring Database Server" and "Database-Related Post-Installation Tasks" in the *CA AuthMinder Installation and Deployment Guide* for detailed instructions.

  - You have installed the required application server.

  Book: See the section, "Requirements for Java-Dependent Components" in the *CA AuthMinder Installation and Deployment Guide* for more information.

- You install AuthMinder and ensure that the components are up and running.

  Book: See chapter, "Deploying AuthMinder on a Single System" for **single-system deployment** and chapter, "Deploying AuthMinder on a Distributed System" for **distributed-system deployment** in the *CA AuthMinder Installation and Deployment Guide*.

Develop Client Applications Using AuthMinder WSDL Files

To generate client applications, you need to use the WSDL documents that are shipped with AuthMinder. These documents define the request and response messages that are exchanged between your application and AuthMinder Server to perform an operation.

The following table lists the WSDL documents that AuthMinder provides. These WSDLs are available at the following location:

On Windows: <*install_location*>\Arcot Systems\wsdls\

On Unix-based platforms: <*install_location*>/arcot/wsdls/

WSDL File	Description
uds/ArcotOrganizationManagementSvc.wsdl	Used to create and manage organizations in your setup.
uds/ArcotConfigManagementSvc.wsdl	Used to create and manage user account types.

WSDL File	**Description**
uds/ArcotUserManagementSvc.wsdl	Used to create and manage users and user accounts.
webfort/ArcotWebFortAdminSvc.wsdl	Used to manage AuthMinder configurations.
webfort/ArcotWebFortIssuanceSvc.wsdl	Used to manage credentials of the users.
webfort/ArcotWebFortAuthSvc.wsdl	Used to authenticate users.
webfort/ArcotWebFortBulkOperationsSvc.wsdl	Used to perform bulk operations such as assigning and fetching OATH tokens that are available to the organizations.

Important! From this release, AuthMinder WSDLs support SOAP 1.2 binding *only*. If you use SOAP 1.1 binding that was supported by the earlier versions, then you will see the Invalid soap message or soap version mismatch error.

You can use any tool of your choice, such as Apache Axis or .NET SOAP framework to generate client stub classes using the WSDL files listed in this table. You can then use the generated stub classes to build your application and access Web services.

Note: If you are using .NET SOAP framework to generate the client stubs, then you must include the following line in your code before you invoke the ArcotWebFortAdminSvc, ArcotWebFortIssuanceSvc, ArcotWebFortAuthSvc, and ArcotWebFortBulkOperationsSvc WebFort Web services.

```
ServicePointManager.Expect100Continue = false; // which is available in System.Net;
```

If you do not include this line, you might see errors.

Quick Summary

The following steps provide the quick recap of the steps that you need to perform to set up your environment to use AuthMinder Web service:

1. Access the WSDL by navigating to the following location:

   On Windows: <*install_location*>\Arcot Systems\wsdls\

   On Unix-based platforms: <*install_location*>/arcot/wsdls/

2. Generate the client stub classes by using the WSDL files.

   You can use a SOAP framework, such as Apache Axis or Microsoft.NET, to generate client stub classes from a WSDL file.

3. Create the client application using the stub classes generated in Step 2.

   Depending on the software that you choose, refer to the respective vendor documentation for more information on writing the client and the files required for the client to connect to the AuthMinder Web service.

4. Connect the client to the AuthMinder Web service end point by using the default URLs listed in the following table.

   Note: This table lists the default URLs on which the Web services are available. If you change the service end point URL, then ensure that you connect your client to the new location that you have configured.

Web Service	**URL**
Organization Management Web Service	*http://<Apphost>:CA Portal/arcotuds/services/ArcotUserRegistryMgmtSvc* ■ **Apphost**: Host name or the IP address of the system where User Data Service (UDS) is deployed. ■ **Port**: The port number at which the application server (on which UDS is deployed) is listening to.
User Management Web Service	*http://<Apphost>:CA Portal/arcotuds/services/ArcotUserRegistrySvc* ■ **Apphost**: Host name or the IP address of the system where UDS is deployed. ■ **Port**:The port number at which the application server (on which UDS is deployed) is listening to.

Web Service	**URL**
Configuration Registry Web Service	*http://<Apphost>:CA Portal/arcotuds/services/ArcotConfigRegistrySvc* ■ **Apphost**: Host name or the IP address of the system where User Data Service (UDS) is deployed. ■ **Port**: The port number at which the application server (on which UDS is deployed) is listening to.
Administration Web Service	*http://<Apphost>:CA Portal/ArcotWebFortAdminSvc* ■ **Apphost**: Host name or the IP address of the system where AuthMinder Server is installed. ■ **Port**: The port number at which the Administration Web Services protocol is listening to. By default, this port number is **9745**.
Issuance Web Service	*http://<Apphost>:CA Portal/ArcotWebFortIssuanceSvc* ■ **Apphost**: Host name or the IP address of the system where AuthMinder Server is installed. ■ **Port**: The port number at which the Transaction Web Services protocol is listening to. By default, this port number is **9744**.
Authentication Web Service	*http://<Apphost>:CA Portal/ArcotWebFortAuthSvc* ■ **Apphost**: Host name or the IP address of the system where AuthMinder Server is installed. ■ **Port**: The port number at which the Transaction Web Services protocol is listening to. By default, this port number is **9744**.
Bulk Operation Web Service	*http://<Apphost>:CA Portal/ArcotWebFortBulkOperationsSvc* ■ **Apphost**: Host name or the IP address of the system where AuthMinder Server is installed. ■ **Port**: The port number at which the bulk operations Web Services protocol is listening. By default, this port number is **9745**.

Note: To secure the connection using SSL, enable the Web Services protocols for SSL connection. Refer to *CA AuthMinder Administration Guide* for more information.

1. Send the requests to the AuthMinder Web services through the client.

   The AuthMinder Web service processes the request and returns the message, response code, reason code, and transaction ID in the response.

Chapter 2: Understanding AuthMinder WorkFlows

AuthMinder enables you to design different workflows that can be built using the Administration, Authentication, and Issuance Web services. Based on your organization’s requirements, you can design these workflows without significantly changing the existing online experience of your users in most cases.

Note: The tasks that are listed in this chapter can be customized in multiple ways. The workflows depicted here are examples of the typical workflows. You need not follow the exact steps for each procedure mentioned in this chapter.

This chapter describes the sample workflows and provides an overview of each:

- Enrollment Workflows (see page 21)
- ArcotID PKI Authentication Workflow (see page 28)
- ArcotID PKI Roaming Download Workflow (see page 30)
- Forgot Your Password Workflow (see page 32)
- Workflow Summary (see page 33)

Enrollment Workflows

Enrollment is the process of creating a user and creating credentials for the user. The user can reside either in the AuthMinder database or in an external directory service such as Microsoft's *Active Directory Service* (ADS) or SunOne Directory Server. If directory service is used, then user need not be created in AuthMinder, but their attributes must be mapped to the AuthMinder database attributes.

Note: See *CA AuthMinder Administration Guide* for information on how to map the user attributes from the external directory to the AuthMinder database entries.

Based on whether you are enrolling a new user or migrating existing users to AuthMinder authentication, the enrollment workflow can include:

- Enrolling New Users (see page 22)
- Migrating Existing Users (see page 23)

Enrolling New Users

The user enrollment is performed by User Data Service (UDS) component of AuthMinder. UDS is used to manage organizations and users in the system. UDS also serves as an abstraction layer that provides AuthMinder seamless access to the third-party data repositories deployed by your organization.

UDS is shipped as a library file and as a WAR file (arcotuds.war).

If you are using a **relational database** to store the user information, then AuthMinder uses the library file to connect to the database for performing user operations.

If you are using an **LDAP directory server** and you want AuthMinder to seamlessly access it, then you *must* deploy the arcotuds.war file in the application server where your application integrated with AuthMinder is deployed. In this case, the attributes in the LDAP must be mapped to the user attributes that CA supports. This mapping information is stored in the relational database where the schema is seeded.

Book: Refer to *CA AuthMinder Installation and Deployment Guide* for more information on deploying UDS file. Refer to *CA AuthMinder Administration Guide* for more information on mapping user attributes.

The typical steps for enrolling a new user are:

1. User accesses the enrollment page of your application.

   The user enters the information, such as user name, first name, last name, email address, and contact information required to create the users.

   Note: The username must be unique for an organization, which means two users in the same organization *cannot* have same username.

2. Your application collects information entered by the user in the preceding step and calls createUser operation in the ArcotUserRegistrySvc service.

   At this stage, your application makes an explicit call to the createUser operation in ArcotUserRegistrySvc service. In this call, you pass user details such as, user name, last name, organization, PAM (Personal Assurance Message), email address, and telephone number.

   See "Creating Users" (see page 89) for more information on the operation used for creating users.

3. User Data Service checks if the user exists.

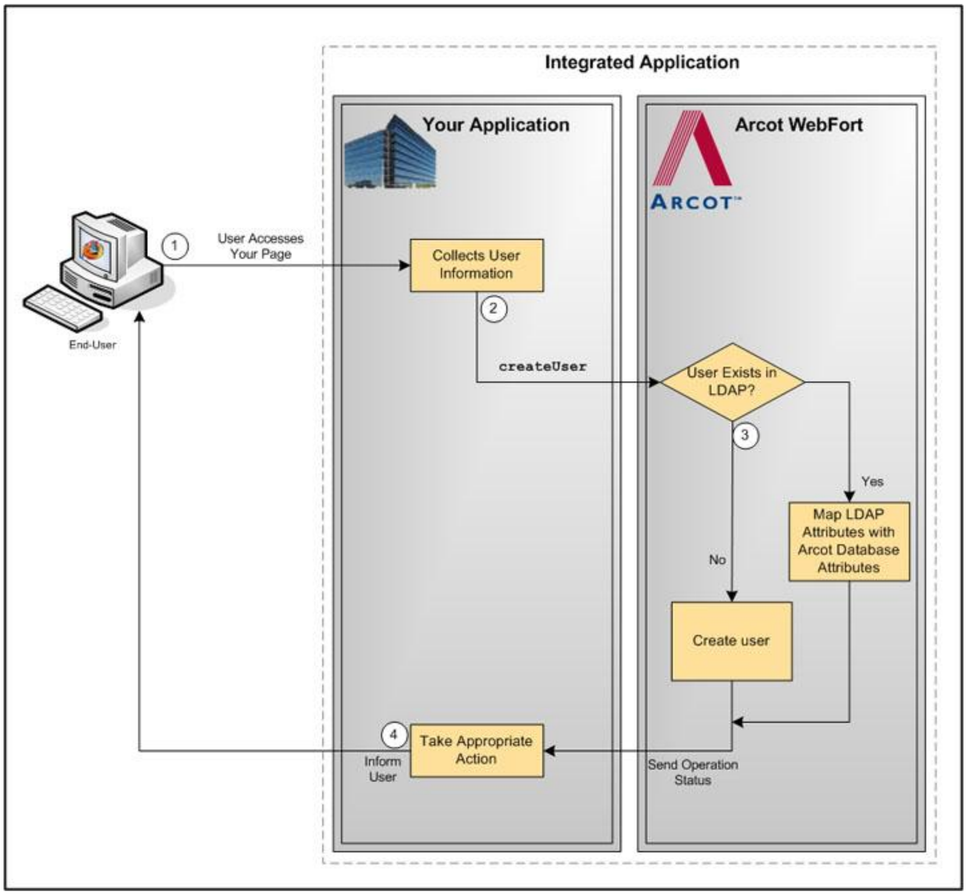
   UDS checks whether the user is present in the directory service (LDAP). If the user exists, then it maps the LDAP attributes to AuthMinder database.

   If the user does not exist in the AuthMinder database, then UDS creates the user record in the database. With this, the user is enrolled with AuthMinder.

4. AuthMinder informs the calling application.

   AuthMinder sends the status of the operation to your calling application.

The following figure illustrates the enrollment workflow when you call the createUser operation:



Migrating Existing Users

AuthMinder enables you to easily migrate the users from your existing authentication method to strong authentication methods supported by AuthMinder.

- Migrating All Users (see page 24)
- Migrating Selected Users (see page 26)

Migrating All Users

The typical steps to migrate all users are:

1. User logs in to your application.

   The user logs in to your application by using *your existing* authentication method.

2. Your application collects the required information from the user to create the credential.

   Your application can display the appropriate pages to the user. For example, you can prompt the user to set the password for ArcotID PKI or you can set the existing password as the ArcotID PKI password, and collect questions and answers if *Question and Answer* (QnA) is used for secondary authentication.

3. Your application invokes the CreateCredential operation of the ArcotWebFortIssuanceSvc service.

   This operation creates ArcotID PKI for the user.

   See "Creating Credentials" (see page 206) for more information on the API used for creating credentials for the users.

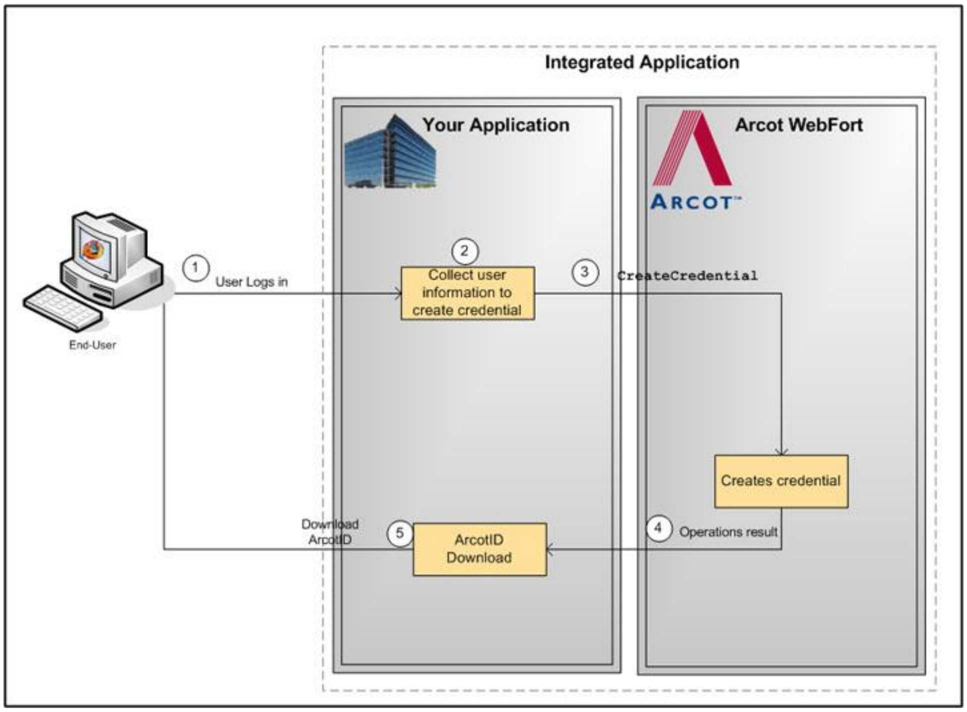
4. AuthMinder returns the result.

   If the create operation was successful, then user’s ArcotID PKI is returned.

5. Your application downloads the ArcotID PKI on the user’s system.

   If the create operation was successful, then the application downloads the ArcotID PKI to the end-user’s system without any user interaction.

The following figure illustrates the workflow for migrating all users in the system:


Integrated Application
Your Application
Arcot WebFort
ARCOT
End-User
1 User Logs in
2 Collect user information to create credential
3 CreateCredential
Creates credential
4 Operations result
5 ArcotID Download
Download ArcotID

Migrating Selected Users

The typical steps to migrate selected users are:

1. User logs in to your application.

   The user logs in to your application by using *your existing* authentication method.

2. Application gets the user status.

   Application retrieves user information and identifies whether the user account is marked for migration.

3. Application redirects user.

   Upon successful authentication, the user is redirected to migration page.

4. Your application collects the required information from user to create the credential.

   Your application can display the appropriate pages to the user. For example, you can prompt the user to set the password for ArcotID PKI or you can set the existing password as the ArcotID PKI password, and collect questions and answers if QnA is used for secondary authentication.

5. Your application invokes the CreateCredential operation of the ArcotWebFortIssuanceSvc service.

   This operation creates ArcotID PKI for the user.

   See "Creating Credentials" (see page 206) for more information on the API used for creating credentials for the users.

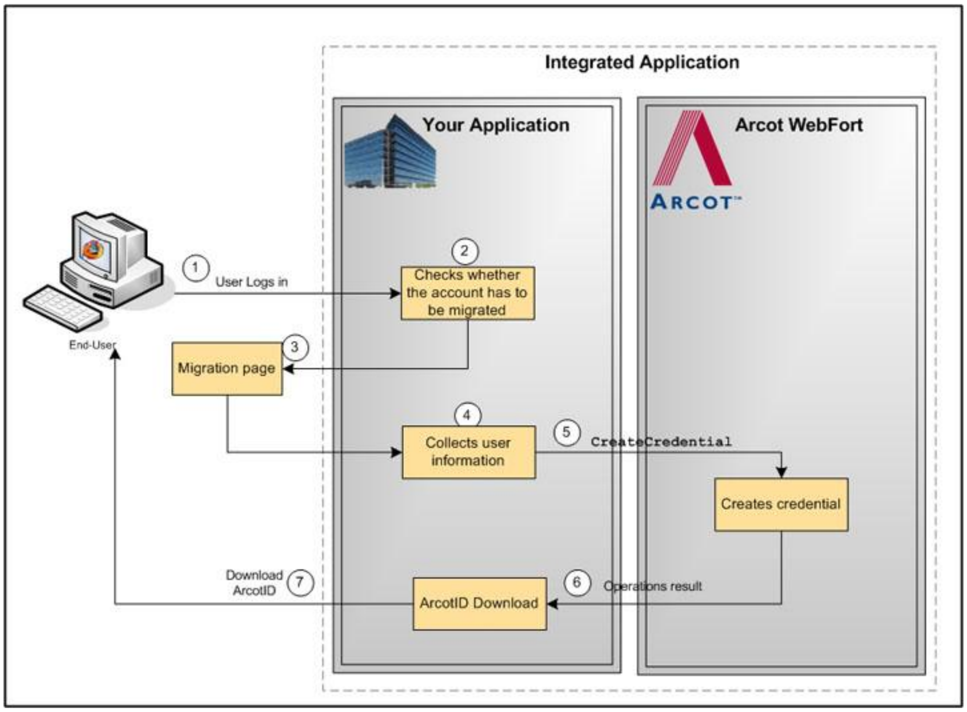
6. WebFort returns the result.

   If the create operation was successful, then user’s ArcotID PKI is returned.

7. Application downloads the ArcotID PKI on the user’s system.

   If the create operation was successful, then the application downloads the ArcotID PKI to the end-user’s system without any user interaction.

The following figure illustrates the workflow for migrating the selected users to ArcotID PKI authentication:


Integrated Application
Your Application
Arcot WebFort
ARCOT
End-User
1 User Logs in
2 Checks whether the account has to be migrated
3 Migration page
4 Collects user information
5 CreateCredential
Creates credential
6 Operations result
ArcotID Download
Download ArcotID 7

ArcotID PKI Authentication Workflow

During authentication, when a user specifies the credential in the authentication page, the credential is first verified by AuthMinder Server, after which the user is authenticated. The following workflow lists the steps for ArcotID PKI authentication:

Note: In case of other credentials, see chapter, "Authenticating Users" (see page 253) for details of operations to invoke.

1. Application calls AuthMinder's GetArcotIDChallenge operation of the ArcotWebFortAuthSvc service.

   Your application loads the ArcotID PKI Client and makes an explicit call to fetch the challenge. See "ArcotID PKI Authentication" (see page 254) for more information on the Web services details.

2. User provides the credentials.

   User specifies the user name and ArcotID PKI password to log in.

3. Your application passes the user name and password to the ArcotID PKI Client.

   The ArcotID PKI Client signs the challenge.

4. AuthMinder verifies the signed challenge.

   Your application invokes the verifyArcotIDSignedChallenge operation of the ArcotWebFortAuthSvc service to verify the challenge that is signed by using the ArcotID PKI Client.

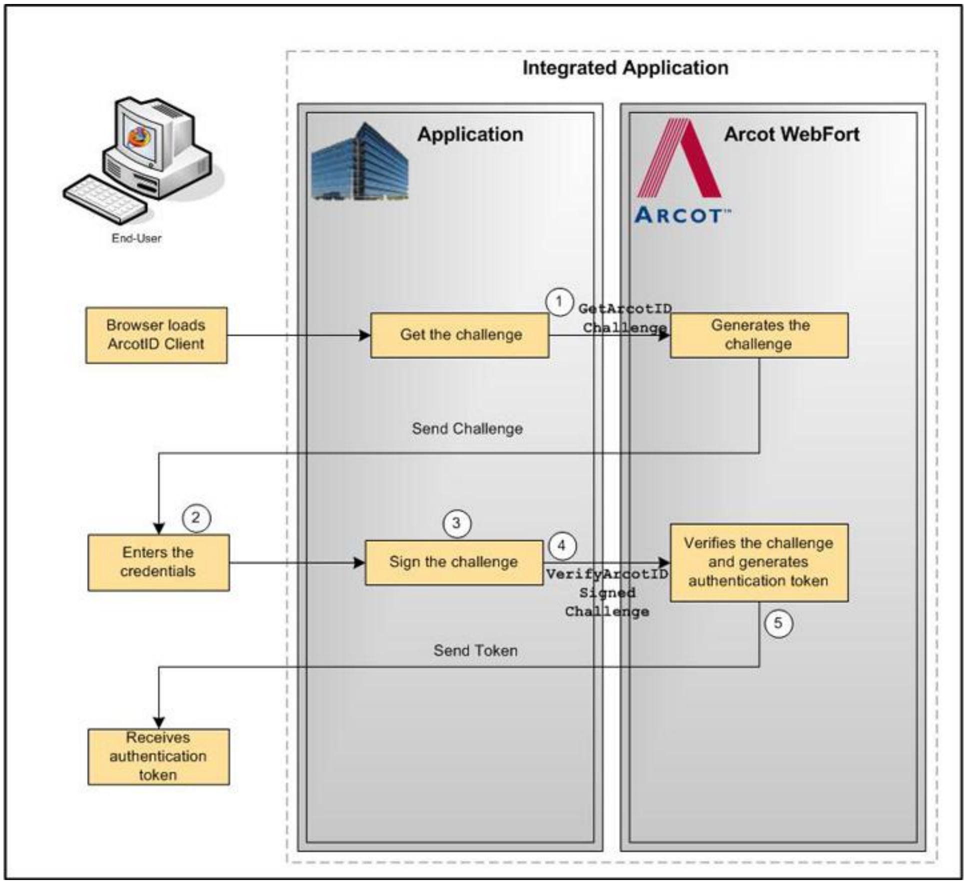
   See "ArcotID PKI Authentication" (see page 254) for more information on the API used for authenticating users with their ArcotID PKI credential.

5. AuthMinder authenticates the user.

   If the verifyArcotIDSignedChallenge operation returns the successful response, then the authentication token generated indicates that the user is authenticated successfully.

   See "Verifying the Authentication Tokens" (see page 301) for more information on the different tokens supported by AuthMinder.

The following figure illustrates the workflow for ArcotID PKI authentication process:


Integrated Application
Application
Arcot WebFort
ARCOT
End-User
Browser loads ArcotID Client
Get the challenge
1
GetArcotID Challenge
Generates the challenge
Send Challenge
2
Enters the credentials
3
Sign the challenge
4
VerifyArcotID Signed Challenge
Verifies the challenge and generates authentication token
5
Send Token
Receives authentication token

ArcotID PKI Roaming Download Workflow

To perform ArcotID PKI authentication, the ArcotID PKI of the user must be present on the user's system that is used by the authentication session. If the user is travelling or does not have access to the system, where their ArcotID PKI is stored, then the user has to download the ArcotID PKI from AuthMinder Server and then perform the authentication.

The typical steps for roaming download of the ArcotID PKI are:

1. User logs in to your online application.

   Your application authenticates the user.

2. User chooses to download the ArcotID PKI.

   Your application displays the appropriate page to the user to download their ArcotID PKI.

3. AuthMinder performs secondary authentication.

   Based on the secondary authentication mechanism that you are using, your application displays appropriate pages to the user. For example, your application can prompt the user to:

   - Answer the security questions that they selected while enrolling with your application.
   - Enter the OTP, which is sent to the user by email, SMS, or other customized method.

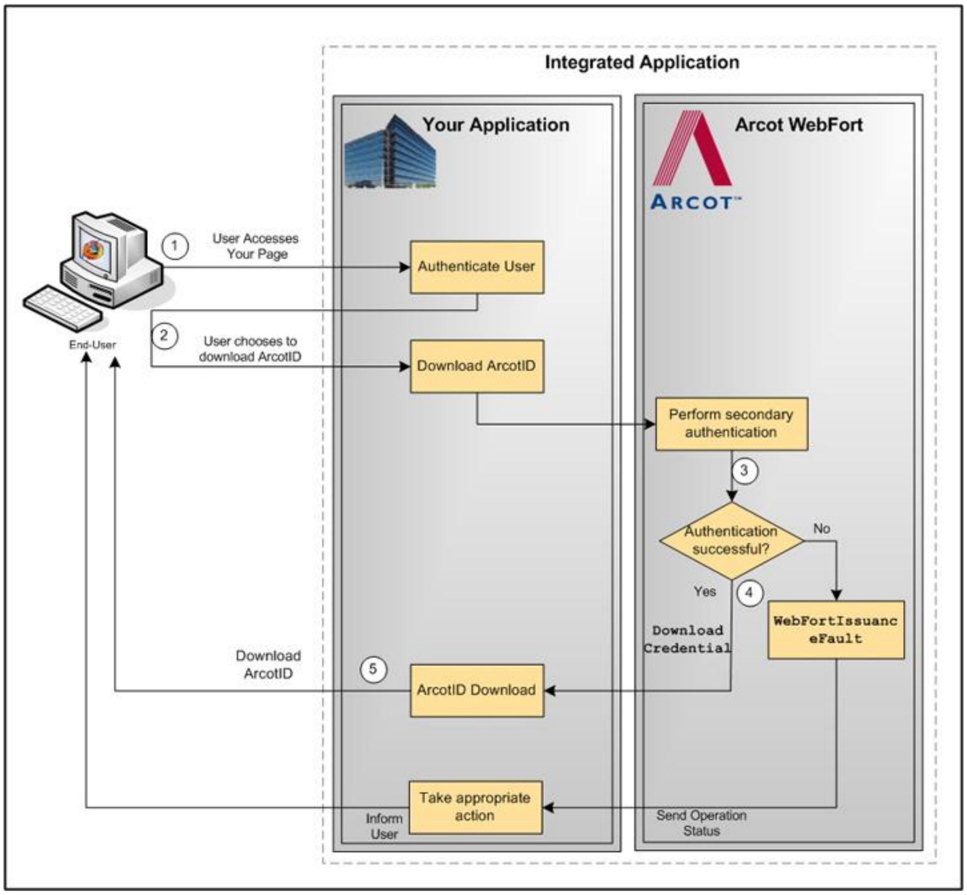
4. Your application calls AuthMinder's DownloadCredential operation of the ArcotWebFortIssuanceSvc service.

   If the secondary authentication was successful, only then your application should call the DownloadCredential operation. This call downloads the corresponding ArcotID PKI to the your application.

5. Download the ArcotID PKI to user's system.

   Invoke the ImportArcotID() client-side JavaScript API to download the ArcotID PKI to the end-user's system without any user interaction.

The following figure illustrates the workflow for roaming download of ArcotID PKI:


Integrated Application
Your Application
Arcot WebFort
ARCOT
End-User
1
User Accesses Your Page
Authenticate User
2
User chooses to download ArcotID
Download ArcotID
Perform secondary authentication
3
Authentication successful?
No
Yes
4
WebFortIssuanceFault
Download Credential
Download ArcotID
5
ArcotID Download
Take appropriate action
Inform User
Send Operation Status

Forgot Your Password Workflow

If a user forgets their ArcotID PKI password, then *Forgot Your Password* (FYP) workflow can be used to reset the password.

In this method, the user is prompted to answer the questions, which they had set during enrollment or you can use any other customized method of your choice.

The typical steps for FYP workflow are:

1. User accesses your online application.
2. User provides the user name.

   User specifies the user name to log in.
3. User clicks the FYP link.

   Because the user does not remember their password, they click the FYP link.
4. AuthMinder performs secondary authentication.

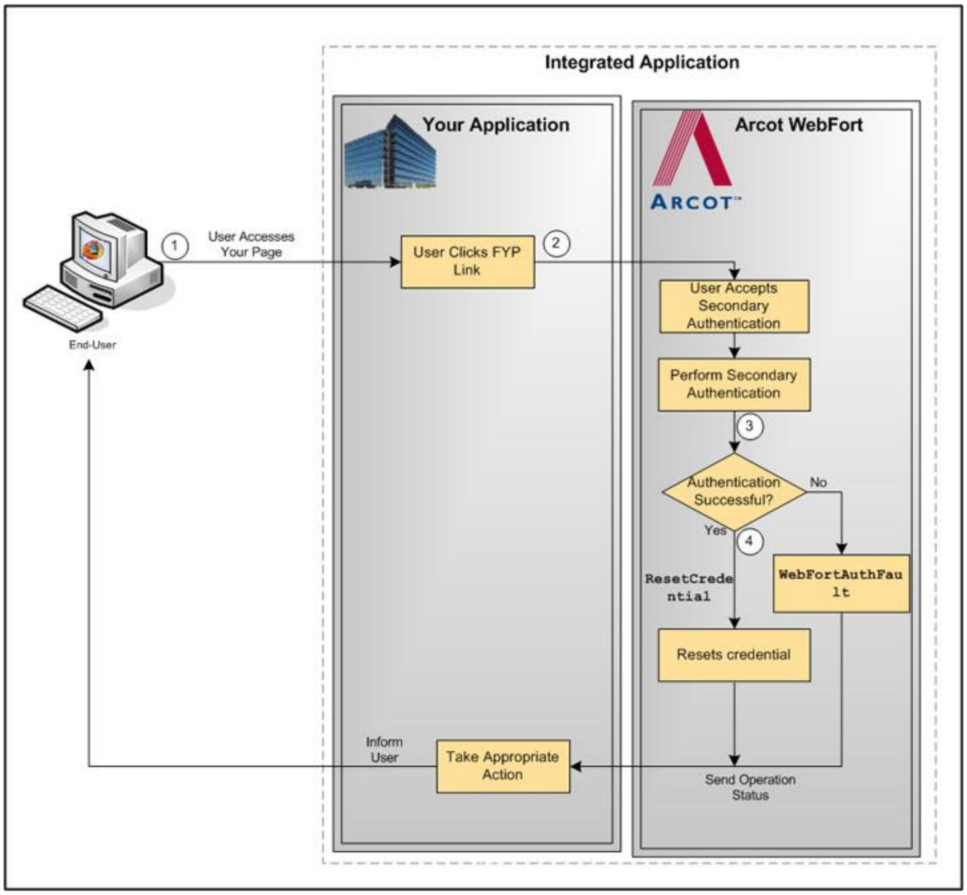
   Based on the secondary authentication mechanism that you are using, the appropriate pages are displayed to the user. For example, the user can be prompted to:

   - Answer the security questions that they selected while enrolling with your application.
   - Enter the OTP, which is sent to them by email, SMS, or other customized method.
5. Your application calls AuthMinder's ResetCredential operation of the ArcotWebFortIssuanceSvc service.

   If the secondary authentication was successful, then your application must invoke the ResetCredential operation. Your application prompts the user for new password and pass this as input for ResetCredential operation.

   See "Resetting Credentials" (see page 219) for more information on the Web services used to reset the credential.

The following figure illustrates the Forgot Your Password workflow:



Workflow Summary

The following table provides a brief summary of the workflows that can be implemented by using the AuthMinder Web services:

Workflow	Description	Dependant Workflows
Enrollment	Creates a new user in the AuthMinder database, when you call CreateUser operation.	None
Creating the Credentials	Create the credentials for the user.	■ Enrollment

Workflow	**Description**	**Dependant Workflows**
Authentication	Authenticates the user by using the credentials provided by the user.	■ Enrollment ■ Creating the Credentials
ArcotID PKI Download	Downloads the ArcotID PKI of the user to the system.	■ Enrollment ■ Creating the Credentials ■ Secondary authentication
Migration	Migrates the user to ArcotID PKI authentication.	None
FYP	Resets the password.	■ Enrollment ■ Creating the Credentials

Chapter 3: Managing Web Services Security

To restrict the rogue requests to Web services, you can prompt the incoming requests for authentication. To enable this feature, you need to ensure that the calling application includes the user credentials in the incoming call header.

The Web services authentication and authorization works as follows:

1. The calling application authenticates to the AuthMinder Web services by including the required credentials in the call header.
2. The Web service authenticates these credentials and, if valid, provides the calling application with an authentication token.
3. The calling application includes the authentication token and the authorization elements in the header of the subsequent calls.

This chapter covers the following information:

- Authentication Header Elements (see page 35)
- Authorization Header Elements (see page 36)
- SOAP Header Namespace (see page 36)

Authentication Header Elements

The following table lists the elements that have to be included in the call header for authentication.

Note: For configuration management, credential management, user authentication, and bulk operations Web services that are discussed in chapter, "Managing AuthMinder Configurations" (see page 151), chapter, "Performing Credential Operations" (see page 199), chapter, "Authenticating Users" (see page 253), and chapter, "Performing Bulk Operations" (see page 305) you can either pass the authentication details in the call header or as an additional input in the call body.

Element	Mandatory	Description
userID	Yes	The unique identifier of the user whose account has to be authenticated.
orgName	Yes	The organization name to which the authenticating user belongs.
credential	Yes	The credential of the user that is to be used for authentication.

Authorization Header Elements

The following table lists the elements that you need to pass in the call header for authorization:

Element	Mandatory	Description
authToken	Yes	The authentication token that is returned after successful user verification. This token indicates that the user is already authenticated, and therefore eliminates the need for user credentials for successive authentication attempts. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Note: You can set any *one* of the following elements.		
targetorg	No	Specifies the organization for which the calling application must authorize before performing any operation. **Note:** If you want to enable authorization for more than one organization, then accordingly repeat this entry for every organization.
targetAllOrgs	No	Specifies whether authorization is required to perform operations on all organizations. Set the value of this element to TRUE to enable authorization for all organizations.
globalEntity	No	Specifies whether authorization is required to perform global configurations. Set this value to TRUE if you want to enable authorization for the global configuration operations such as, fetching AuthMinder attributes for users and fetching UDS attributes.

SOAP Header Namespace

The authentication and authorization header elements must have the namespace, as mentioned in the following table:

Web Service	Namespace
User Data Service Web Services	
■ User Management ■ User Registry Management ■ Configuration Registry	*http://ws.arcot.com/UDSTransaction/1.0*
AuthMinder Web Services	

Web Service	**Namespace**
Credential Issuance	*http://ws.arcot.com/WebFortIssuanceAPI/7.0/msgs*
User Authentication	*http://ws.arcot.com/WebFortAuthAPI/7.0/msgs*
Administration	*http://ws.arcot.com/ArcotWebFortAdminSvc/1.0/msgs*
Bulk Upload	*http://ws.arcot.com/WebFortBulkOperationsAPI/7.0/msgs*

Chapter 4: Managing Organizations

Important! To use the Web service operations that are discussed in this chapter, you *must* deploy the User Data Service (**arcotuds.war**) file.
See section, "Deploying User Data Service" section in the *CA AuthMinder Installation and Deployment Guide* for more information.

In AuthMinder, an *organization* can either map to a complete enterprise (or a company) or a specific division, department, or other entities within the enterprise. The organization structure provided by AuthMinder is flat. In other words, organizational hierarchy (in the form of parent and child organizations) is *not* supported, and all organizations are created at the same level as the Default Organization.

This chapter discusses the Web service operations that AuthMinder provides to create and manage organizations. It covers the following topics:

- Creating Organizations (see page 40)
- Updating Organizations (see page 46)
- Updating Organization Status (see page 48)
- Refreshing the Organization Cache (see page 50)
- Fetching Default Organization Details (see page 52)
- Fetching Organization Details (see page 55)
- Searching Organizations (see page 57)
- Fetching AuthMinder Database Attributes (see page 59)
- Fetching Directory Service Attributes (see page 62)
- Deleting Organizations (see page 64)

You must use the ArcotOrganizationManagementSvc.wsdl file to perform the operations discussed in this chapter.

Creating Organizations

When you deploy Administration Console, an organization is created by default. This out-of-the-box organization is referred to as *Default Organization* (DEFAULTORG). For a single organization setup, instead of creating an organization you can rename the default organization, change its configurations, and then continue to use the default organization.

For a multi-organization setup, you need to create additional organizations. You can do this either by using Administration Console or by using Web services.

This section walks you through the following topics for creating organization:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: After you create an organization, you *must* refresh the system cache for the new organization to take effect. See "Refreshing the Organization Cache" (see page 50) for more information on how to refresh the cache.

Preparing the Request Message

The createOrgRequest message is used to create organizations in the AuthMinder database. The following table lists the elements of this request message:

Element	**Mandatory**	**Description**
orgName	Yes	The unique name of the organization that you want to create. This name is used to log in to Administration Console.
displayName	Yes	A descriptive name for the organization.
keyLabel	No	The label for the key that is used to encrypt the sensitive user and organization data. Setting the key label is a one-time operation. After you set this value, you *cannot* modify it. **Note:** If this value is not specified, then the Master Key is used as the key label.

Element	Mandatory	Description
repositoryType	No	The repository where the accounts of the users belonging to the organization must reside. The repository can be one of the following: ■ **ARUSER**: Indicates that the user accounts will be created in a Relational Database Management System (RDBMS). AuthMinder supports MS SQL, Oracle, IBM DB2, and MySQL. ■ **LDAP**: Indicates that the user accounts existing in your directory service will be used. **Note**: If you choose this option, then ensure that you have deployed User Data Service (UDS) successfully.

Element	Mandatory	Description
ldapDetails	No Required *only* if repositoryType =LDAP	The details of the directory service where the user information is available: ■ **host** The host name of the system where the your directory service is available. ■ **port** The port number at which the directory service is listening. ■ **schemaName** The LDAP schema used by the directory service. This schema specifies the types of objects that a directory service can contain, and specifies the mandatory and optional attributes of each object type. Typically, the schema name for Active Directory is user and for SunOne Directory, it is inetOrgPerson. ■ **baseDN/dnEntry** The name-value key pairs of the base *Distinguished Name* (DN) of the directory service. This value indicates the starting node in the LDAP hierarchy to search in the directory service. For example, to search or retrieve a user with a DN of cn=rob laurie, ou=sunnyvale, o=arcot, c=us, you must specify the base DN as the following: ou=sunnyvale, o=arcot, c=us Typically, these values are case sensitive and searches all subnodes under the specified base DN.

Element	Mandatory	Description
connectionCredential	No Required *only* if repositoryType =LDAP	The information required to connect to the directory service: ■ **ssl** The type of connection that has to be established with the directory service. Possible values are: **TCP**:Indicates that the directory service will listen to incoming requests on TCP. **1WAY**: Indicates that the directory service will listen to incoming requests on one-way SSL. **2WAY**: Indiactes that the directory service will listen to incoming requests on two-way SSL. ■ **loginName** The complete distinguished name of the LDAP repository user who has the privilege to log into repository sever and manage the base DN. For example, uid=gt,dc=arcot,dc=com ■ **loginPassword** The password of the user provided in loginName. ■ (Optional) **serverTrustCert** The based64-encoded trusted root certificate of the server that issued the SSL certificate to the directory service. This parameter is required *only* if ssl is set to 1WAY or 2WAY. ■ (Optional) **clientKeyStore** The password for the client key store and the base64-encoded root certificate of UDS. This parameter is required *only* if ssl is set to 2WAY.
redirectSearchSchema	No Required *only* if repositoryType =LDAP	The schema to be used when searching for values whose attributes are in a different node.
redirectSearchAttribute	No Required *only* if repositoryType =LDAP	The value of the attribute to be searched in redirectSearchSchema.

Element	Mandatory	Description
repositoryattribute	No Required *only* if repositoryType =LDAP	The user attribute in the directory service that has to be mapped to the AuthMinder attribute. Based on this mapping, UDS searches for the user in the directory service.
arcotattribute	No Required *only* if repositoryType =LDAP	The AuthMinder attribute to which the directory service attribute must be mapped. For example, you can map the UID attribute in the directory service to the USERNAME attribute.
status	No	The status of the organization in the database. Following are the supported values: ■ INITIAL Indicates that the organization is not yet activated and *cannot* be used for any operations. ■ ACTIVE Indicates that the organization has been successfully created and activated. You can perform any supported operation on the organization. ■ INACTIVE Indicates that the organization has been deactivated. To perform any further operation, you must first activate the organization. ■ DELETED Indicates that the organization has been deleted and cannot be used anymore. **Note:** If the organization status element is not set, then the organization is created with the INITIAL state.
description	No	A description for the organization that will help the administrators managing the organization to easily identify the organization.
customAttribute	No	Name-value pairs that you can use to set any additional user or organization information.
clientTxId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To create organizations:

1. (Optional) Include the authentication and authorization details in the header of the createOrg operation.

   See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.

2. Use the createOrgRequest elements to set the organization information.
3. Use the createOrgRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the createOrg operation of the ArcorUserRegistryMgmtSvc service to create the organization.

   This operation returns the createOrgResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, createOrgResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table.

The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	**Description**
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Updating Organizations

The updateOrg operation enables you to update the following organization information:

- Display name
- Description
- Custom attributes

Note: In addition to the elements that are required to perform the tasks mentioned in the preceding list, the updateOrgRequest contains other elements for repository (directory service or AuthMinder database) configuration and user attribute mapping. After you create an organization, you *cannot* change the repository type and the related settings. Therefore, these elements are *not* applicable when you update an organization. Even if you set these elements, they will *not* be considered.

This section walks you through the following topics for updating organizations:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: After you update an organization, you *must* refresh the system cache for the changes to take effect. See "Refreshing the Organization Cache" (see page 50) for more information on how to refresh the system cache.

Preparing the Request Message

The updateOrgRequest message is used to update organizations in the AuthMinder database. The following table lists the elements of this request message.

Note: This table lists *only* the elements that you can use to update the organization information. You can ignore other additional updateOrgRequest elements that are not applicable, such as repository type (repositoryDetails) configuration, user attribute mapping (mappingDetails) configuration, and status.

Element	**Mandatory**	**Description**
orgName	Yes	The name of the organization that has to be updated.
displayName	No	The descriptive name of the organization.
description	No	A description for the organization that will help the administrators easily identify the organization.
customAttribute	No	Name-value pairs that you can use to set any additional user or organization information.

Element	Mandatory	Description
clientTxId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To update an organization:

1. (Optional) Include the authentication and authorization details in the header of the updateOrg operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the upateOrgRequest elements to update the organization information.
3. Use the upateOrgRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the updateOrg operation of the ArcorUserRegistryMgmtSvc service to update the organization.

   This operation returns the updateOrgResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, updateOrgResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	Description
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Updating Organization Status

The updateOrgStatus operation is used to update the status of the organization in the AuthMinder database.

This section walks you through the following topics for updating the organization status:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: After you update the organization status, you *must* refresh the system cache for the changes to take effect. See "Refreshing the Organization Cache" (see page 50) for more information on how to refresh the system cache.

Preparing the Request Message

The updateOrgStatusRequest message is used to update the organization status. The following table lists the elements of this request message:

Element	**Mandatory**	**Description**
status	Yes	The status of the organization in the database. Following are the supported values: ■ INITIAL Indicates that the organization is not yet activated and *cannot* be used for any operations. ■ ACTIVE Indicates that organization has been successfully created and activated. You can perform any operation on the organization. ■ INACTIVE Indicates that the organization has been deactivated. To perform any further operation, you need to *first* activate the organization. ■ DELETED Indicates that the organization has been deleted and cannot be used anymore.
OrgName	Yes	The unique name with which the organization is identified.
clientTxId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To update the organization status:

1. (Optional) Include the authentication and authorization details in the header of the updateOrgStatus operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the upateOrgStatusRequest elements to update the organization status.
3. Use upateOrgStatusRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the updateOrgStatus operation of the ArcorUserRegistryMgmtSvc service to update the organization status.

   This operation returns the updateOrgStatusResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, updateOrgStatusResponse returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	Description
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Refreshing the Organization Cache

The refreshCache operation is used to refresh the organization configurations that are stored in the cache. This section walks you through the following topics for refreshing the organization cache:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The refreshCacheRequest is used to refresh the organization cache. The following table lists the elements of this request message:

Element	**Mandatory**	**Description**
systemCache	No	Specify whether you want to refresh all the cache of the AuthMinder setup. Possible values are: ■ True: Indicates that all the cache, which includes all organizations and server cache. ■ False: If you select this option, then you can refresh the organization cache, by selecting allOrganization or OrgName.
Note: You can set any *one* of the following elements.		
allOrganization s	No	Specifies whether the cache of all organizations has to be refreshed. Set the value of this element to TRUE to refresh the cache of all organizations.
OrgName	Yes	The unique name with which the organizations are identified.
clientTxId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To refresh the organization cache:

1. (Optional) Include the authentication and authorization details in the header of the refreshCache operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.

2. Use the refreshCacheRequest elements for updating the organization configurations.

3. Use refreshCacheRequest message and construct the input message by using the details obtained in preceding step.

4. Invoke the refreshCache operation of the ArcorUserRegistryMgmtSvc service to refresh the organization cache.

   This operation returns the refreshCacheResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, refreshCacheResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	**Description**
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication is valid for *one* day, after which you need to authenticate again.

Fetching Default Organization Details

The Master Administrator (MA) sets the default organization in the system. Typically, when you create administrators or enroll users without specifying their organization, they are created in this default organization. The retrieveDefaultOrg operation is used to fetch the details of the default organization.

This section walks you through the following topics for fetching the default organization details:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The retrieveDefaultOrgRequest is used to fetch the default organization information. The following table lists the elements of this request message:

Element	Mandatory	Description
clientTxId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch the default organization information:

1. (Optional) Include the authentication and authorization details in the header of the retrieveDefaultOrg operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the retrieveDefaultOrgRequest elements for fetching the default organization information.
3. Use retrieveDefaultOrgRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the retrieveDefaultOrg operation of the ArcorUserRegistryMgmtSvc service to fetch the default organization details.

   This operation returns the retrieveDefaultOrgResponse message that includes the transaction identifier, authentication token, and default organization details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, retrieveDefaultOrgResponse, returns the transaction identifier and authentication token in the SOAP envelope header. The SOAP body includes the default organization details for a successful transaction and the Fault response for an error condition.

The following table provides information about the elements returned for a successful transaction. See appendix, "Error Codes" (see page 353) if there are any errors.

Element	Description
Header Elements	
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
orgName	The unique name of the organization.
displayName	The descriptive name of the organization.

Element	Description
repositoryDetails	The repository where the accounts of the users belonging to the organization resides. Following are the supported values: ■ ARUSER ■ LDAP
dateCreated	The timestamp when the organization was created.
dateModified	The timestamp when the organization was last modified.
description	The description for the organization that will help the administrators managing the organization.
status	The status of the default organization in the database. Following are the supported values: ■ INITIAL ■ ACTIVE ■ INACTIVE ■ DELETED
preferredLocale	The locale that is configured for the organization. If you do not specify the locale, then the default locale, en-US is set.
customAttribute	The name-value pairs of the custom attributes that have been set for the organization.

Fetching Organization Details

The retrieveOrg operation is used to read the details of an organization.

Note: If you want to fetch details of multiple organizations at a time, then use the listOrgs operation. See section, "Searching Organizations" (see page 57) for more information on how to use this.

This section walks you through the following topics for fetching the details of an organization:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The retrieveOrgRequest is used to fetch the details of an organization. The following table lists the elements of this request message:

Element	**Mandatory**	**Description**
orgName	Yes	The unique name with which the organization is identified.
clientTxId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch the organization details:

1. (Optional) Include the authentication and authorization details in the header of the retrieveOrg operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the retrieveOrgRequest elements for fetching the organization details.
3. Use retrieveOrgRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the retrieveOrg operation of the ArcorUserRegistryMgmtSvc service to fetch the organization details.

   This operation returns retrieveOrgResponse message that includes the transaction identifier, authentication token, and organization details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, retrieveOrgResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the organization details for a successful transaction and the fault response for an error condition.

See the table containing information about the listEmailTypeResponse response message for more information on the elements returned for a successful transaction. See appendix, "Error Codes" (see page 353) if there are any errors.

Searching Organizations

The listOrgs operation is used to read the details of multiple organizations. You can search organizations by their organization name, status, and partial or complete display name.

This section walks you through the following topics for searching organizations:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The listOrgsRequest is used to fetch the details of multiple organizations. The following table lists the elements of this request message:

Element	**Mandatory**	**Description**
namePattern	No	The search pattern that you want to use to search organizations. You can enter the partial or complete display name of an organization. If you enter the partial name, then all organizations with the display name matching the search pattern will be fetched.
orgName	No	The unique name with which the organization is identified. **Note:** If you want to search for more than one organization, then repeat this element for different organizations.
OrgStatus	No	The status of the organization in the database. Following are the supported values: ■ INITIAL ■ ACTIVE ■ INACTIVE ■ DELETED
clientTxId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To search multiple organizations:

1. (Optional) Include the authentication and authorization details in the header of the listOrgs operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the listOrgsRequest elements for fetching the organization details.
3. Use the listOrgsRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the listOrgs operation of the ArcorUserRegistryMgmtSvc service to fetch the organization details.

   This operation returns the listOrgsResponse message that includes the transaction identifier, authentication token, and organization details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, listOrgsResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the organization details for a successful transaction and the fault response for an error condition.

See the table containing information about the listEmailTypeResponse response message for more information on the elements returned for a successful transaction. See appendix, "Error Codes" (see page 353) if there are any errors.

Fetching AuthMinder Database Attributes

The listArcotAttributes operation is used to fetch the user attributes that are used to store the user information in the AuthMinder database.

This section walks you through the following topics for fetching user attributes supported by AuthMinder database:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The listArcotAttributesRequest message is used to fetch the user attributes. The following table lists the elements of this request message:

Element	**Mandatory**	**Description**
clientTxId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch the AuthMinder database attributes:

1. (Optional) Include the authentication and authorization details in the header of the listArcotAttributes operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the listArcotAttributesRequest elements to fetch the user attributes.
3. Use listArcotAttributesRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the listArcotAttributes operation of the ArcorUserRegistryMgmtSvc service to fetch the user attributes supported by AuthMinder database.

   This operation returns the listArcotAttributesResponse message that includes the transaction identifier, authentication token, and user attributes. See the following section for more information on the response message.

Interpreting the Response Message

The response message, listArcotAttributesResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the AuthMinder attributes for a successful transaction and the fault response for an error condition.

The following table provides more information about the elements returned for a successful transaction. See appendix, "Error Codes" (see page 353) if there are any errors.

Element	Description
Header Elements	
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
DATECREATED	The timestamp when the user account was created.
DATEMODIFIED	The timestamp when the user account was last modified.
EMAILADDR	The email address of the user.
FNAME	The first name of the user.

Element	Description
IMAGE	The personal assurance image that the user selected.
LNAME	The last name of the user.
MNAME	The middle name of the user.
PAM	The Personal Assurance Message (PAM) that is displayed when the user tries to access any AuthMinder-protected resource. PAM is the text string that serves as server verification to the client and is set by the user during enrollment.
PAMURL	The URL that lists the images, which can be used by the user to select their personal assurance image.
STATUS	The status of the user in the database. Following are the supported values: ■ INITIAL ■ ACTIVE ■ INACTIVE ■ DELETED
TELEPHONENUMBER	The telephone number of the user.
USERID	The unique identifier for the user.

Fetching Directory Service Attributes

The listRepositoryAttributes operation is used to fetch the directory service user attributes that are mapped to AuthMinder-supported user attributes.

This section walks you through the following topics for fetching the user attributes that the directory service supports:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The listRepositoryAttributesRequest message is used to fetch directory service user attributes that are mapped to AuthMinder-supported user attributes. The following table lists the elements of this request message:

Element	**Mandatory**	**Description**
repositoryType	Yes	The directory service where the user information resides. Possible values are: ■ ARUSER: For organizations that are created in the AuthMinder database. ■ LDAP: For organizations that are mapped with LDAP repository.
ldapDetails	No	The details of the directory service where the user information is available. See "ldapDetails" in the table containing information about the createOrgRequest message.
connectionCredential	No	The information required to connect to the directory service. See "connectionCredential" in the table containing information about the createOrgRequest message.
redirectSearchSchema	No	The schema that has to be used to search for the values whose attributes are in a different node.
redirectSearchAttribute	No	The value of the attribute that has to be searched in the redirectSearchSchema.
clientTxId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch the user attributes:

1. (Optional) Include the authentication and authorization details in the header of the listRepositoryAttributes operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the listRepositoryAttributesRequest elements to set the directory service information.
3. Use listRepositoryAttributesRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the listRepositoryAttributes operation of the ArcorUserRegistryMgmtSvc service to fetch the user attributes.

   This operation returns the listRepositoryAttributesResponse message that includes the transaction identifier, authentication token, and user attributes. See the following section for more information on the response message.

Interpreting the Response Message

The response message, listRepositoryAttributesResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the user attributes for a successful transaction, and the Fault response for an error condition.

The following table provides more information about the elements returned for a successful transaction. See to appendix, "Error Codes" (see page 353) if there are any errors.

Element	Description
Header Elements	
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
The user attributes used to store user information.	

Deleting Organizations

The deleteOrg operation is used to delete organizations in AuthMinder. After you delete an organization, the information related to that organization is still maintained in the system. Therefore, you cannot create an organization with the same name as that the of deleted organization.

This section walks you through the following topics for deleting organizations:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: After you delete an organization, you *must* refresh the system cache for the changes to take effect. See "Refreshing the Organization Cache" (see page 50) for more information on how to refresh the system cache.

Preparing the Request Message

The deleteOrgRequest message is used to delete organizations. The following table lists the elements of this request message:

Element	Mandatory	Description
orgName	Yes	The unique name with which the organization is identified.
clientTxId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To delete organizations:

1. (Optional) Include the authentication and authorization details in the header of the deleteOrg operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the deleteOrgRequest elements for deleting the organization details.
3. Use deleteOrgRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the deleteOrg operation of the ArcorUserRegistryMgmtSvc service to delete the organization.

   This operation returns the deleteOrgResponse message that includes the transaction identifier, authentication token, and organization details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, deleteOrgResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	Description
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Chapter 5: Configuration Management Web Service

Important! To use the Web service operations that are discussed in this chapter, you *must* deploy the User Data Service (**arcotuds.war**) file.
See section, "Deploying User Data Service" in the *CA AuthMinder Installation and Deployment Guide*.

This chapter describes the operations that are used to manage account types, fetch the email and telephone types configured for the users, and fetch the user attributes that are configured for encryption. This chapter covers the following topics:

- Managing Account Types (see page 68)
- Fetching Email and Telephone Types (see page 77)
- Fetching User Attributes Configured for Encryption (see page 81)

You must use the ArcotConfigManagementSvc.wsdl file to perform the operations discussed in this chapter.

Managing Account Types

All AuthMinder users are identified in the system by a unique user name. AuthMinder now supports the concept of an account or account ID, which is an alternate ID to identify the user in addition to the user name. A user can have none or one or more accounts or account IDs.

An account type is an attribute that qualifies the account ID and provides additional context about the usage of the account ID. To assign multiple accounts to a user, you must first create an account type, and then create an account for each account type.

For example, consider a financial institution that identifies the customers by their unique customer identifier. If the customer enhances their portfolio with a fixed deposit, then the financial institution can create an account type called *FIXED_DEPOSIT* and create an account in this account type with the fixed deposit number, for example 000203876544.

Now the customer can login either with their unique customer identifier or the account type and account ID (FIXED_DEPOSIT and 000203876544) combination.

You can configure the account type to be available to specific organizations only or to all organizations, including those that will be created in the future. At the organization level, each organization can choose to support a set of account types.

This section covers the following operations related to account type:

- Creating Account Types (see page 69)
- Updating Account Types (see page 71)
- Fetching Account Types (see page 73)
- Deleting Account Types (see page 75)

Creating Account Types

This section walks you through the following topics for creating account types:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: After you create an account type, you *must* refresh the system cache for the new account type to take effect. See "Refreshing the Organization Cache" (see page 50) for more information on how to refresh the cache.

Preparing the Request Message

The createAccountTypeRequest message is used to create account types in the AuthMinder database. The following table lists the elements of this request message:

Element	Mandatory	Description
accountType/name	Yes	The name of the account type that you want to create.
accountType/displayName	Yes	A descriptive name for the account type.
accountType/customAttribute	No	Name-value pairs that you can use to specify additional information related to account types.
targetAllOrgs	No	Indicates whether the account type should be assigned to all the organizations. Following are the supported values: ■ true: Account type is assigned to all the organizations. ■ false: Account type is assigned only to the organizations that are listed in the ListOfOrganizations element. **Note:** By default, the value of this element is set to false.
ListofOrganizations/Organization/ orgName	No	The name of the organization to which the account type must be assigned.
ListofOrganizations/Organization/ customAttribute	No	The custom attribute that you have set for the organization to which you want to assign the account type.

Element	Mandatory	Description
clientTxId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To create account types:

1. (Optional) Include the authentication and authorization details in the header of the createAccountType operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the createAccountTypeRequest elements to set the account information.
3. Use createAccountTypeRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the createAccountType operation of the ArcotConfigRegistrySvc service to create the account type.

   This operation returns the createAccountTypeResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, createAccountTypeResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	Description
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Updating Account Types

The updateAccountType operation is used to update the account type information and the list of organizations to which the account type belongs.

This section walks you through the following topics for updating existing account types:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: After you update an account type, you *must* refresh the system cache for the new account type to take effect. See "Refreshing the Organization Cache" (see page 50) for more information on how to refresh the cache.

Preparing the Request Message

The updateAccountTypeRequest message is used to update account types in the AuthMinder database. The following table lists the elements of this request message:

Element	Mandatory	Description
name	Yes	The name of the account type that you want to update.
displayName	No	The descriptive name of the account type.
customAttribute	No	Name-value pairs that contain the user or organization information that you want to update.
removeCustomAttribute	No	The name of the account type custom attribute that you want to delete.
targetAllOrgs	No	Indicates whether the updated account type should be assigned to all the organizations. Following are the supported values: ■ true: Updated account type is assigned to all the organizations. ■ false: Updated account type is assigned only to the organizations that are listed in the ListOfOrganizations element. **Note:** By default, the value of this element is set to false.
ListofOrganizations/orgName	No	The name of the organization to which the account type must be assigned.
ListofOrganizations/customAttribute	No	The custom attribute that you have specified for the organization.

Element	Mandatory	Description
RemoveOrganizations/orgName	No	The name of the organization that you want to disassociate with the account type.
clientTxId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To update account types:

1. (Optional) Include the authentication and authorization details in the header of the updateAccountType operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the updateAccountTypeRequest elements to set the account information.
3. Use updateAccountTypeRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the updateAccountType operation of the ArcotConfigRegistrySvc service to update the account type.

   This operation returns the updateAccountTypeResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, updateAccountTypeResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	Description
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Fetching Account Types

The listAccountTypes operation is used to fetch the account types that are associated with an organization.

This section walks you through the following topics for fetching the account types:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The listAccountTypeRequest message is used to fetch account types that are associated with an organization. The following table lists the elements of this request message:

Element	**Mandatory**	**Description**
targetAllOrgs	Yes	Indicates whether to fetch the account types assigned to all the organizations. Following are the supported values: ■ true: Account types assigned to all the organizations are fetched. ■ false: Account types assigned to the organizations that are listed in the orgName element are fetched.
orgName	No	The name of the organization to which the account types to be fetched belongs.
clientTxId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To list the account types of an organization:

1. (Optional) Include the authentication and authorization details in the header of the listAccountTypes operation. See chapter , "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the listAccountTypeRequest elements to set the account information.
3. Use listAccountTypeRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the listAccountTypes operation of the ArcotConfigRegistrySvc service to list the account types.

   This operation returns the listAccountTypeResponse message that includes the transaction identifier, authentication token, and the account types associated with an organization. See the following section for more information on the response message.

Interpreting the Response Message

The response message, listAccountTypeResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the account type details for a successful transaction, and the fault response for an error condition.

The following table provides more information about the elements returned for a successful transaction. See appendix "Error Codes" (see page 353) if there are any errors.

Element	Description
Header Elements	
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
AccountType/name	The name of the account type.

Element	Description
AccountType/displayName	The descriptive name of the account type.
AccountType/customAttribute	Name-value pairs that are used to specify additional account type information.

Deleting Account Types

The deleteAccountType operation is used to delete the account types that are associated with an organization.

This section walks you through the following topics for deleting account types:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: After you delete an account type, you *must* refresh the system cache for the deleted account type to take effect. See "Refreshing the Organization Cache" (see page 50) for more information on how to refresh the cache.

Preparing the Request Message

The deleteAccountTypeRequest message is used to delete account types in the AuthMinder database. The following table lists the elements of this request message:

Element	Mandatory	Description
accountType	Yes	The name of the account type that you want to delete.
clientTxId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To delete account types:

1. (Optional) Include the authentication and authorization details in the header of the deleteAccountType operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the deleteAccountTypeRequest elements to get the account type that has to be deleted.
3. Use the deleteAccountTypeRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the deleteAccountType operation of the ArcotConfigRegistrySvc service to delete the account type.

   This operation returns the deleteAccountTypeResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, deleteAccountTypeResponse, returns the transaction identifier and authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	**Description**
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Fetching Email and Telephone Types

AuthMinder enables you to specify multiple email addresses and telephone numbers while creating users in an organization. Email and telephone types are used to define multiple email addresses and telephone numbers. These types can be defined globally or can be specific to an organization. If the email address or telephone number types are mandatory for an organization, then you must provide these values while you create users in that organization.

This section covers the following sections that discuss how to fetch the email and telephone types that are configured for an organization:

- Fetching Email Types (see page 77)
- Fetching Telephone Types (see page 79)

Fetching Email Types

The listEmailTypes operation is used to fetch the email address types that are configured for an organization.

This section walks you through the following topics for fetching the email address types configured for an organization:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The listEmailTypeRequest message is used to fetch email address types that are configured for the organization. The following table lists the elements of this request message:

Element	**Mandatory**	**Description**
orgName	No	The name of the organization for which the email address types have to be fetched.
clientTxId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch email address types:

1. (Optional) Include the authentication and authorization details in the header of the listEmailTypes operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the listEmailTypeRequest elements to get the organization name.
3. Use the listEmailTypeRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the listEmailTypes operation of the ArcotConfigRegistrySvc service to fetch the email address types.

   This operation returns the listEmailTypeResponse message that includes the transaction identifier, authentication token, and email address types. See the following section for more information on the response message.

Interpreting the Response Message

The response message, listEmailTypeResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the email address types for a successful transaction and the Fault response for an error condition.

The following table provides more information about the elements returned for a successful transaction. See appendix, "Error Codes" (see page 353) if there are any errors.

Element	Description
Header Elements	
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
isGlobal	Specifies whether the email type is configured at the global level. Possible values are: ■ True: Indicates that the email type is configured at the global level. ■ False: Indicates that the email type is configured at the organization level.

Element	Description
emailType/name	The name of the email address type.
emailType/displayName	The display name of the email address type.
emailType/priority	The priority of the email type if more than one email type has been configured.
emailType/isMandatory	Indicates whether the email type is mandatory.

Fetching Telephone Types

The listTelephoneTypes operation is used to fetch the telephone types that are configured for an organization.

This section walks you through the following topics for fetching the telephone types configured for an organization:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The listTelephoneTypeRequest message is used to fetch the telephone types that are configured for the organization. The following table lists the elements of this request message:

Element	Mandatory	Description
orgName	No	The name of the organization for which the telephone address types have to be fetched.
clientTxId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch telephone types:

1. (Optional) Include the authentication and authorization details in the header of the listTelephoneTypes operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the listTelephoneTypeRequest elements to get the organization name.
3. Use the listTelephoneTypeRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the listTelephoneTypes operation of the ArcotConfigRegistrySvc service to fetch the telephone types.

   This operation returns the listTelephoneTypeResponse message that includes the transaction identifier, authentication token, and telephone types. See the following section for more information on the response message.

Interpreting the Response Message

The response message, listTelephoneTypeResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the telephone types for a successful transaction and the fault response for an error condition.

The following table provides more information about the elements returned for a successful transaction. See appendix, "Error Codes" (see page 353) if there are any errors.

Element	**Description**
Header Elements	
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
isGlobal	Specifies whether the telephone type is configured at the global level. Possible values are: ■ True: Indicates that the telephone type is configured at the global level. ■ False: Indicates that the telephone type is configured at the organization level.

Element	Description
TelephoneType/name	The name of the telephone type.
TelephoneType/displayName	The display name of the telephone type.
TelephoneType/priority	The priority of the telephone type if more than one telephone type has been configured.
TelephoneType/isMandatory	Indicates whether the telephone type is mandatory.

Fetching User Attributes Configured for Encryption

The administrators of an organization can choose to store the user attributes in an encrypted format. To fetch such attributes that are configured to be stored in encrypted format, you need to use the listConfiguredAttributesForEncryption operation.

This section walks you through the following topics for fetching the user attributes that are configured for encryption:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The listConfiguredAttributesForEncryptionRequest message is used to fetch the user attributes that are configured for encryption. The following table lists the elements of this request message:

Element	Mandatory	Description
orgName	No	The name of the organization for which the user attributes have to be fetched.
clientTxId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch user attributes configured for encryption:

1. (Optional) Include the authentication and authorization details in the header of the listConfiguredAttributesForEncryption operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the listConfiguredAttributesForEncryptionRequest elements to get the organization name.
3. Use the listConfiguredAttributesForEncryptionRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the listConfiguredAttributesForEncryption operation of the ArcotConfigRegistrySvc service to fetch the user attributes.

   This operation returns the listConfiguredAttributesForEncryptionResponse message that includes the transaction identifier, authentication token, and user attributes. See the following section for more information on the response message.

Interpreting the Response Message

The response message, listConfiguredAttributesForEncryptionResponse, returns the transaction identifier and authentication token in the SOAP envelope header. The SOAP body includes the user attributes configured for encryption for a successful transaction and the Fault response for an error condition.

The following table provides more information about the elements returned for a successful transaction. See appendix, "Error Codes" (see page 353) if there are any errors.

Element	**Description**
Header Elements	
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	

Element	Description
isGlobal	Specifies whether the user attributes are configured for encryption at the global level. Possible values are: ■ True: Indicates that the user attributes are configured for encryption at the global level. ■ False: Indicates that the user attributes are configured for encryption at the organization level.
attribute	The name of the user attribute that is configured for encryption.

Chapter 6: Managing Users and Accounts

Important! To use the Web service operations that are discussed in this chapter, you *must* deploy the User Data Service (**arcotuds.war**) file.
See section, "Deploying User Data Service" in the *CA AuthMinder Installation and Deployment Guide*.

For AuthMinder to authenticate users, users have to be created in the database, which is a one-time process. The user can either be created in the AuthMinder database or AuthMinder organization can be configured to connect to LDAP for user information.

This chapter discusses the Web service operations that are used to create and manage users, create and manager user accounts, and authenticate LDAP users. This chapter covers the following topics:

- Before You Proceed (see page 85)
- Performing User Operations (see page 89)
- Performing User Account Operations (see page 118)
- Setting the Personal Assurance Message (see page 135)
- Fetching the Personal Assurance Message (see page 137)
- Setting Custom User Attributes (see page 139)
- Authenticating LDAP Users (see page 141)

You must use the ArcotUserManagementSvc.wsdl file to perform the operations discussed in this chapter.

Before You Proceed

This section lists the supported user states, transitions supported between the user states, and the user operations that are possible on a particular organization and user status combination. Before you proceed with the user and user account operations that are discussed in this chapter, read this section to understand whether the operation can be performed based on the organization and user status.

The following topics are covered in this section:

- User States (see page 86)
- Supported User State Transitions (see page 86)
- User Operations and States (see page 87)
- User Account Operations and States (see page 88)

User States

AuthMinder supports the following states for users in the system:

- **INITIAL**

  Indicates that the user has been created in the system, but cannot perform any operation. To create a user in this state, you need to specify the status in the createUser operation.

- **ACTIVE**

  Indicates that the user can perform any operation in the system. This is the default status of the user when you create a user in the system.

- **INACTIVE**

  Indicates that the user has been deactivated and cannot perform any operation. You can deactivate a user permanently or for a specific period. You might need to deactivate the user for a specified period in situations where an employee goes for a long vacation and you want to disable their logins during this period to prevent any unauthorized access.

  To deactivate the user for a specific period, you must specify the startLockTime and endLockTime elements. If you do not specify these values, then the user will be permanently deactivated.

- **DELETED**

  Indicates that the user no longer exists in the system.

Supported User State Transitions

The following table lists the transitions possible between the supported user states:

Current State	**Change State to**				
	INITIAL	**ACTIVE**	**INACTIVE (Temporary)**	**INACTIVE (Permanent)**	**DELETED**
INITIAL	Yes	Yes	No	No	Yes
ACTIVE	No	Yes	Yes	Yes	Yes
INACTIVE	No	Yes	Yes	Yes	Yes
DELETED	No	No	No	No	Yes

User Operations and States

The following table lists the user operations and whether each operation is allowed on a specific combination of the organization and user status:

User Operation	Organization Status	User Status	Allowed
Create User	INITIAL	NA	No
	ACTIVE	NA	Yes
	INACTIVE	NA	No
	DELETED	NA	No
Update User	INITIAL	NA	No
	ACTIVE	Any User State	Yes
	INACTIVE	Any User State	Yes
	DELETED	Any User State	Yes
Update User Status	INITIAL	NA	No
	ACTIVE	ACTIVE INACTIVE DELETED	Yes
	INACTIVE	ACTIVE INACTIVE DELETED	Yes
	DELETED	Any User State	No
Delete User	INITIAL	NA	No
	ACTIVE	INITIAL ACTIVE INACTIVE	Yes
	INACTIVE	INITIAL ACTIVE INACTIVE	Yes
	DELETED	Any User State	No

User Account Operations and States

The following table lists the user account operations and whether each operation is allowed on a specific combination of the organization and user status:

User Account Operation	Organization Status	User Status	Allowed
Add User Account	INITIAL	NA	No
	ACTIVE/INACTIVE	INITIAL	Yes
		ACTIVE	Yes
		INACTIVE	Yes
		DELETED	No
	DELETED	Any User State	No
Update User Account	INITIAL	NA	No
	ACTIVE/INACTIVE	INITIAL	Yes
		ACTIVE	Yes
		INACTIVE	Yes
		DELETED	No
	DELETED	Any User State	No
Update User Account	INITIAL	NA	No
	ACTIVE/INACTIVE	INITIAL	Yes
		ACTIVE	Yes
		INACTIVE	Yes
		DELETED	No
	DELETED	Any User State	No
Delete User Account	INITIAL	NA	No
	ACTIVE/INACTIVE	INITIAL	Yes
		ACTIVE	Yes
		INACTIVE	Yes
		DELETED	No
	DELETED	Any User State	No

Performing User Operations

This section covers the following operations:

- Creating Users (see page 89)
- Updating Users (see page 94)
- Updating User Status (see page 98)
- Fetching User Details (see page 100)
- Searching Users by Using Pagination (see page 105)
- Searching All Users (see page 108)
- Checking User Status (see page 112)
- Updating User Status (see page 114)
- Deleting Users (see page 116)

Creating Users

This section walks you through the following topics for creating the users:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The createUserRequest message is used to create users in the AuthMinder database. The following table lists the elements of this request message:

Element	Mandatory	Description
userId/orgName	No	The name of the organization to which the user must belong to. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
userID/userName	Yes	The unique identifier with which the user is identified in the system.
userId/userRefId	No	The unique identifier that is assigned to the user when they are created. This identifier is used as a reference to track different operations performed by a user.

Element	Mandatory	Description
dateCreated	No	The timestamp when the user was created in the system. **Note:** Not applicable for the createUserRequest operation.
dateModified	No	The timestamp when the user details were last modified. **Note:** Not applicable for the createUserRequest operation.
emailId	Yes	The email ID of the user that has to be registered. The default qualifier is EMAILID. **Note:** You can repeat this entry if you want to configure multiple email IDs for a user, and accordingly use the qualifier based on the email types configured using Administration Console. Refer to the *CA AuthMinder Administration Guide* for more information on configuring multiple email IDs.
telephoneNumber	Yes	The telephone number of the user that has to be registered. The default qualifier is TELEPHONE. **Note:** You can repeat this entry if you want to configure multiple telephone numbers for a user, and accordingly use the qualifier based on the telephone types configured using Administration Console. Refer to the CA AuthMinder Administration Guide for more information on configuring multiple telephone numbers.
firstName	No	The first name of the user.
middleName	No	The middle name of the user.
lastName	No	The last name of the user.
pam	No	The Personal Assurance Message (PAM) that is displayed to the user when they try to access a resource protected by AuthMinder.
pamImageURL	No	The URL which contains the image that is displayed to the user, when they try to access a resource protected by AuthMinder.
image	No	The picture that the user wants to upload to identify themselves.
status	No	The status of the user. To create the user, the status must be ACTIVE.

Element	Mandatory	Description
customAttribute	No	The additional user information that you want to pass as a name-value pair. ■ name Indicates the name of the attribute that you want to create. ■ value Indicates the corresponding value for the name.
startLockTime	No	The timestamp when the user has to be deactivated.
endLockTime	No	The timestamp when the deactivated user has to be activated.
account/accountType	Yes *Only* if the account element is defined.	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
account/accountID	No	The alternate identifier that is used to identify the user in addition to the user name. The account ID is also known as account.
account/accountStatus	No	The status of the account. Following are the supported values: ■ 0-9: Indicates that the account is in the INITIAL state. ■ 10-19: Indicates that the account is in the ACTIVE state. ■ 20-29: Indicates that the account is in the INACTIVE state. ■ 30-39: Indicates that the account is in the DELETED state. ■ >39: Indicates that the account state is UNKNOWN.
account/accountIDAttribute	No	The alternate identifier that is used to identify the user in the system. **Note:** You *cannot* pass more than three account ID attributes for a user.
account/dateCreated	No	The timestamp when the account ID was created. **Note:** Not applicable for the createUserRequest operation.

Element	Mandatory	Description
account/dateModified	No	The timestamp when the account ID was last modified. **Note:** Not applicable for the createUserRequest operation.
account/accountCustomAttribute	No	The additional account information that you want to pass as a name-value pair. ■ attributeName Indicates the name of the attribute that you want to create. ■ attributeValue Indicates the corresponding value for the name.
clientTxId	No	The unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To create users in the AuthMinder database:

1. (Optional) Include the authentication and authorization details in the header of the createUser operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the createUserRequest elements to provide the user information.
3. Use the createUserRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the createUser operation of the ArcotUserRegistrySvc service to create users.

   This operation returns the createUserResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, createUserResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	**Description**
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Updating Users

This section walks you through the following topics for updating the user information:

- Preparing the Request Message
- Interpreting the Response Message
- Interpreting the Response Message

Preparing the Request Message

The updateUserRequest message is used to update the user information in the AuthMinder database. The following table lists the elements of this request message:

Element	Mandatory	Description
userId/orgName	No	The name of the organization to which the user belongs to. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
userID/userName	Yes	The unique identifier with which the user is identified in the system.
userId/userRefId	No	The identifier that is used as a reference to track different operations performed by a user.
dateCreated	No	The timestamp when the user was created in the system.
dateModified	No	The timestamp when the user details were last modified.
emailId	No	The email ID of the user that has to be registered. The default qualifier is EMAILID. **Note:** You can repeat this entry if you want to configure multiple email IDs for a user, and accordingly use the qualifier based on the configured email types.
telephoneNumber	No	The telephone number of the user that has to be registered. The default qualifier is TELEPHONE. **Note:** You can repeat this entry if you want to configure multiple telephone numbers for a user, and accordingly use the qualifier based on the configured telephone types.
firstName	No	The first name of the user.
middleName	No	The middle name of the user.
lastName	No	The last name of the user.

Element	Mandatory	Description
pam	No	The Personal Assurance Message (PAM) is a text string that is displayed to the user, when they try to access the AuthMinder-protected resource.
pamImageURL	No	The URL which contains the image that is displayed to the user when they try to access the AuthMinder-protected resource.
image	No	The picture that the user wants to upload to identify themselves.
status	No	The status of the user. Following are the supported values: ■ INITIAL ■ ACTIVE ■ INACTIVE ■ DELETED
customAttribute	No	The additional user information that you want to pass as a name-value pair. ■ name Indicates the name of the attribute that you want to create. ■ value Indicates the corresponding value for the name.
startLockTime	No	The timestamp when the user has to be deactivated.
endLockTime	No	The timestamp when the deactivated user has to be activated.
account/accountType	Yes *Only* if the account element is defined.	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
account/accountID	No	The alternate identifier that is used to identify the user in addition to the user name. The account ID is also known as account.

Element	Mandatory	Description
account/accountStatus	No	The status of the account. Following are the supported values: ■ 0-9: Indicates that the account is in the INITIAL state. ■ 10-19: Indicates that the account is in the ACTIVE state. ■ 20-29: Indicates that the account is in the INACTIVE state. ■ 30-39: Indicates that the account is in the DELETED state. ■ >39: Indicates that the account state is UNKNOWN.
account/accountIDAttribute	No	The alternate identifier that is used to identify the user in the system. **Note:** You *cannot* pass more than three account ID attributes for a user.
account/dateCreated	No	The timestamp when the account ID was created. **Note:** Not applicable for the updateUser operation.
account/dateModified	No	The timestamp when the account ID was last modified. **Note:** Not applicable for the updateUser operation.
account/accountCustomAttribute	No	The additional account information that you want to pass as a name-value pair. ■ attributeName Indicates the name of the attribute that you want to create. ■ attributeValue Indicates the corresponding value for the name.
updateUserFlags/updateImage	No	The flag to indicate whether the user image can be changed. Supported values are: ■ 0: Indicates that the image cannot be changed. ■ 1: Indicates that the image can be changed.
clientTxId	No	The unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To update users in the AuthMinder database:

1. (Optional) Include the authentication and authorization details in the header of the updateUser operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the updateUserRequest elements to update the user information.
3. Use the updateUserRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the updateUser operation of the ArcotUserRegistrySvc service to update user information.

   This operation returns the updateUserResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, updateUserResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	**Description**
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Updating User Status

The updateUserStatus operation is used to change the status of the user. In a single call, you can update the status of multiple users.

The status of a user can be any of the following:

- INITIAL
- ACTIVE
- INACTIVE
- DELETED

This section walks you through the following topics for changing the user status:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the updateUserStatusRequest message:

Element	Mandatory	Description
userId/orgName	No	The name of the organization to which the user belongs to. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
userID/userName	Yes	The unique identifier with which the user is identified in the system.
userId/userRefId	No	The identifier that is used as a reference to track different operations performed by a user.
Note: If want to update the status of more than one user, then repeat the userID element with the user details.		
status	Yes	The status that you want to assign to the user. Following are the supported values: ■ INITIAL ■ ACTIVE ■ INACTIVE ■ DELETED
startLockTime	No	The timestamp when the user has to be deactivated.

Element	Mandatory	Description
endLockTime	No	The timestamp when the deactivated user has to be activated.
clientTxId	No	The unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To update user status in the AuthMinder database:

1. (Optional) Include the authentication and authorization details in the header of the updateUserStatus operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the updateUserStatusRequest elements to update the user status.
3. Use the updateUserStatusRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the updateUserStatus operation of the ArcotUserRegistrySvc service to update the user status.

   This operation returns the updateUserStatusResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, updateUserStatusResponse returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	Description
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Fetching User Details

The retrieveUser operation is used to search the details of a particular user.

This section walks you through the following topics for fetching the user details:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the retrieveUserRequest message:

Element	**Mandatory**	**Description**
userIdentifier	Yes	The unique identifier (user name) with which the user is identified in the system.
orgName	No	The name of the organization to which the user belongs to. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
accountType	No	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
filter/includeImage	No	The flag to indicate whether the user image has to be retrieved or not. Supported values are: ■ 0: Indicates that the image must not be retrieved. This is the default value. ■ 1: Indicates that the image must be retrieved.
filter/includeAccounts	No	The flag to indicate whether the user accounts have to be retrieved or not. Supported values are: ■ 0: Indicates that the user accounts must not be retrieved. This is the default value. ■ 1: Indicates that the user accounts must be retrieved.

Element	Mandatory	Description
filter/deepSearch	No	The flag to indicate whether the user must be searched based on more than one parameter. Supported values are: ■ 0: Indicates that the users will be searched based on their user names *only*. ■ 1: Indicates that the users will be searched using the following details: First search attribute: User name Second search attribute: Account ID Third search attribute: Account ID attribute If the user details are not found using the first search attribute, then the second attribute is used. If both the first and second attributes fail to fetch the user details, then the third attribute is used to search the user details.
clientTxId	No	The unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To retrieve the details of a user:

1. (Optional) Include the authentication and authorization details in the header of the retrieveUser operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the retrieveUserRequest elements to collect the user details.
3. Use the retrieveUserRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the retrieveUser operation of the ArcotUserRegistrySvc service to fetch the user details.

   This operation returns the retrieveUserResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, retrieveUserResponse, returns the transaction identifier and authentication token in the SOAP envelope header. The SOAP body includes the user details for a successful transaction and the Fault response for an error condition.

The following table provides more information on the elements returned for a successful transaction. See appendix, "Error Codes" (see page 353) if there are any errors.

Element	Description
Header Elements	
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
userId/orgName	The name of the organization to which the user belongs to.
userId/userName	The unique identifier with which the user is identified in the system.
userId/userRefId	The identifier that is used as a reference to track different operations performed by a user.
dateCreated	The timestamp when the user was created in the system.

Element	Description
dateModified	The timestamp when the user details were last modified.
emailId	The email ID of the user that has been registered. If multiple email IDs are configured for the user, then all email IDs are fetched.
telephoneNumber	The telephone number of the user that has been registered. If multiple telephone numbers are configured for the user, then all numbers are fetched.
firstName	The first name of the user.
middleName	The middle name of the user.
lastName	The last name of the user.
pam	The Personal Assurance Message (PAM) string is displayed to the user, when they try to access a resource protected by AuthMinder.
pamImageURL	The URL which contains the image that is displayed to the user, when they try to access the AuthMinder-protected resource.
image	The picture that the user wants to upload to identify themselves.
status	The status of the user. Following are the supported values: ■ INITIAL ■ ACTIVE ■ INACTIVE ■ DELETED **Note:** If you do not pass the value, then by default the status is set as ACTIVE.
customAttribute	The additional user information in name-value pairs. ■ name Indicates the name of the attribute that you want to create. ■ value Indicates the corresponding value for the name.
startLockTime	The timestamp when the user was deactivated.
endLockTime	The timestamp when the deactivated user has to be activated.
account/accountType	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
account/accountID	The alternate identifier that is used to identify the user in addition to the user name. The account ID is also known as account.

Element	Description
account/accountStatus	The status of the account. Following are the supported values: ■ 0-9: Indicates that the account is in the INITIAL state. ■ 10-19: Indicates that the account is in the ACTIVE state. ■ 20-29: Indicates that the account is in the INACTIVE state. ■ 30-39: Indicates that the account is in the DELETED state. ■ >39: Indicates that the account state is UNKNOWN.
account/accountIDAttribute	The alternate identifier that is used to identify the user in the system.
account/dateCreated	The timestamp when the account ID was created.
account/dateModified	The timestamp when the account ID was last modified.
account/accountCustomAttribute	The additional account information that you want to pass as a name-value pair. ■ attributename Indicates the name of the custom attribute. ■ attributevalue Indicates the corresponding value for the name.

Searching Users by Using Pagination

When you search for users in the AuthMinder database or directory service, the information is fetched and displayed in the alphabetical order of the user names. If you have a large setup with many users, then to search for a user you will have to navigate through the search result to search for a particular user. To increase the search efficiency in such cases, you can search the users by specifying the start and end index.

Note: If you are searching for the users in the LDAP organization, then ensure that LDAP supports pagination search.

This section walks you through the following topics for searching the active users based on the search index:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the listUsersRequest message:

Element	**Mandatory**	**Description**
orgName	No	The name of the organization to which the user belongs to. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
startIndex	Yes	The index entry starting from which the user information has to be fetched. For example, if the complete search fetches 60 results and if the startIndex is set to 45, then the user information from search result entry 45 is returned.
endIndex	Yes	The index page where the user search must end. For example, if the complete search fetches 60 results and if the startIndex is set to 45 and endIndex is set to 55, then the user information from the search result entry 45 to 55 is returned.
clientTxId	No	The unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To search for users based on pagination:

1. (Optional) Include the authentication and authorization details in the header of the listUsers operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the listUsersRequest elements to collect the start and end index.
3. Use the listUsersRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the listUsers operation of the ArcotUserRegistrySvc service to fetch the user details for the specified start and end index.

   This operation returns the listUsersResponse message that includes the transaction identifier, authentication token, and user details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, listUsersResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the user details and status for a successful transaction and the Fault response for an error condition.

The following table provides more information about the return elements for successful transaction. See appendix, "Error Codes" (see page 353) if there are any errors.

Element	Description
Header Elements	
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
count	The total number of users returned in the search result.
userId/orgName	The name of the organization to which the user belongs.
userId/userName	The unique identifier with which the user is identified in the system.
userId/userRefId	The identifier that is used as a reference to track different operations performed by a user.

Element	**Description**
dateCreated	The timestamp when the user was created in the system.
dateModified	The timestamp when the user details were last modified.
emailId	The email ID of the user that has been registered. If multiple email IDs are configured for the user, then all the email IDs are fetched.
telephoneNumber	The telephone number of the user that has been registered. If multiple telephone numbers are configured for the user, then all the numbers are fetched.
firstName	The first name of the user.
middleName	The middle name of the user.
lastName	The last name of the user.
pam	The Personal Assurance Message (PAM) string displayed to the user when they try to access a resource protected by AuthMinder.
pamImageURL	The URL, which contains the image that is displayed to the user, when they try to access a resource protected by AuthMinder.
image	The picture that the user wants to upload to identify themselves.
status	The status of the user. Following are the supported values: ■ INITIAL ■ ACTIVE ■ INACTIVE ■ DELETED
customAttribute	The additional user information that you want to pass as name-value pair. ■ name Indicates the name of the attribute that you want to create. ■ value Indicates the corresponding value for the name.
startLockTime	The timestamp when the user has to be deactivated.
endLockTime	The timestamp when the deactivated user has to be activated.

Searching All Users

You must use the searchUsers operation to search for all the users in the system.

This section walks you through the following topics for searching the users:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the searchUsers message:

Element	**Mandatory**	**Description**
orgPattern	No	The pattern that is used to search the organizations. For example, *ac pattern will search for users in the following organizations: ■ Acme ■ Acro **Note:** If the organization pattern is not passed, then the Default Organization is used for the operation.
orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
searchExpression	Yes	The expression that is used to search for users. For example, if you search for *m, then the following user details will be fetched: ■ John Smith ■ Mathew
count	No	If the search result exceeds this value, then only the search results equal to this value are fetched.
filter/includeImage	No	The flag to indicate whether the user image has to be retrieved or not. Supported values are: ■ 0: Indicates that the image must not be retrieved. ■ 1: Indicates that the image must be retrieved.

Element	**Mandatory**	**Description**
filter/includeAccounts	No	The flag to indicate whether the user accounts have to be retrieved or not. Supported values are: ■ 0: Indicates that the user accounts must not be retrieved. ■ 1: Indicates that the user accounts must be retrieved.
filter/deepSearch	No	The flag to indicate whether the user must be searched based on more than one parameter. Supported values are: ■ 0: Indicates that the users will be searched based on their user names *only*. ■ 1: Indicates that the users will be searched using the following details: **First search attribute**: User name **Second search attribute**: Account ID **Third search attribute**: Account ID attributes If the user details are not found using the first search attribute, then the second attribute is used. If both the first and second attributes fail to fetch the user details, then the third attribute is used to search the user details.
status	No	The status of the user. Following are the supported values: ■ INITIAL ■ ACTIVE ■ INACTIVE ■ DELETED **Note:** If you do not pass the value, then by default the status is set as ACTIVE.
account/accountType	Yes *Only* if the account element is defined.	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
account/accountID	No	The alternate identifier that is used to identify the user in addition to the user name. The account ID is also known as account.

Element	Mandatory	Description
account/accountStatus	No	The status of the account. Following are the supported values: ■ 0-9: Indicates that the account is in the INITIAL state. ■ 10-19: Indicates that the account is in the ACTIVE state. ■ 20-29: Indicates that the account is in the INACTIVE state. ■ 30-39: Indicates that the account is in the DELETED state. ■ >39: Indicates that the account state is UNKNOWN.
account/accountIDAttribute	No	The alternate identifier that is used to identify the user in the system. **Note:** You *cannot* pass more than three account ID attributes for a user.
account/dateCreated	No	The timestamp when the account ID was created.
account/dateModified	No	The timestamp when the account ID was last modified.
account/accountCustomAttribute	No	The additional account information that you want to pass as a name-value pair. ■ attributeName Indicates the name of the attribute that you want to create. ■ attributeValue Indicates the corresponding value for the name.
RepositoryUserAttributes/attributeName	No	The name of the user attribute that is used to store the user information. For example, First Name or Email Address.
clientTxId	No	The unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To search for users:

1. (Optional) Include the authentication and authorization details in the header of the searchUsers operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the searchUsersRequest elements to collect the user information.
3. Use the searchUsersRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the searchUsers operation of the ArcotUserRegistrySvc service to fetch the information of all the users.

   This operation returns the searchUsersResponse message that includes the transaction identifier, authentication token, and user details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, searchUsersResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the user details and status for a successful transaction and the Fault response for an error condition.

The return elements of searchUsersResponse are the same as those for retrieveUserResponse. See the table containing information about the retrieveUserResponse elements for more information on the user details that is returned for a successful transaction. See appendix, "Error Codes" (see page 353) if there are any errors.

Fetching User Status

You must use the getUserStatus operation to know the current status of the user in the database.

This section walks you through the following topics for checking the user status:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the getUserStatusRequest message:

Element	Mandatory	Description
userId/orgName	No	The name of the organization to which the user belongs to. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
userId/userName	Yes	The unique identifier with which the user is identified in the system.
userId/userRefId	No	The identifier that is used as a reference to track different operations performed by a user.
clientTxId	No	The unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To check the user status:

1. (Optional) Include the authentication and authorization details in the header of the getUserStatus operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the getUserStatusRequest elements to collect the user details.
3. Use the getUserStatusRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the getUserStatus operation of the ArcotUserRegistrySvc service to check the user status.

   This operation returns the getUserStatusResponse message that includes the transaction identifier, authentication token, and user details and status. See the following section for more information on the response message.

Interpreting the Response Message

The response message, getUserstatusResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the user details and status for a successful transaction and the Fault response for an error condition.

The following table provides more information about the return elements for successful transaction. See appendix, "Error Codes" (see page 353) if there are any errors.

Element	Description
Header Elements	
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
userId/orgName	The name of the organization to which the user belongs.
userId/userName	The unique identifier with which the user is identified in the system.
userId/userRefId	The identifier that is used as a reference to track different operations performed by a user.

Element	Description
status	The status of the user. Following are the supported values: ■ INITIAL ■ ACTIVE ■ INACTIVE ■ DELETED

Updating User Status

You must use the updateUserStatus operation to change the current status of the user in the database.

This section walks you through the following topics for updating the user status:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the updateUserStatusRequest message:

Element	Mandatory	Description
userId/orgName	No	The name of the organization to which the user belongs **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
userId/userName	Yes	The unique identifier with which the user is identified in the system.
userId/userRefId	No	The identifier that is used as a reference to track different operations performed by a user.
status	Yes	The status that you want to assign to the user. Following are the supported values: ■ INITIAL ■ ACTIVE ■ INACTIVE **Note:** If the current status of the user is DELETED, then you cannot update the status of that user.
startLockTime	No	The timestamp when the user has to be deactivated.

Element	Mandatory	Description
endLockTime	No	The timestamp when the deactivated user has to be activated.
clientTxId	No	The unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To update the user status:

1. (Optional) Include the authentication and authorization details in the header of the updateUserStatus operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the updateUserStatusRequest elements to collect the user details.
3. Use the updateUserStatusRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the updateUserStatus operation of the ArcotUserRegistrySvc service to check the user status.

   This operation returns the updateUserStatusResponse message that includes the transaction identifier and the authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, updateUserStatusResponse returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	Description
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Deleting Users

This section walks you through the following topics for deleting users:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The deleteUserRequest message is used to delete users in the AuthMinder database. The following table lists the elements of this request message:

Element	Mandatory	Description
userId/orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
userID/userName	Yes	The unique identifier with which the user is identified in the system.
userId/userRefId	No	The unique identifier that is assigned to the user when they are created.
clientTxId	No	The unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To delete users:

1. (Optional) Include the authentication and authorization details in the header of the deleteUser operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the deleteUserRequest elements to provide the user information, as listed in the table shown in the preceding section.
3. Use the deleteUserRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the deleteUser operation of the ArcotUserRegistrySvc service to delete users.

   This operation returns the deleteUserResponse message that includes the transaction identifier and authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, deleteUserResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See chapter, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	**Description**
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Performing User Account Operations

In addition to the user name, which is the unique user identifier, users can also be identified by their accounts (also known as account ID*).* A user can have none or one or multiple accounts. To define an account for the user, an account type has to be first configured for the organization to which the user belongs.

An account type provides additional context about the usage of the account. An account type can have only *one* account ID. If you want to assign multiple account IDs for a user, then you need to first configure the account type for each account ID that you plan to create for the user.

This section covers the following topics related to user account operations:

- Adding User Accounts (see page 119)
- Updating User Accounts (see page 122)
- Fetching All Accounts of a User (see page 124)
- Fetch a User Account Details (see page 126)
- Fetching User Details Using Accounts (see page 129)
- Deleting User Accounts (see page 133)

Note: Account are dependent on user name and account type. Before adding user accounts, you must ensure that the user has already been created in the system, as discussed in the "Performing User Operations" (see page 89) section, and that the account type has been defined for the organization to which the user belongs to, as discussed in chapter, "Configuration Management Web Service" (see page 67).

Adding User Accounts

You must use the addUserAccount operation to add accounts for the users. This section walks you through the following topics for adding user accounts.

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the addUserAccountRequest message:

Element	Mandatory	Description
userId/orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
userId/userName	Yes	The unique identifier with which the user is identified in the system.
userId/userRefId	No	The identifier that is used as a reference to track different operations performed by a user.
account/accountType	Yes	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
account/accountID	No	The alternate identifier that is used to identify the user in addition to the user name. The account ID is also known as account.
account/accountStatus	No	The status of the account. Following are the supported values: ■ 0-9: Indicates that the account is in the INITIAL state. ■ 10-19: Indicates that the account is in the ACTIVE state. ■ 20-29: Indicates that the account is in the INACTIVE state. ■ 30-39: Indicates that the account is in the DELETED state. ■ >39: Indicates that the account state is UNKNOWN.

Element	Mandatory	Description
account/accountIDAttribute	No	The alternate identifier that is used to identify the user in the system. **Note:** You *cannot* pass more than three account ID attributes for a user.
account/dateCreated	No	The timestamp when the account ID was created.
account/dateModified	No	The timestamp when the account ID was last modified.
account/accountCustomAttribute	No	The additional account information that you want to pass as a name-value pair. ■ attributeName Indicates the name of the attribute that you want to create. ■ attributeValue Indicates the corresponding value for the name.
clientTxId	No	The unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To add user accounts:

1. (Optional) Include the authentication and authorization details in the header of the addUserAccount operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the addUserAccountRequest elements to collect the user details that are listed in the preceding section.
3. Use the addUserAccountRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the addUserAccount operation of the ArcotUserRegistrySvc service to add accounts for the user.

   This operation returns the addUserAccountResponse message that includes the transaction identifier, authentication token, and user account details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, addUserAccountResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	**Description**
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Updating User Accounts

You must use the updateUserAccount operation to update the existing accounts of the users. This section walks you through the following topics for updating the user accounts.

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The updateUserAccountRequest message elements are same as those for addUserAccountRequest (see page 119).

Invoking the Web Service

To update user accounts:

1. (Optional) Include the authentication and authorization details in the header of the updateUserAccount operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the updateUserAccountRequest elements to collect the user account details, as listed in Adding User Accounts (see page 119).
3. Use the updateUserAccountRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the updateUserAccount operation of the ArcotUserRegistrySvc service to update accounts of the user.

   This operation returns the updateUserAccountResponse message that includes the transaction identifier, authentication token, and user account details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, updateUserAccountResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	**Description**
udsTransactionID	The unique identifier of the transaction that is performed using UDS.

Element	Description
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Fetching All Accounts of a User

To fetch the details of all accounts that are created for a user, you must use the listUserAccounts operation. This section walks you through the following topics for fetching the details of the user accounts.

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: If you want to fetch details of a particular account, then use the retrieveUserAccount operation. See "Fetch a User Account Details" (see page 126) for more information.

Preparing the Request Message

The listUserAccountRequest message elements are same as those for addUserAccountRequest. See Adding User Accounts (see page 119) for more information.

Invoking the Web Service

To fetch user accounts:

1. (Optional) Include the authentication and authorization details in the header of the listUserAccounts operation. See "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the listUserAccountRequest elements to collect the user account details, as listed in Adding User Accounts (see page 119).
3. Use the listUserAccountRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the listUserAccounts operation of the ArcotUserRegistrySvc service to fetch the account details of the user.

   This operation returns the listUserAccountResponse message that includes the transaction identifier, authentication token, and user account details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, listUserAccountResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the user account details for a successful transaction and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. See "Error Codes" (see page 353) if there are any errors.

Element	Description
Header Elements	
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
account/accountType	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
account/accountID	The alternate identifier that is used to identify the user in addition to the user name. The account ID is also known as account.
account/accountStatus	The status of the account. Following are the supported values: ■ 0-9: Indicates that the account is in the INITIAL state. ■ 10-19: Indicates that the account is in the ACTIVE state. ■ 20-29: Indicates that the account is in the INACTIVE state. ■ 30-39: Indicates that the account is in the DELETED state. ■ >39: Indicates that the account state is UNKNOWN.
account/accountIDAttribute	The alternate identifier that is used to identify the user in the system.
account/dateCreated	The timestamp when the account ID was created.
account/dateModified	The timestamp when the account ID was last modified.
account/accountCustomAttribute	The additional account information that you want to pass as a name-value pair. ■ attributeName Indicates the name of the attribute that you want to create. ■ attributeValue Indicates the corresponding value for the name.

Fetch a User Account Details

You must use the retrieveUserAccount operation to fetch the details of a particular user account.

This section walks you through the following topics to fetch a single account of a user:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the retrieveUserAccountRequest message:

Element	**Mandatory**	**Description**
userId/orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
userId/userName	Yes	The unique identifier with which the user is identified in the system.
userId/userRefId	No	The identifier that is used as a reference to track different operations performed by a user.
account/accountType	Yes	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
account/accountID	No	The alternate identifier that is used to identify the user in addition to the user name. The account ID is also known as account.
account/accountStatus	No	The status of the account. Following are the supported values: ■ 0-9: Indicates that the account is in the INITIAL state. ■ 10-19: Indicates that the account is in the ACTIVE state. ■ 20-29: Indicates that the account is in the INACTIVE state. ■ 30-39: Indicates that the account is in the DELETED state. ■ >39: Indicates that the account state is UNKNOWN.

Element	**Mandatory**	**Description**
account/accountIDAttribute	No	The alternate identifier that is used to identify the user in the system. **Note:** You *cannot* pass more than three account ID attributes for a user.
account/dateCreated	No	The timestamp when the account ID was created.
account/dateModified	No	The timestamp when the account ID was last modified.
account/accountCustomAttribute	No	The additional account information that you want to pass as a name-value pair. ■ attributeName Indicates the name of the attribute that you want to create. ■ attributeValue Indicates the corresponding value for the name.
clientTxId	No	The unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch the user account details:

1. (Optional) Include the authentication and authorization details in the header of the retrieveUserAccount operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the retrieveUserAccountRequest elements to collect the user and account details, as listed in the preceding table.
3. Use the retrieveUserAccountRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the retrieveUserAccount operation of the ArcotUserRegistrySvc service to fetch the user details based on the account information.

   This operation returns the retrieveUserAccountResponse message that includes the transaction identifier, authentication token, and user account details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, retrieveUserAccountResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the user account details for a successful transaction and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. See appendix, "Error Codes" (see page 353) if there are any errors.

Element	**Description**
Header Elements	
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
account/accountType	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
account/accountID	The alternate identifier that is used to identify the user in addition to the user name. The account ID is also known as account.

Element	Description
account/accountStatus	The status of the account. Following are the supported values: ■ 0-9: Indicates that the account is in the INITIAL state. ■ 10-19: Indicates that the account is in the ACTIVE state. ■ 20-29: Indicates that the account is in the INACTIVE state. ■ 30-39: Indicates that the account is in the DELETED state. ■ >39: Indicates that the account state is UNKNOWN.
account/accountIDAttribute	The alternate identifier that is used to identify the user in the system.
account/dateCreated	The timestamp when the account ID was created.
account/dateModified	The timestamp when the account ID was last modified.
account/accountCustomAttribute	The additional account information that you want to pass as a name-value pair. ■ attributeName Indicates the name of the attribute that you want to create. ■ attributeValue Indicates the corresponding value for the name.

Fetching User Details Using Accounts

To fetch the user details using their account information, you must use the listUsersForAccount operation. This section walks you through the following topics for fetching the user information based on the user accounts:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the listUsersForAccountRequest message:

Element	Mandatory	Description
orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.

Element	Mandatory	Description
accountType	No	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
Note: The accountID and accountIDAttribute elements are optional, but you must pass at least one element.		
accountID	No	The alternate identifier that is used to identify the user in addition to the user name. The account ID is also known as account.
accountIDAttribute	No	The alternate identifier that is used to identify the user in the system. **Note:** You *cannot* pass more than three account ID attributes for a user.
filter/includeImage	No	The flag to indicate whether the user image has to be retrieved or not. Supported values are: ■ 0: Indicates that the image must not be retrieved. ■ 1: Indicates that the image must be retrieved.
filter/includeAccounts	No	The flag to indicate whether the user accounts have to be retrieved or not. Supported values are: ■ 0: Indicates that the user accounts must not be retrieved. ■ 1: Indicates that the user accounts must be retrieved.
clientTxId	No	The unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch the user details using their account information:

1. (Optional) Include the authentication and authorization details in the header of the listUsersForAccount operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the listUsersForAccountRequest elements to collect the user account information, as listed in the preceding table.
3. Use the listUsersForAccountRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the listUsersForAccount operation of the ArcotUserRegistrySvc service to fetch the user details based on the account information.

   This operation returns listUsersForAccountResponse message that includes the transaction identifier, authentication token, and user details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, listUsersForAccountResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the user details for a successful transaction and the fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. See appendix, "Error Codes" (see page 353) if there are any errors.

Element	Description
Header Elements	
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
userId/orgName	The name of the organization to which the user belongs.
userId/userName	The unique identifier with which the user is identified in the system.
userId/userRefId	The identifier that is used as a reference to track different operations performed by a user.
dateCreated	The timestamp when the user was created in the system.

Element	**Description**
dateModified	The timestamp when the user details were last modified.
emailId	The email ID of the user that has been registered. If multiple email IDs are configured for the user, then all the email IDs are fetched.
telephoneNumber	The telephone number of the user that has been registered. If multiple telephone numbers are configured for the user, then all the numbers are fetched.
firstName	The first name of the user.
middleName	The middle name of the user.
lastName	The last name of the user.
pam	The Personal Assurance Message (PAM) string that is displayed to the user when they try to access a resource protected by AuthMinder.
pamImageURL	The URL, which contains the image that is displayed to the user when they try to access a resource protected by AuthMinder.
image	The picture that the user wants to upload to identify themselves.
status	The status of the user. Following are the supported values: ■ INITIAL ■ ACTIVE ■ INACTIVE ■ DELETED
customAttribute	The additional user information that you want to pass as a name-value pair. ■ name Indicates the name of the attribute that you want to create. ■ value Indicates the corresponding value for the name.
startLockTime	The timestamp when the user has to be deactivated.
endLockTime	The timestamp when the deactivated user has to be activated.
account/accountType	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.
account/accountID	The alternate identifier that is used to identify the user in addition to the user name. The account ID is also known as account.

Element	Description
account/accountStatus	The status of the account. Following are the supported values: ■ 0-9: Indicates that the account is in the INITIAL state. ■ 10-19: Indicates that the account is in the ACTIVE state. ■ 20-29: Indicates that the account is in the INACTIVE state. ■ 30-39: Indicates that the account is in the DELETED state. ■ >39: Indicates that the account state is UNKNOWN.
account/accountIDAttribute	The alternate identifier that is used to identify the user in the system.
account/dateCreated	The timestamp when the account ID was created.
account/dateModified	The timestamp when the account ID was last modified.
account/accountCustomAttribute	The additional account information that you want to pass as a name-value pair. ■ attributeName Indicates the name of the attribute that you want to create. ■ attributeValue Indicates the corresponding value for the name.

Deleting User Accounts

You must use the deleteUserAccount operation to delete accounts for the users. This section walks you through the following topics for deleting user accounts:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the deleteUserAccountRequest message:

Element	Mandatory	Description
accountType	Yes	The attribute that qualifies the account ID and provides additional context about the usage of the account ID.

Element	Mandatory	Description
clientTxId	No	The unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To delete user accounts:

1. (Optional) Include the authentication and authorization details in the header of the deleteUserAccount operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the deleteUserAccountRequest elements to collect the user details, as listed in the preceding table.
3. Use the deleteUserAccountRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the deleteUserAccount operation of the ArcotUserRegistrySvc service to delete accounts for the user.

   This operation returns the deleteUserAccountResponse message that includes the transaction identifier, authentication token, and user account details. See the following section for more information on the response message.

Interpreting the Response Message

The response message, deleteUserAccountResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	Description
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Setting the Personal Assurance Message

The Personal Assurance Message (PAM) is a text string that is displayed to the user, when they try to access a resource protected by AuthMinder. This string assures the user that they are connected to the genuine network or resource.

To set the PAM for a user, you must use the setPAM operation. This section walks you through the following topics for setting the PAM for users:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the setPAMRequest message:

Element	**Mandatory**	**Description**
UserId/orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
UserId/userName	Yes	The unique identifier with which the user is identified in the system.
UserId/userRefId	No	The identifier that is used as a reference to track different operations performed by a user.
PAM	No	The Personal Assurance Message (PAM) string displayed to the user when they try to access a resource protected by AuthMinder. **Note:** If you do not pass the PAM element, then an empty value will be set as PAM.
pamImageURL	No	The URL, which contains the image that is displayed to the user when they try to access a resource protected by AuthMinder.
clientTxId	No	The unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To set the PAM for a user:

1. (Optional) Include the authentication and authorization details in the header of the setPAM operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the setPAMRequest elements to collect the user information, as listed in the preceding table.
3. Use the setPAMRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the setPAM operation of the ArcotUserRegistrySvc service to set the PAM for the user.

   This operation returns the setPAMResponse message that includes the transaction identifier and authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, setPAMResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	**Description**
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Fetching the Personal Assurance Message

To read the PAM that is set for a user, you must use the getPAM operation. This section walks you through the following topics for fetching the PAM of the users:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the getPAMRequest message:

Element	Mandatory	Description
UserId/orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
UserId/userName	Yes	The unique identifier with which the user is identified in the system.
UserId/userRefId	No	The identifier that is used as a reference to track different operations performed by a user.
clientTxId	No	The unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch the PAM of a user:

1. (Optional) Include the authentication and authorization details in the header of the getPAM operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the getPAMRequest elements to collect the user information, as listed in the preceding table.
3. Use the getPAMRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the getPAM operation of the ArcotUserRegistrySvc service to get the PAM for the user.

   This operation returns the getPAMResponse message that includes the transaction identifier, authentication token, and PAM. See the following section for more information on the response message.

Interpreting the Response Message

The response message, getPAMResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the PAM for a successful transaction and the fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. See appendix, "Error Codes" (see page 353) if there are any errors.

Element	Description
Header Elements	
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
UserId/orgName	The name of the organization to which the user belongs.
UserId/userName	The unique identifier with which the user is identified in the system.
UserId/userRefId	The identifier that is used as a reference to track different operations performed by a user.

Element	Description
PAM	The Personal Assurance Message (PAM) that is displayed to the user when they try to access a resource protected by AuthMinder.
pamImageURL	The URL which contains the image that is displayed to the user when they try to access a resource protected by AuthMinder.

Setting Custom User Attributes

In addition to the standard user information that AuthMinder supports, you can set additional user information by using custom attributes. You must pass the additional information as name-value pairs.

To set the custom user attributes, you must use the setCustomAttributes operation. This section walks you through the following topics for setting custom attributes:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the setCustomAttributesRequest message:

Element	Mandatory	Description
UserId/orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
UserId/userName	Yes	The unique identifier with which the user is identified in the system.
UserId/userRefId	No	The identifier that is used as a reference to track different operations performed by a user.
customAttribute	No	The additional user information that you want to pass as a name-value pair. ■ name Indicates the name of the attribute that you want to create. ■ value Indicates the corresponding value for the name.

Element	Mandatory	Description
clientTxId	No	The unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To set additional information for an user:

1. (Optional) Include the authentication and authorization details in the header of the setCustomAttributes operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the setCustomAttributesRequest elements to collect the user information, as listed in the preceding table.
3. Use the setCustomAttributesRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the setCustomAttributes operation of the ArcotUserRegistrySvc service to set the user information.

   This operation returns the setCustomAttributesResponse message that includes the transaction identifier and authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, setCustomAttributesResponse, returns the transaction identifier and authentication token in the SOAP envelope header. These elements are explained in the following table. The SOAP body returns a success message if the operation was performed successfully. If there are any errors, then the Fault response is returned. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	Description
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.

Authenticating LDAP Users

This section discusses the operations used to authenticate users whose accounts are present in the directory service. It covers the following topics:

- Using the LDAP Password (see page 141)
- Using Directory Service Attributes (see page 143)

Important! The operations discussed in this section are applicable *only* for organizations with repository type as **LDAP**.

Using the LDAP Password

Administration Console supports LDAP authentication as one of authentication mechanisms for users to log in to Administration Console. In this case, users log in to the Console by specifying their LDAP user name and password.

To use the LDAP authentication mechanism to authenticate users, you must use the authenticateUser operation. This section walks you through the following topics for authenticating users using **LDAP User Password** authentication mechanism:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the authenticateUserRequest message:

Element	**Mandatory**	**Description**
UserCredential/userId/orgName	No	The name of the organization to which the user belongs. **Note:** If the organization name is not passed, then the Default Organization is used for the operation.
UserCredential/userId/userName	Yes	The unique identifier with which the user is identified in the system.
UserCredential/userId/userRefId	No	The identifier that is used as a reference to track different operations performed by a user.
UserCredential/userCredential/type	Yes	The credential that has to be used to authenticate the user. You must set the type as password.

Element	Mandatory	Description
clientTxId	No	The unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To authenticate users using the LDAP authentication mechanism:

1. (Optional) Include the authentication and authorization details in the header of the authenticateUser operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the authenticateUserRequest elements to collect the user and credential information, as listed in the preceding table.
3. Use the authenticateUserRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the authenticateUser operation of the ArcotUserRegistrySvc service to set the user information.

   This operation returns the authenticateUserResponse message that includes the transaction identifier and authentication token. See the following section for more information on the response message.

Interpreting the Response Message

The response message, authenticateUserResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the authentication status for a successful transaction, and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. Refer to appendix, "Error Codes" (see page 353) if there are any errors.

Element	Description
Header Elements	
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	

Element	Description
AuthResult/status	The authentication status of the user in the LDAP. Possible values are: ■ SUCCESS ■ FAILURE

Using Directory Service Attributes

This section discusses the operations that are used to authenticate users using their directory service attributes:

- Fetching User Attributes (see page 143)
- Fetching User Attribute Values (see page 145)
- Verifying User Attributes (see page 147)

Fetching User Attributes

The attributes (with string or integer value) that are used to store the user information in the directory service can be read using the getQnAAttributes operation. This section walks you through the following topics related to this operation:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the QnAAttributesRequest message:

Element	Mandatory	Description
orgName	Yes	The name of the LDAP organization to which the user attributes that you want to fetch belongs.
clientTxId	No	The unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch the user attributes:

1. (Optional) Include the authentication and authorization details in the header of the getQnAAttributes operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the getQnAAttributesRequest elements to collect the organization information, as listed in the preceding table.
3. Use the QnAAttributesRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the getQnAAttributes operation of the ArcotUserRegistrySvc service to fetch the user attributes of the LDAP organization.

   This operation returns the QnAAttributesResponse message that includes the transaction identifier, authentication token, and user attributes. See the following section for more information on the response message.

Interpreting the Response Message

The response message, QnAAttributesResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the user attributes for a successful transaction and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. See appendix, "Error Codes" (see page 353) if there are any errors.

Element	Description
Header Elements	
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
The user attributes configured in the LDAP.	

Fetching User Attribute Values

The getQnAValues operation is used to read the values that are set for the user attributes present in the directory service. You can fetch the values for one or more attributes. This section walks you through the following topics related to this operation:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the QnAValuesRequest message:

Element	**Mandatory**	**Description**
username	Yes	The unique identity of the user whose attribute values you want to fetch.
orgname	Yes	The name of the LDAP organization to which the user attribute values that you want to fetch belongs.
attributes/attribute	Yes	The name of the attributes whose value you want to fetch.
clientTxId	No	The unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To fetch the values of user attributes:

1. (Optional) Include the authentication and authorization details in the header of the getQnAValues operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the getQnAValuesRequest elements to collect the user, organization, and attribute information, as listed in the preceding table.
3. Use the QnAValuesRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the getQnAValues operation of the ArcotUserRegistrySvc service to fetch the values of the user attributes that are stored in directory service.

   This operation returns the QnAValuesResponse message that includes the transaction identifier, authentication token, and attribute values. See the following section for more information on the response message.

Interpreting the Response Message

The response message, QnAValuesResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the user attribute values for a successful transaction and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. See appendix, "Error Codes" (see page 353) if there are any errors.

Element	**Description**
Header Elements	
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
The values for the user attributes that are configured in the directory service.	

Verifying User Attributes

You can authenticate the users of an organization (mapped to LDAP repository) by using their LDAP attributes. You must use the performQnAVerification operation to perform this authentication. This section walks you through the following topics related to this operation:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The following table lists the elements of the QnAVerificationRequest message:

Element	**Mandatory**	**Description**
username	Yes	The unique identifier of the user whose attributes you want to verify.
orgname	Yes	The name of the LDAP organization to which the user belongs to.
attributes/attribute	Yes	The name (attrName) and value (attrValue) of the attribute that has to be verified.
ignorecase	Yes	Specifies whether the case of the attribute values passed in the input must match the case of the values stored in the directory service. Possible values are: ■ 0: Indicates that the case must match. ■ 1: Indicates that the case of the input values will be ignored.
clientTxId	No	The unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Invoking the Web Service

To authenticate users with their LDAP attributes:

1. (Optional) Include the authentication and authorization details in the header of the performQnAVerification operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. Use the performQnAVerificationRequest elements to collect the user, organization, and attribute information, as listed in the preceding table.
3. Use the QnAVerificationRequest message and construct the input message by using the details obtained in preceding step.
4. Invoke the performQnAVerification operation of the ArcotUserRegistrySvc service to fetch the values of the user attributes that are stored in directory service.

   This operation returns the QnAVerificationResponse message that includes the transaction identifier, authentication token, and verification result. See the following section for more information on the response message.

Interpreting the Response Message

The response message, QnAAVerificationResponse, returns the transaction identifier and the authentication token in the SOAP envelope header. The SOAP body includes the verification result for each attribute and the Fault response for an error condition.

See the following table for more information on the elements returned for a successful transaction. See appendix, "Error Codes" (see page 353) if there are any errors.

Element	Description
Header Elements	
udsTransactionID	The unique identifier of the transaction that is performed using UDS.
authToken	The authentication token that is returned if the credential verification to access Web services was successful. This token eliminates the need for you to present the authentication credential for successive access to the Web services. By default, the authentication token is valid for *one* day, after which you need to authenticate again.
Body Elements	
QnAResponseAttribute/name	The name of the user attribute that was verified.

Element	Description
QnAResponseAttribute/result	The result of the verification. Possible values are: ■ MATCHED ■ NOT_MATCHED ■ NOT_VERIFIED ■ NOT_FOUND

Chapter 7: Managing AuthMinder Configurations

Managing configurations is a key part of AuthMinder management. You can manage AuthMinder configurations at two levels:

- Global, applicable to all organizations
- Organization-level, applicable to an individual organization

When you set global configurations at the system level, all organizations in the system can inherit them. You can also override these global settings at the organization-level, and apply them only to the specific organization where they were set. The changes you make to the configuration globally or at an organization-level are not applied automatically. You need to refresh all server instances to apply these configuration changes.

This chapter discusses the Web services operations that AuthMinder provides to create and manage configuration profiles used to configure various credentials and authentication policies. In AuthMinder, a profile comprises a logical grouping of configuration settings for a particular credential. This chapter covers the following topics:

- Creating Configurations (see page 152)
- Updating Configurations (see page 183)
- Fetching Configurations (see page 184)
- Assigning Default Configurations (see page 187)
- Fetching Server Events (see page 191)
- Checking Key Availability in HSM (see page 193)
- Deleting Configurations (see page 195)

To perform the operations discussed in this chapter, you need to use the ArcotWebFortAdminSvc.wsdl file.

Creating Configurations

Each end user in AuthMinder is associated with at least one credential (such as ArcotID PKI, QnA, Password, or OTP) that they must use to log in to the system. With a large number of end users enrolled with AuthMinder, you might find that the same credential template can be applied as-is to many users. In such cases, AuthMinder provides you the flexibility to create common ready-to-use credential configurations, known as credential profiles that can be shared among multiple organizations and, thereby, applied to multiple users. As a result, credential profiles simplify the management of credential issuance.

Credential profiles specify issuance configuration properties, and credential attributes such as, validity period, key strengths, and details related to password strength. AuthMinder is shipped with a default profile for each credential.

Also, AuthMinder supports multiple authentication mechanisms. Every time an end user attempts authentication against AuthMinder, the authentication process is controlled by a set of rules referred to as *authentication policies*. These rules can be configured to track the number of failed authentication attempts allowed before credential lockout, and also to track user status before authentication.

You can create configurations either by using Administration Console or by using Administration Web services. This section walks you through the following topics for configuring credential profiles, authentication policies, domain key and master keys, RADIUS, plug-ins, ASSP, and SAML tokens:

- Preparing the Request Message (see page 152)
- Invoking the Web Service (see page 181)
- Interpreting the Response Message (see page 182)

Preparing the Request Message

The createRequest message is used to set the following information:

- Credential Profiles (see page 153)
- Authentication Policies (see page 162)
- Miscellaneous Configurations (see page 166)
- Domain Key and Master Keys (see page 166)
- RADIUS Configurations (see page 169)
- Credential Type Resolution Configurations (see page 174)
- Plug-In Configurations (see page 175)
- ASSP Configurations (see page 178)
- SAML Token Configurations (see page 179)

Credential Profiles

AuthMinder provides profiles for all the supported credentials. To modify the default credential profile or add new profiles, use the elements discussed in this section:

- Common Profile Elements
- ArcotID PKI Credential Profile Elements
- Questions and Answers (QnA) Credential Profile Elements
- Password Credential Profile Elements
- OTP Credential Profile Elements
- OATH OTP Credential Profile Elements
- ArcotID OTP Credential Profile Elements
- EMV OTP Credential Profile Elements

Common Profile Elements

The following table lists the common elements that are applicable to all credentials:

Element	Mandatory	Description
clientTxId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
Additional Input (additionalInput) Elements		
pairs	No	AuthMinder's additionalInput element enables you to set additional inputs if you want to augment AuthMinder's authentication capability by specifying additional information. In such cases, you need to set the extra information in *name-value* pairs. ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information.

Element	Mandatory	Description
Organization Detail (configurations/orgDetails) Elements		
orgName	Yes	Indicates the name of the organization to which you want to apply these configuration settings.
OR		
isGlobal	Yes	Indicates whether you want to apply these configuration settings at the global level, which means these configurations will be available to all the organizations in the system.
Credential Configuration Elements The following elements are applicable to all credential configurations, namely **arcotIDIssuanceConfigs, qnaIssuanceConfigs, passwordIssuanceConfigs, serverOTPIssuanceConfigs, oathIssuanceConfigs, arcotOTPIssuanceConfigs, emvIssuanceConfigs**.		
name	No	Indicates the name of the new profile. Each profile is identified by a unique profile name.
status	No	Indicates the configuration status. Possible values are: ■ ACTIVE ■ DISABLED ■ DELETED ■ DEFAULT ■ READONLY
multipleUsageCount	No	Indicates the number of times a credential can be used.
usageType	No	Multiple credentials of the same type can be issued for a user. The usage type identifies the purpose for which each credential is used. For example, a user can have a temporary password to perform a remote login to the network. The usage type for this password can be *temporary*.

Element	Mandatory	Description
validity/ validityBegin and validityEnd	No	When creating a credential, you can set a period for which the credential will be valid. The validityBegin and validityEnd elements enable you to set the validity period by using the following elements: ■ year The year when the validity period begins or ends. ■ month The month when the validity period begins or ends. ■ day The day on which the validity period begins or ends ■ hour The hour at which the validity period begins or ends. ■ minute The minute at which the validity period begins or ends. ■ second The second at which the validity period begins or ends. ■ dateType The start date or end date of the validity period. Following are the supported date types: **1** Uses the current date of AuthMinder Server to set the validity or disable period. This is not applicable for validityEnd. **2** Indicates that the credential will be valid forever and will not expire. This is not applicable for validityBegin. **3** Uses the absolute date that is specified by your application to set the validity or disable period. **4** Uses a relative date corresponding to the start date. For example, if the relative date is one month, then the end date would be one month after the start date.

Element	Mandatory	Description
userCheck	No	AuthMinder uses the user check information before performing some of the credential operations. The following elements are used to perform user checks: ■ userActiveCheck Indicates the user status. The issuance operation will fail if the user is in the disabled state. ■ userAttributesToCheck Indicates whether the user attributes match certain values. You can set the attributes in name-value pairs. name Indicates the attribute whose value you want to match before creating the credential. value Indicates the corresponding value for the name.
customAttributes	No	This element is used to define any custom attributes for a credential profile. This helps in maintaining any additional credential information. For example, if you do not want the user to download their ArcotID PKI on more than five systems, then you can create an attribute with this information. You can set the custom attributes in name-value pairs. ■ name Indicates the name with which you want to create the custom attribute. ■ value Indicates the corresponding value for the name.

ArcotID Credential Profile Elements

The following table lists the elements that are specific to the ArcotID PKI credential profile (arcotIDIssuanceConfigs):

Element	Mandatory	Description
keyLength	No	Indicates the size (in bits) of the key to be used in ArcotID PKI’s Cryptographic Camouflage algorithm.

Element	Mandatory	Description
unsignedAttributes	No	Indicates the attributes that are set while creating or after creating an ArcotID PKI for the user. Such attributes are called *unsigned attributes* because these attributes (name-value pairs) are set in the unsigned portion of the ArcotID PKI. ■ name Indicates the name with which you want to create the unsigned attribute. ■ value Indicates the corresponding value for the name.
passwordStrengthParameters	No	The effectiveness of the password, which is determined by a combination of the length of the password and number of alphabets, numerals, and special characters in it. The following elements are used to set the strength of a password: ■ minLength The least number of characters that the password can contain.The minimum length must be between 1 and 64 characters. ■ maxLength The maximum number of characters that the password can contain.The maximum length must be between 1 and 64 characters. ■ minAlphaChars The least number of alphabetic characters (a-z and A-Z) that the password can contain. ■ minNumericChars The least number of numeric characters (0 through 9) that the password can contain. ■ minSpecialChars The least number of special characters that the password can contain. By default, all the special characters excluding ASCII (0-31) characters are allowed. **Note:** The sum of all the elements must be less than minLength.

Element	Mandatory	Description
historyConfig	No	This element is used to enforce users to not reuse old ArcotID PKI passwords. Any one of the following elements can be used for configuration: ■ count Use this element if you want the current ArcotID PKI password to be different from the last *n* passwords. ■ time Use this element if you want the current ArcotID PKI password to be different from the passwords created during a specified duration. For information about the elements used to specify duration, see the "validity/ validityBegin and validityEnd" element.

Questions and Answers (QnA) Credential Profile Elements

The following table lists the elements that are specific to the QnA credential profile (qnaIssuanceConfigs):

Element	Mandatory	Description
maxQuestions	No	Indicates the maximum number of questions and answers the user must set during issuance.
minQuestions	No	Indicates the minimum number of questions and answers the user can set during issuance.
questions	No	A list of pre-configured questions that users can use to set up their QnA credential.
isCaseSensitive	No	Indicates whether the answers entered by the users must be case-sensitive or not.
questionReturn Mode	No	Indicates how the questions must be selected for the users to provide their answers. The supported values are: ■ **1** Indicates a static set wherein a fixed set of questions are selected from the configured set and presented to users. ■ **2** Indicates a random set wherein the questions are selected randomly from the configured set and presented to users.

Password Credential Profile Elements

The following table lists the elements that are specific to the Password credential profile (passwordIssuanceConfigs):

Element	**Mandatory**	**Description**
enforceUniquenessAcrossUsageTypes	No	Multiple passwords that are set using the usageType can be unique or the same.
generatePassword	No	Indicates whether the password should be generated by AuthMinder Server.
passwordStrengthParameters	No	The effectiveness of password, which is determined by a combination of the length of the password and number of alphabets, numerals, and special characters in it. The following elements are used to set the strength of a password: ■ minLength The least number of characters that the password can contain.The minimum length must be between 1 and 64 characters. ■ maxLength The maximum number of characters that the password can contain.The maximum length must be between 1 and 64 characters. ■ minAlphaChars The least number of alphabetic characters (a-z and A-Z) that the password can contain. This value must be lesser than or equal to the value specified in minLength. ■ minNumericChars The least number of numeric characters (0 through 9) that the password can contain. ■ minSpecialChars The least number of special characters that the password can contain. By default, all the special characters excluding ASCII (0-31) characters are allowed. **Note:** The sum of all the elements must be less than minLength.

Element	Mandatory	Description
historyConfig	No	This element is used to enforce users to not reuse old passwords. Select any one. ■ count Use this element if you want the current password to be different from the last passwords. ■ time Use this element if you want the current password to be different from the passwords created during a specified duration. For information about the elements used to specify duration, see the "validity/ validityBegin and validityEnd" element.

OTP Credential Profile Elements

The following table lists the elements that are specific to the OTP credential profile (serverOTPIssuanceConfigs):

Element	Mandatory	Description
length	No	The length of the OTP. By default, the OTP length is 5.
type	No	Indicates whether the OTP is numeric or alphanumeric. Following are the supported values: ■ 1: Generates a numeric OTP. ■ 2: Generates an alphanumeric OTP.

OATH OTP Credential Profile Elements

The OATH OTP credential (oathIssuanceConfigs) does not have any specific configurations.

ArcotID OTP Credential Profile Elements

The following table lists the elements that are specific to the ArcotID OTP (also known as ArcotID OTP-OATH) credential profile (arcotOTPIssuanceConfigs):

Element	Mandatory	Description
length	No	The length of the OTP.
type	No	The type of the OTP. Following are the supported values: ■ HOTP ■ TOTP

Element	Mandatory	Description
provisioningAttributes	No	User attributes that must be set at the time of issuing the credential.
customCardAttributes	No	Additional attributes that you need to pass for the ArcotID OTP-OATH credential in the name-value pair format. These attributes are added in the card.

EMV OTP Credential Profile Elements

The following table lists the elements that are specific to the EMV OTP (also known as ArcotID OTP-EMV) credential profile (emvIssuanceConfigs):

Element	Mandatory	Description
accountType	No	The Primary Account Number (PAN) of the EMV card.
attributeForPanSequence	No	The user account attribute that contains the PAN sequence, which identifies and differentiates cards with the same PAN.
provisioningAttributes	No	User attributes that must be set at the time of issuing the credential.
emvAttributes	No	EMV-specific attributes.
customCardAttributes	No	Additional attributes that can be added at the time of issuing the credential. These attributes are added in the card.

Authentication Policies

The createRequest message is used to create authentication policies in the AuthMinder database.

This section lists the elements that are required to set the credential policy information.

- Common Policy Elements
- ArcotID PKI Authentication Policy Elements
- QnA Authentication Policy Elements
- Password Authentication Policy Elements
- OTP-Based Authentication Policy Elements

Common Policy Elements

The following table lists the common policy-related elements that are applicable to all credentials:

Element	Mandatory	Description
name	No	Indicates the name of the new policy.
status	No	Indicates the status of the configuration. Possible values are as follows: ■ ACTIVE ■ DISABLED ■ DELETED ■ DEFAULT ■ READONLY
maxStrikes	No	Indicates the number of failed attempts after which the user's credentials will be locked out.
warningPeriod	No	Indicates the number of days before the warning is sent to the calling application about the user's impending credential expiration.
gracePeriod	No	Indicates the number of days a user is allowed to authenticate successfully with their expired ArcotID PKI credential.
autoUnlockPeriod	No	Indicates the number of hours after which a locked credential can automatically be used to log in again.

Element	Mandatory	Description
userCheck	No	AuthMinder uses the user check information before performing some of the operations. The following elements are used to perform user checks: ■ userActiveCheck Indicates whether the user is active. ■ userAttributesToCheck Indicates whether the user attributes match certain values. You can set the attributes in name-value pairs. name Indicates the name with which you want to create the attribute. value Indicates the corresponding value for the name.
matchAcrossUsageType	No	Indicates a match across usage types. Multiple credentials of the same type can be issued for a user. A description is necessary to identify the purpose for which each credential is used. For example, a user can have a temporary password to perform a remote login to the network. The usage type for this password can be *temporary*.
usageTypeToMatch	No	Indicates the usage type that needs to be matched.

ArcotID PKI Authentication Policy Elements

The following table lists the elements that are specific to the ArcotID PKI credential authentication policy (arcotIDAuthConfigs):

Element	Mandatory	Description
challengeTimeout	No	Indicates the duration for which the ArcotID PKI challenge must be valid. By default, the validity period is 300 seconds.

QnA Authentication Policy Elements

The following table lists the elements that are specific to the QnA credential authentication policy (qnaAuthConfigs):

Element	**Mandatory**	**Description**
numQuestionsToChallenge	No	Indicates the number of questions that AuthMinder must ask users during authentication. The default value is 3.
minAnswersRequired	No	Indicates the minimum number of questions for which correct answers are required during authentication. The default value is 3.
questionsChallengeMode	No	Indicates how the questions are selected for the challenge. The supported values are: ■ **1** This indicates a random set wherein the questions are selected randomly from the configured set. ■ **2** This indicates an alternate set wherein a new set of questions is selected from the configured set, which means the questions that were asked in the last authentication prompt are skipped.
questionSetChangeOption	No	Specifies when AuthMinder Server must select a new set of questions for the challenge. ■ **1** This indicates that a fixed set of questions are selected from the configured set and presented to the users. ■ **2** This indicates that a random set of questions are selected from the configured set and presented to the users.
isCVMEnabled	No	Indicates whether caller side verification is enabled or not. The supported values are: ■ 0: Indicates the feature is disabled. ■ 1: Indicates the feature is enabled. See "Questions and Answers Authentication" (see page 264) for more information on caller side verification.
challengeTimeout	No	Indicates the duration for which the QnA challenge must be valid. By default, the validity period is 300 seconds.

Password Authentication Policy Elements

The following table lists the elements that are specific to the Password credential authentication policy (passwordAuthConfigs):

Element	Mandatory	Description
numPositionsToChallenge	No	Indicates the total number of password character positions that have to be challenged by AuthMinder Server. **Note:** Applicable *only* for partial passwords.
challengeTimeout	No	Indicates the duration for which the password challenge has to be valid. By default, the validity period is 300 seconds.

OTP-Based Authentication Policy Elements

The following table lists the elements that are specific to the OATH OTP, ArcotID OTP, and EMV OTP credential authentication policy (oathAuthConfigs,arcotOTPAuthConfigs, and emvAuthConfigs).

Note: The OTP generated by AuthMinder Server (serverOTPAuthConfigs) does not have any specific configurations.

Element	Mandatory	Description
otpCounterTolerance	No	This element contains the OTP counter tolerance parameters. ■ authLookAhead Indicates the number of times the OTP counter on AuthMinder Server is increased to verify the OTP entered by the user. ■ authLookBack Indicates the number of times the OTP counter on AuthMinder Server is decreased to verify the OTP entered by the user. ■ reSyncLookAhead Indicates the number of times the OTP counter on AuthMinder Server is increased to synchronize with the OTP counter on the client device. ■ reSyncLookBack Indicates the number of times the OTP counter on AuthMinder Server is decreased to synchronize with the OTP counter on the client device.

Miscellaneous Configurations

The following table lists the authentication-related miscellaneous configurations:

Element	Mandatory	Description
miscellaneousConfigs	No	■ nativeTokenTimeout Native tokens are AuthMinder-proprietary tokens that can be used multiple times before they expire. This element indicates the validity time, in seconds, of the token. ■ ottLength A one-time token (OTT) can be used only once before it expires. This element indicates the length of the OTT. ■ ottTimeout Indicates the validity time, in seconds, of the OTT.

Domain Key and Master Keys

Keys are used to protect the shared secret that is used to generate and authenticate credentials, which include ArcotID PKI, OATH OTP, ArcotID OTP-OATH, and ArcotID OTP-EMV. The key used to create and manage the ArcotID PKI is called Domain Key and the keys used to create and manage other credentials are called Master Keys.

When the user tries to authenticate using their credential, AuthMinder first checks whether the key that was used to protect the credential is valid. If the key is valid, then the user will be authenticated on providing the correct credential. Else, the user authentication fails.

By default, a key configuration is created when the AuthMinder Server is started for the first time. You can use this default configuration or create your own configuration using the keyConfigs element. You can create multiple key configurations, but only the configuration that is assigned to the credential type is used for creating credentials and authenticating those configurations. The other active configurations are used for authentication only.

The keyConfigs element is used to create the key configurations. The following table lists the key management-specific elements of this message:

Element	Mandatory	Description
name	No	Name for the configuration.
status	No	Indicates the status of the configuration.
label	No	The label that will be used to store the Domain Key.

Element	Mandatory	Description
keyStatus	No	Indicates the status of the key. Following are the supported values: ■ **1**: The key is active. The configurations created using this key can be used for both authentication and issuance operations. ■ **2**: The key is inactive. The configurations created using this key might have expired. In this case, you can extend the validity and continue to use the credentials. ■ **3**: The key is retired. The configurations created using this key are not valid anymore, and the credentials associated with this configuration will expire.
keyInHSM	No	Indicates whether you want to store the key in the Hardware Security Module (HSM).

Element	Mandatory	Description
validity/ validityBegin and validityEnd	No	While creating a key, you can set a period for which the key will be valid. When the key expires, the credentials issued with that key also expires. The validityBegin and validityEnd elements enable you to set the validity period by using the following elements: ■ year The year when the validity period begins or ends. ■ month The month when the validity period begins or ends. ■ day The day on which the validity period begins or ends ■ hour The hour at which the validity period begins or ends. ■ minute The minute at which the validity period begins or ends. ■ second The second at which the validity period begins or ends. ■ dateType The start date or end date of the validity period. The following are the supported date types: **1** Uses the current date of AuthMinder Server to set the validity or disable period. This is not applicable for validityEnd. **2** Indicates that the credential will be valid forever and will not expire. This is not applicable for validityBegin. **3** Uses the absolute date that is specified by your application to set the validity or disable period. **4** Uses a relative date corresponding to the start date. For example, if the relative date is one month, then the end date would be one month after the start date.

RADIUS Configurations

If configured, AuthMinder can serve as a RADIUS Server to the configured Network Access Server (NAS) or the RADIUS clients.

This section walks you through preparing request messages for the following:

- RADIUS Client
- RADIUS Server

RADIUS Client

The radiusClientConfigs element of the createRequest message is used to configure RADIUS Client. The following table lists the elements of this message:

Element	**Mandatory**	**Description**
name	No	Name for the configuration.
status	No	Indicates the status of the configuration.

Element	Mandatory	Description
radiusClient	No	Contains the following elements: ■ authType The authentication mechanism that will be used for VPN authentication. The supported authentication mechanisms are: - RADIUS OTP - In-Band Password To use this method, configure the credential type resolution. - EAP ■ description A string to describe the RADIUS client. The description helps to identify the RADIUS client, if multiple clients are configured. ■ maxPacketSize The packet size for the RADIUS messages. ■ protocolVersion The RADIUS version supported for the client being added. The supported values are: - 1.0 - 2.0 ■ sharedSecret The secret shared between the RADIUS client and AuthMinder Server. ■ additionalRADIUSAttributes Contains attributes that you want AuthMinder Server to return in the response message sent to the RADIUS client after successful authentication. The attributes are set in name-value pairs. ■ defaultOrg Name of the default organization that is supported by the RADIUS client. This attribute is used in In-Band authentication and is used to resolve the organization name during authentication..

Element	Mandatory	Description
		■ orgsSupported List of organizations that are supported by the RADIUS client, these organization are configured at the global-level. This attribute is used in In-Band authentication and is used to resolve the organization name during authentication. ■ packetDropConditions The conditions for which the AuthMinder server will not process the RADIUS requests. Following are the possible values: 1102: For user not found condition 5800: For credential not found condition 1000: For internal error 1051: For invalid requests
radiusClient	No	■ enableRetry Indicates whether the RADIUS client should try to send the request to AuthMinder Server if it does not receive any response. ■ retryWindow Indicates the duration in seconds for which the client must wait to receive a response, in case the enableRetry element is set to true. After this period, the retry is considered invalid.
eapAuthTypeData	No	Contains the following elements related EAP authentication. Set any of the following elements: ■ serverCertKeyPair/KeyPairInHSM Set the serverCertKeyPair element to AuthMinder Server certificate chain in PEM format. ■ serverCertKeyPair/KeyPairInP12 Set cerKeyP12 to the base64-encoded format of AuthMinder Server certificate in PKCS#12 format. Set certKeyP12Password to the password of the PKCS#12 file.

RADIUS Server

AuthMinder can be used as a proxy server to pass any password-based authentication requests to other servers that work on RADIUS protocol.

The radiusServerConfigs element of the createRequest message is used to configure RADIUS Server.

The following table lists the elements of this message:

Element	Mandatory	Description
name	No	Name for the configuration.
status	No	Indicates the status of the configuration.
isEnabled	No	An option to enable AuthMinder Server to pass the RADIUS requests to the other configured RADIUS server.
useSystemConfig	No	An option to use system configuration or organization level configuration.

Element	Mandatory	Description
radiusServers	No	Contains the following elements: ■ authType The authentication mechanism that will be used for VPN authentication. The supported authentication mechanisms are: - RADIUS OTP - In-Band Password ■ description A string to describe the RADIUS server. The description helps to identify the RADIUS server, if multiple servers are configured. ■ maxPacketSize The packet size for the RADIUS messages. ■ protocolVersion The RADIUS version supported for the server being added. The supported values are: - 1.0 - 2.0 ■ sharedSecret The secret symmetric key shared between the RADIUS server and AuthMinder Server. ■ additionalRADIUSAttributes Contains attributes that you want AuthMinder Server to forward to the RADIUS server. The attributes are set in name-value pairs. ■ ipAddress The IP Address of the RADIUS server. ■ port The port number on which the RADIUS server is listening.

Element	Mandatory	Description
		■ readTimeout Indicates the maximum time to wait for a response from the RADIUS server. ■ retryCount Indicates the number of times AuthMinder Server should try to connect to RADIUS server if there is no response from the AuthMinder Server. ■ failoverOrder If multiple servers are configured, then this element identifies the server priority, based on this the requests are sent to a particular server in case of failover.

Credential Type Resolution Configurations

The authentication requests that are presented to AuthMinder Server must specify the type of credential that has to be used to process the request. In case of RADIUS and ASSP authentication requests, the input requests do not specify the type of credential. By default, RADIUS uses One-Time Password and ASSP uses password credential for authentication.

To support any password-based authentication mechanisms for RADIUS and ASSP, or to use verifyPlain authentication, you must create the *Credential Type Resolution* configuration. You can map the input request to any of the following password type of credentials that AuthMinder supports:

- Password
- One-Time Password
- One-Time Token (OTT)
- OATH OTP
- ArcotID OTP OATH
- ArcotID OTP EMV
- LDAP Password
- Native Token

The credTypeResolutionConfigs element of the createRequest message is used to configure credential types.

The following table lists the elements of this message:

Element	Mandatory	Description
name	No	Name for the configuration.

Element	Mandatory	Description
status	No	Indicates the status of the configuration.
credType	No	The type of credential that has to be used to authenticate users with this credential type resolution configuration. Following are the supported values: ■ **1**: For password ■ **4**: For OTP ■ **5**: For OTT ■ **7**: For OATH OTP ■ **8**: For ArcotID OTP-OATH ■ **9**: For ArcotID OTP-EMV ■ **10**: For LDAP password ■ **11**: For Native token
userAttributeForCredType	No	The custom attributes of the user that defines the credential type to be used to authenticate the user. Include credType as one of the attributes. **Note:** The user attributes that you provide here must match the attributes that you have specified for the user during user creation.

Plug-In Configurations

Plug-ins enable you to extend the functionality of AuthMinder Server. Using the pluginConfigs element of the createRequest message, you can register and configure a plug-in.

Important! To use plug-in configurations, first log in to Adminstration Console as the MA and then register a plug-in.

The following table lists the pluginConfigs element of the createRequest message:

Element	Mandatory	Description
name	No	Name for the configuration.
status	No	Indicates the status of the configuration.

Element	Mandatory	Description
pluginConfigElements	No	Contains the following elements: ■ name Indicates the name of the plug-in that you want to configure. ■ type Indicates the type of the plug-in element. For example, integer, list, boolean, string, or file. ■ description User friendly description for the plug-in element. ■ defaultValue The default value of the plug-in element. ■ assignedValue The value assigned for the plug-in element in string format. ■ listValues If the plug-in element is of the type list, then this input specifies the values that are supported by the list. For example, if the plug-in element type is OTP, then the listValues would be Numeric, Alphabetic, or Alpha-Numeric. ■ assignedValueBytes The value assigned for the plug-in element in base64-encoded format.

Element	Mandatory	Description
events	No	An *event* is a pre-defined operation in the AuthMinder system. You must define the events that will invoke a configured plug-in. Use the following elements to configure the plug-in events: ■ eventsSupported The events supported by the plug-in. - **eventID** A numeric representation of the event. When associating an event with a plug-in, the eventID is required. - **eventName** A string representation of the event. When associating an event with a plug-in. ■ eventsAvailable For a given organization, the unused events that can be associated with a plug-in. - **eventID** A numeric representation of the event. When associating an event with a plug-in, the event ID is required. - **eventName** A string representation of the event. When associating an event with a plug-in, the eventName is not required. ■ eventsAssociated Events already associated with a plugin or those that are available for use. - **eventID** A numeric representation of the event. When associating an event with a plug-in, the eventID is required. - **eventName** A string representation of the event. When associating an event with a plug-in, the eventName is not required.
isTemplate	No	Indicates whether the plug-in information is to be read from a template.

ASSP Configurations

Adobe Signature Service Protocol (ASSP) is used for signing PDF documents using CA SignFort. Before signing, users are authenticated by using AuthMinder authentication methods. A SAML token is returned to the user after successful authentication. This token is then verified by the SignFort Server. Using the asspConfigs element of the createRequest message, you can configure ASSP.

The following table lists the asspConfigs element of the createRequest message:

Element	**Mandatory**	**Description**
name	No	Name for the configuration.
status	No	Indicates the status of the configuration.
tokenServer	No	The name of the server that issues the authentication token.
roamingURL	No	The ArcotID PKI Roaming URL that will be used to download ArcotID PKIs in case of ArcotID PKI Roaming Download. In case of ArcotID PKI Roaming Download, if the user does not have their ArcotID PKI present on their current system, then the ArcotID PKI Roaming URL is used to authenticate to AuthMinder Server and download the user's ArcotID PKI.
mechanisms	No	The authentication mechanism that will be used for authentication. The information required is: ■ **mechanism** The authentication mechanism that will be used to authenticate the user before signing. ■ **status** **1** indicates that ASSP authentication mechanism will be used.
samlTokenSigningCertKeyPair	No	Indicates the Store path that contains AuthMinder Server certificate, and the private key that will be used by AuthMinder Server to issue the SAML token. Following are the choices: ■ KeyPairInHSM Set the certChainPEM element to AuthMinder Server certificate chain in PEM format. ■ KeyPairInP12 Set cerKeyP12 to the base64-encoded format of AuthMinder Server certificate in PKCS#12 format. Set certKeyP12Password to the password of the PKCS#12 file.

Element	Mandatory	Description
samlTokenAttributes	No	The attributes of the SAML token. The attributes required are: ■ issuerName The name of the Issuer who will provide the SAML token generated by AuthMinder. ■ oneTimeUse Indicates whether the SAML token is to be used only once for authentication. ■ assertionTimeOut The duration after which the SAML token cannot be used. ■ audiences The details of the audience who can use the SAML token.
kerberosUseProcessCredentials	No	Indicates whether Kerberos is to be used as the authentication method.
kerberosUserName	No	User name for Kerberos authentication.
kerberosPassword	No	Password for Kerberos authentication.
kerberosDomainName	No	Domain name for Kerberos authentication.

SAML Token Configurations

AuthMinder supports different types of authentication tokens, and Secure Assertion Markup Language (*SAML*) tokens are one among them (in addition to Native, OTT, and Custom token types.)

If you want to issue SAML as authentication tokens, then you must configure the SAML token properties. Using the samlTokenConfigs element of the createRequest message, you can configure SAML.

The following table lists the samlTokenConfigs element of the createRequest message:

Element	Mandatory	Description
name	No	Name for the configuration.
status	No	Indicates the status of the configuration.

Element	Mandatory	Description
tokenSigningCertKeyPair	No	Indicates the path that contains AuthMinder Server certificate, and the private key that will be used by AuthMinder Server to issue the SAML token. Following are the choices: ■ KeyPairInHSM Set the certChainPEM element to AuthMinder Server certificate chain in PEM format. ■ KeyPairInP12 Set cerKeyP12 to the base64-encoded format of AuthMinder Server certificate in PKCS#12 format. Set certKeyP12Password to the password of the PKCS#12 file.
digestMethod	No	The algorithm that will be used for hashing the SAML tokens.
signatureMethod	No	The algorithm that will be used for signing the SAML tokens.
samlTokenAttributes	No	The attributes of the SAML token. The attributes required are: ■ issuerName The URL of AuthMinder Server. ■ oneTimeUse Indicates whether the SAML token is to be used only once for authentication. ■ assertionTimeOut The duration after which the SAML token cannot be used. ■ audiences The details of the audience who can use the SAML token.
subjectFormatSAML11	No	The format of the SAML subject for SAML 1.1.
subjectFormatSAML20	No	The format of the SAML subject for SAML 2.0.

Element	Mandatory	Description
additionalAttributes	No	You can set additional attributes, if required for the SAML token generation. Following are the required elements: ■ attributeNameSpace The attribute namespace. ■ nameFormat The attribute name format. ■ attributeName The name of the attribute. ■ FriendlyName The friendly name for the attribute.

Invoking the Web Service

To create configurations:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the create operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. Use the createRequest elements to set the configuration information, as listed in the tables in Credential Profiles (see page 153) and Authentication Policies (see page 162).
3. Use the createRequest message and construct the input message by using the details obtained in the preceding step.
4. Invoke the create operation of the ArcotWebFortAdminSvc service to create the configuration.
5. This operation returns createResponse message that includes the transaction identifier, message, reason code, and response code. Refer to the next section for more information on the response message.

Interpreting the Response Message

For successful transactions, the response message, createResponse returns the elements explained in the following table. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	Description
message	Indicates the status of the transaction.
reasonCode	Unique code that is sent by AuthMinder Server if the operation fails.
responseCode	Code returned by the SDK in case of errors.
transactionID	The unique identifier of the transaction.
additionalOutput	The values returned for the additional input.

Updating Configurations

The update operation enables you to update the following:

- Credential Profiles
- Authentication Policies
- Domain Keys and Master Keys
- RADIUS
- Plug-ins
- ASSP
- SAML Tokens

This section walks you through the following topics for updating configurations:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: After you update configurations, you must refresh the system cache for the changes to take effect. See "Refreshing the Organization Cache" (see page 50) for more information on how to refresh the system cache.

Preparing the Request Message

The UpdateRequest message is used to update configurations in the AuthMinder database. All the createRequest elements listed in the tables in Credential Profiles (see page 153) and Authentication Policies (see page 162) can be updated using the update operation. For information about each element, refer to the tables in Credential Profiles (see page 153) and Authentication Policies (see page 162).

Invoking the Web Service

To update configurations:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the update operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. Use the UpateRequest elements to update the configuration information.
3. Use UpateRequest message and construct the input message by using the details obtained in the preceding step.
4. Invoke the update operation of the ArcotWebFortAdminSvc service to update the configuration.

This operation returns updateResponse message that includes the transaction identifier, message, reason code, and response code. Refer to the next section for more information on the response message.

Interpreting the Response Message

For successful transactions, the response message, updateResponse returns the elements explained in the following table. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	Description
message	Indicates the status of the transaction.
reasonCode	Unique code that is sent by AuthMinder Server if the operation fails.
responseCode	Response code returned by the AuthMinder Server in case of errors.
transactionID	The unique identifier of the transaction.
additionalOutput	The values returned for the additional input.

Fetching Configurations

The fetch operation is used to fetch configurations.

This section walks you through the following topics for fetching configuration details:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The fetchRequest message is used to fetch configuration details. The following table lists the common elements of this request message that are applicable to all credentials:

Element	Mandatory	Description
clientTxnId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
Additional Input (additionalInput) Elements		

Element	Mandatory	Description
pairs	No	AuthMinder's additionalInput element enables you to set additional inputs if you want to augment AuthMinder's authentication capability by specifying additional information. In such cases, you need to set the extra information in *name-value* pairs. ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information.
Organization Detail (configurationNames/orgDetails) Elements		
orgName	Yes	Fetches the configurations that are used by the specified organizations.
OR		
isGlobal	Yes	Indicates if you want to fetch the configuration applied at the global-level.
Configuration Elements The following two elements are applicable to all configurations (**configurationNames/configurationname<Names>**).		
configNames	No	Fetches the configuration names for a particular organization.
isAllConfigs	No	Indicates whether all configurations must be fetched.
isFetchOnlyNames	No	Indicates whether only the configuration name has to be fetched.

Invoking the Web Service

To fetch configuration details:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the fetch operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. Use the FetchRequest elements to fetch the configuration details, as listed in the preceding table.
3. Use the FetchRequest message and construct the input message by using the details obtained in the preceding step.
4. Invoke the fetch operation of the ArcotWebFortAdminSvc service to fetch the configuration details.

Interpreting the Response Message

For successful transactions, the response message, fetchResponse returns the elements explained in the following table. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	**Description**
message	Indicates the status of the transaction.
reasonCode	Unique code that is sent by AuthMinder Server if the operation fails.
responseCode	Code returned by the SDK in case of errors.
transactionID	The unique identifier of the transaction.
additionalOutput	The return values of the additional input.
configurations	Returns the following: ■ **configurationObjects**: The configured objects. ■ **configurationNames**: The names of the configurations.

Assigning Default Configurations

After you have created the required credential profiles and authentication policies, you need to assign them globally as a Global Administrator (GA) or to a specific organization. If the Organization Administrator (OA) does not specify profiles and policies at their organization level, then the global profiles and policies are used by default. On the other hand, if a GA or an OA overwrites these configurations at their individual organization level, then those configurations are applicable for the organization.

The assignDefault operation is used to assign default configurations. This section walks you through the following topics for assigning default configurations:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The assignDefaultRequest message is used to assign default configurations. The following table lists the elements of this request message:

Element	**Mandatory**	**Description**
clientTxId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
Additional Input (additionalInput) Elements		

Element	Mandatory	Description
pairs	No	AuthMinder's additionalInput element enables you to set additional inputs if you want to augment AuthMinder's authentication capability by specifying additional information. In such cases, you need to set the extra information in *name-value* pairs. ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information.
Assign Default Elements (assignDefault/OrgDetails)		
orgName	Yes	Indicates the name of the organization to which you want to apply these configuration settings.
OR		
isGlobal	Yes	Indicates whether you want to apply these configuration settings at the global level, which means these configurations will be available to all the organizations in the system.
Assign Default Elements (assignDefault/<CredentialType>ConfigName)		
arcotIDIssuanceConfigName	No	Name of the ArcotID PKI configuration credential profile.
qnaIssuanceConfigName	No	Name of the QnA configuration credential profile.
passwordIssuanceConfigName	No	Name of the Password configuration credential profile.
serverOTPIssuanceConfigName	No	Name of the OTP configuration credential profile.
oathIssuanceConfigName	No	Name of the OATH OTP configuration credential profile.

Element	Mandatory	Description
arcotOTPIssuanceConfigName	No	Name of the ArcotID OTP configuration credential profile.
emvIssuanceConfigName	No	Name of the EMV OTP configuration credential profile.
arcotIDAuthConfigName	No	Name of the ArcotID PKI authentication policy profile.
qnaAuthConfigName	No	Name of the QnA authentication policy profile.
passwordAuthConfigName	No	Name of the Password authentication policy profile.
serverOTPAuthConfigName	No	Name of the OTP authentication policy profile.
oathAuthConfigName	No	Name of the OATH OTP authentication policy profile.
arcotOTPAuthConfigName	No	Name of the ArcotID OTP authentication policy profile.
emvAuthConfigName	No	Name of the EMV OTP authentication policy profile.
radiusServerConfigName	No	Name of the RADIUS configuration.
arcotIDDomainKeyConfigName	No	Name of the ArcotID PKI domain key configuration profile.
oathOTPMasterKeyConfig	No	Name of the OATH OTP master key configuration profile.
arcotOTPMasterKeyConfig	No	Name of the ArcotID OTP master key configuration profile.
emvOTPMasterKeyConfig	No	Name of the EMV OTP master key configuration profile.
asspConfigName	No	Name of the ASSP configuration profile.
samlTokenConfigName	No	Name of the SAML Token configuration profile.
credTypeResolutionConfigName	No	Name of the configuration to be used to map the incoming authentication request to a particular credential type.

Element	Mandatory	Description
radiusCredTypeResolutionConfigName	No	In the case of RADIUS clients, by default, when AuthMinder is configured to authenticate RADIUS clients of SSL VPN type, it typically uses RADIUS OTP, which is a One-Time Password (OTP) for authenticating these clients. In addition to this method, AuthMinder now supports other password-based authentication mechanism for authenticating RADIUS clients.
asspCredTypeResolutionConfigName	No	By default, Adobe Signing Service Protocol (ASSP) uses password authentication mechanism to authenticate users before they sign the PDF documents. In addition to this method, AuthMinder supports other password-based authentication mechanism for this purpose.
miscConfigName	No	Name of the miscellaneous configurations.

Invoking the Web Service

To assign default configurations:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the assignDefault operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.

2. Use the assignDefault elements to assign default configurations, as listed in the preceding table.

3. Use the assignDefault message and construct the input message by using the details obtained in the preceding step.

4. Invoke the assignDefault operation of the ArcotWebFortAdminSvc service to delete the configuration details.

Interpreting the Response Message

For successful transactions, the response message, assignDefaultResponse returns the elements explained in the following table. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	Description
message	Indicates the status of the transaction.

Element	Description
reasonCode	Unique code that is sent by AuthMinder Server if the operation fails.
responseCode	Code returned by the SDK in case of errors.
transactionID	The unique identifier of the transaction.
additionalOutput	The additional output corresponding to the additional input.

Fetching Server Events

The FetchEventDetailsRequest message is used to fetch events that are used to invoke a configured plug-in.

This section walks you through the following topics for fetching configuration details:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The FetchEventDetailsRequest message is used to fetch events. The following table lists the common elements of this request message that are applicable to all credentials:

Element	Mandatory	Description
clientTxId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
Additional Input (additionalInput) Elements		
pairs	No	Fetches additional inputs defined to augment AuthMinder's user issuance capability using plug-ins, write completely new custom user issuance methods, or track transactions end-to-end. In all of these cases, the extra information is set in name-value pairs. ■ name Indicates the name used to create the key pair. ■ value Indicates the corresponding value for the name.
Organization Detail (orgDetails) Elements		

Element	Mandatory	Description
orgName	Yes	Deletes the name of the organization to which you applied the configuration settings.
OR		
isGlobal	Yes	Indicates if you want to apply these configuration settings at the global level.

Invoking the Web Service

To fetch event details:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the fetchServerEvents operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. Use the FetchServerEventsRequest elements to fetch event details, as listed in the preceding table.
3. Use the FetchServerEventsRequest message and construct the input message by using the details obtained in the preceding step.
4. Invoke the fetch operation of the ArcotWebFortAdminSvc service to fetch the event details.

Interpreting the Response Message

For successful transactions, the response message, fetchResponse returns the elements explained in the following table. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	Description
message	Indicates the status of the transaction.
reasonCode	Unique code that is sent by AuthMinder Server if the operation fails.
responseCode	Code returned by the SDK in case of errors.
transactionID	The unique identifier of the transaction.

Checking Key Availability in HSM

The isKeyAvailableInHSM operation is used to check if the key is present in the HSM. This section walks you through the following topics for checking the HSM keys:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: After you delete configurations, you must refresh the system cache for the changes to take effect. See "Refreshing the Organization Cache" (see page 50) for more information on how to refresh the system cache.

Preparing the Request Message

The isKeyAvailableInHSMRequest message is used to check the key in HSM. The following table lists the elements of this request message.

Element	**Mandatory**	**Description**
clientTxId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
Additional Input (additionalInput) Elements		
pairs	No	AuthMinder's additionalInput element enables you to set additional inputs if you want to augment AuthMinder's authentication capability by specifying additional information. In such cases, you need to set the extra information in *name-value* pairs. ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information.
Key Details		

Element	Mandatory	Description
keyLabel	Yes	The label of the key that refers to the key that is available in the HSM.

Invoking the Web Service

To delete configuration details:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the isKeyAvailableInHSM operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. Use the isKeyAvailableInHSMRequest elements to check the key details, as listed in the preceding table.
3. Use the isKeyAvailableInHSMRequest message and construct the input message by using the details obtained in the preceding step.
4. Invoke the isKeyAvailableInHSM operation of the ArcotWebFortAdminSvc service to delete the configuration details.

Interpreting the Response Message

For successful transactions, the response message, isKeyAvailableInHSMResponse returns the elements explained in the preceding table. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	Description
message	Indicates the status of the transaction.
reasonCode	Unique code that is sent by AuthMinder Server if the operation fails.
responseCode	Code returned by the SDK in case of errors.
transactionID	The unique identifier of the transaction.
additionalOutput	The return values corresponding to the additional input.

Deleting Configurations

The delete operation is used to delete configurations in AuthMinder. After you delete a configuration, the information related to that configuration is still maintained in the system. Therefore, you cannot create a configuration profile with the same name as that of a deleted one. This section walks you through the following topics for deleting configurations:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: After you delete configurations, you must refresh the system cache for the changes to take effect. See "Refreshing the Organization Cache" (see page 50) for more information on how to refresh the system cache.

Preparing the Request Message

The deleteRequest message is used to delete configurations. The following table lists the common elements of this request message that are applicable to all credentials and authentication policies.

Element	Mandatory	Description
clientTxId	No	Unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
Additional Input (additionalInput) Elements		

Element	Mandatory	Description
pairs	No	AuthMinder's additionalInput element enables you to set additional inputs if you want to augment AuthMinder's authentication capability by specifying additional information. In such cases, you need to set the extra information in *name-value* pairs. ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information.
Organization Detail (configurationNames/orgDetails) Elements		
orgName	Yes	Deletes the name of the organization to which you applied the configuration settings.
OR		
isGlobal	Yes	Indicates if you want to apply these configuration settings at the global level.
Credential and Authentication Configuration Elements The following elements are applicable to all credential and authentication policy configurations.		
configNames	No	Deletes the configuration names for a particular organization.
isAllConfigs	No	Indicates whether all configurations must be deleted.

Invoking the Web Service

To delete configuration details:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the delete operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. Use the DeleteRequest elements to delete the configuration details, as listed in the preceding table.
3. Use the DeleteRequest message and construct the input message by using the details obtained in the preceding step.
4. Invoke the delete operation of the ArcotWebFortAdminSvc service to delete the configuration details.

Interpreting the Response Message

For successful transactions, the response message, deleteResponse returns the elements explained in the following table. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	**Description**
message	Indicates the status of the transaction.
reasonCode	Unique code that is sent by AuthMinder Server if the operation fails.
responseCode	Code returned by the SDK in case of errors.
transactionID	The unique identifier of the transaction.
additionalOutput	The return values corresponding to the additional input.

Chapter 8: Performing Credential Operations

This chapter describes the credential lifecycle operations that are supported by the ArcotWebFortIssuanceSvc service. The operations listed in this chapter can be performed on all credentials that are supported by AuthMinder, and can be performed by using any of the following method:

- By using AuthMinder SDKs

  This mode enables you to automate the credential management operations programmatically.

- By using Administration Console

  Administration Console is a Web-based application and is typically suitable for *Customer Support Representatives* (CSRs), who handle the user requests (such as, disabling the credential, enabling the credential, or resetting the credential validity.)

Note: Refer to *CA AuthMinder Administration Guide* for more information on using Administration Console.

This section covers the following credential lifecycle operations:

- Before You Proceed (see page 201)
- Creating Credentials (see page 206)
- Disabling Credentials (see page 215)
- Enabling Credentials (see page 217)
- Resetting Credentials (see page 219)
- Fetching Credential Details (see page 221)
- Reissuing Credentials (see page 223)
- Resetting Credential Validity (see page 225)
- Resetting Custom Attributes (see page 227)
- Fetching QnA Configuration (see page 228)
- Setting Unsigned Attributes (see page 231)
- Deleting Unsigned Attributes (see page 234)
- Adding Elements to ArcotID PKI Key Bag (see page 236)
- Fetching ArcotID PKI Key Bag Elements (see page 239)
- Deleting ArcotID PKI Key Bag Elements (see page 242)
- Downloading Credentials (see page 245)
- Deleting Credentials (see page 247)

To perform the operations discussed in this chapter, you need to use the ArcotWebFortIssuanceSvc.wsdl file.

Note: Each operation discussed in this chapter can be performed simultaneously by using different credentials. If the operation fails for a single credential, then the operations for other credentials are also considered invalid. For example, if you are creating ArcotID PKI, QnA, and OTP, and the ArcotID PKI and OTP creation was successful, while the QnA creation failed, then all the three credentials have to be created again.

Before You Proceed

The Issuance Web service performs the user status checks (if enabled) before performing the credential operations that are discussed in this chapter. This section lists these user status checks, supported credential states, supported transitions between the credential states, and the credential operations that are possible on a particular credential state. It covers the following topics:

- Checking the User Status (see page 201)
- Credential States and Supported Transitions (see page 202)
- Credential Operations and States (see page 204)

Checking the User Status

AuthMinder uses the user status information *before* performing some of the credential operations. A user’s status in the database can be either INITIAL, ACTIVE, INACTIVE, or DELETED.

Note: For Issuance Web service to perform these checks, you must enable this option when you create configurations using the AuthMinder Administration Web Service. Refer to chapter, "Managing AuthMinder Configurations" (see page 151) for more information.

The following table lists all the credential operations and the user checks that are performed depending on the type of operation:

Operation	**Checks**		
	User Existence	**User Status**	**User Attribute**
Create	Yes	Yes	Yes
Delete	No	No	No
Disable	No	No	No
Enable	Yes	Yes	No
Fetch	No	No	No
Fetch QnA Configuration	No	No	No
Reissue	Yes	Yes	No
Reset	Yes	Yes	No
Reset Custom Attributes	Yes	Yes	No
Reset Validity	Yes	Yes	No
Download Credential	Yes	Yes	No

Operation	Checks		
	User Existence	User Status	User Attribute
Delete Unsigned Attributes	No	No	No
Set Unsigned Attributes	No	No	No
Add ArcotID Key Bag Elements	No	No	No
Fetch ArcotID Key Bag Elements	No	No	No
Delete ArcotID Key Bag Elements	No	No	No

Credential States and Supported Transitions

AuthMinder supports the following states for a credential that is issued to a user:

- **ACTIVE**

  Indicates that the credential is active and can be used for authentication.

- **DISABLED**

  The credential is disabled by the administrator.

- **LOCKED**

  The credential is locked when the user consecutively fails to authenticate for the maximum number of negative attempts configured. For example if the maximum attempts configured is 3, then the third attempt with wrong credential will lock the credential.

- **VERIFIED**

  The credential is verified when the OTP submitted by the user is authenticated by AuthMinder Server successfully.

  Note: This status is applicable *only* for OTP.

- **DELETED**

  The credential of the user is deleted.

When you perform an operation on a credential, the status of the credential might be changed after the operation is performed successfully on the credential. For example, when the user successfully authenticates with their OTP, then status of the user's OTP is changed to VERIFIED.

The following table lists the transitions possible between the supported credential states:

Current State	**Change State to**				
	Enabled	**Locked**	**Disabled**	**Deleted**	**Verified (for OTP only)**
Enabled	Yes	Yes	Yes	Yes	Yes
Locked	Yes	Yes	Yes	Yes	No
Disabled	Yes	No	Yes	Yes	No
Deleted	No	No	No	Yes	No
Verified	No	No	No	Yes	No

Credential Operations and States

The following table lists all credential operations and whether each operation is allowed on a specific state of the credential. If the state of the credential changes after an operation, then the table also provides the next state of the credential.

Note: *Allowed* indicates that the operation can be performed, but the state of the credential will not change after the operation.

Operation	**State**				
	Enabled	**Locked**	**Disabled**	**Deleted**	**Verified (for OTP only)**
Create	Not allowed	Not allowed	Not allowed	Allowed -> Enabled	Not applicable
Enable	Allowed -> Enabled	Allowed -> Enabled	Allowed -> Enabled	Not allowed	Not applicable
Disable	Allowed -> Disabled	Allowed -> Disabled	Allowed -> Disabled	Not allowed	Not applicable
Fetch	Allowed	Allowed	Allowed	Allowed	Allowed
FetchQnAConfiguration	Allowed	Allowed	Allowed	Allowed	Not applicable
Reset	Allowed -> Enabled	Allowed -> Enabled	Allowed -> Enabled	Not allowed	Not applicable
Reset Validity	Allowed	Allowed	Allowed	Not allowed	Not applicable
Download Credential	Allowed	Allowed	Allowed	Not allowed	Not applicable
Reset Custom Attributes	Allowed	Allowed	Allowed	Not allowed	Not applicable
Reissue	Allowed -> Enabled	Allowed -> Enabled	Allowed -> Enabled	Not allowed	Not applicable

Operation	**State**				
	Enabled	**Locked**	**Disabled**	**Deleted**	**Verified (for OTP only)**
Delete Unsigned Attributes (for ArcotID *only*)	Allowed	Allowed	Allowed	Not allowed	Not applicable
Set Unsigned Attributes (for ArcotID *only*)	Allowed	Allowed	Allowed	Not allowed	Not applicable
Add ArcotID Key Bag Elements	Allowed	Allowed	Allowed	Not allowed	Not applicable
Fetch ArcotID Key Bag Elements	Allowed	Allowed	Allowed	Not allowed	Not applicable
Delete ArcotID Key Bag Elements	Allowed	Allowed	Allowed	Not allowed	Not applicable
Delete	Allowed -> Deleted	Allowed -> Deleted	Allowed -> Deleted	Not allowed	Not applicable

Creating Credentials

The ArcotWebFortIssuanceSvc provides the CreateCredential operation that contains the elements to create the credentials for the user.

This section walks you through:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The CreateCredentialRequestMessage is used to create credentials for the users. The request message contains the elements for user information, common information that applies to all the credentials, and the credential-specific information.

This section lists the following input elements required for creating credentials:

- Common Input Elements (see page 206)
- ArcotID PKI Input Elements (see page 209)
- One-Time Password (OTP) Input Elements (see page 210)
- ArcotID OTP Input Elements (see page 210)
- EMV OTP Input Elements (see page 211)
- Questions and Answers (QnA) Input Elements (see page 211)
- Password Input Elements (see page 212)

Common Input Elements

The following table lists the user and common credential elements:

Element	Mandatory	Description
Common User Input		
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
userName	Yes	Specifies the unique identifier of the user whose credential has to be created.
orgName	No	Specifies the organization name to which the user belongs to.

Element	Mandatory	Description
additionalInput /pairs	No	AuthMinder's additionalInput element enables you to set additional inputs if you want to augment AuthMinder's authentication capability by specifying additional information. In such cases, you need to set the extra information in name-value *pairs.* ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information. **Note:** The additionalInput element is available at the end of the request message. You can add more than one of these elements.
Common Credential Input		
notes	No	Specifies the additional information that you want to maintain for each credential in your application. For example, if you do not want the user to download their ArcotID PKI on more than five systems, then you can create an attribute with this information. You can set the custom attributes in name-value pairs. ■ name Indicates the name with which you want to create the custom attribute. ■ value Indicates the corresponding value for the name.
profileName	No	Specifies the profile name of the credential. If it is not passed, then the default profile for the organization is used. If it is passed, then the profile name must be available at the organization level.
validityEndTime	No	Specifies the duration for which the credential must be valid. The timestamp format is according to the XS:Timestamp.

Element	Mandatory	Description
validityEndTimeEx	No	Specifies the duration for which the credential must be valid. The validityEndTimeEx element uses the ArcotDateType structure. The validityEndTimeEx element takes the following values: ■ year The year when the validity period begins or ends. ■ month The month when the validity period begins or ends. ■ day The day on which the validity period begins or ends ■ hour The hour at which the validity period begins or ends. ■ minute The minute at which the validity period begins or ends. ■ second The second at which the validity period begins or ends. ■ dateType The start date or end date of the validity period. Following are the supported date types: **1** Uses the current date of AuthMinder Server to set the validity or disable period. **2** Indicates that the credential will be valid forever and will not expire. **3** Uses a date that is specified by your application to set the validity or disable period. **4** Uses a relative date corresponding to the disable start date. For example, if the relative date is one month, then the disable end date would be one month after the disable start date.

Element	Mandatory	Description
disableStartTime	No	If your users want to go on a vacation or on long leave, then their credentials can be disabled only for this period, after which the credential will be enabled automatically. This feature facilitates credential activation without the user making a request to User Administrator (UA) to do so. The disableStartTime element is used to specify the duration from when the credential disable period must start. The values of this element are same as "validityEndTimeEx".
disableEndTime	No	Specifies when the credential disable period must end. The values of this element are same as "validityEndTimeEx".

ArcotID PKI Input Elements

ArcotID PKI is a secure software credential that provides two-factor authentication. An ArcotID PKI is a small data file that by itself can be used for strong authentication to a variety of clients such as Web or *Virtual Private Networks* (VPNs). The ArcotID PKI file is associated with a password. When the user authenticates with their ArcotID PKI, they have to provide this password and their ArcotID PKI must be present on the system from where the authentication is being performed.

The arcotIDInput element contains the elements that are required for creating ArcotID PKI. The following table lists the ArcotID PKI-specific elements:

Element	Mandatory	Description
password	No	The ArcotID PKI password that has to be set for the user.
unsignedAttributes	No	You can define ArcotID PKI attributes while or after creating an ArcotID PKI for the user. Such attributes are called unsigned attributes because these attributes (name-value pairs) are set in the unsigned portion of the ArcotID PKI. The attributes are defines as follows; ■ name The name of the unsigned attribute. ■ value The value corresponding to the name. The value must be specified in XS:base64Binary format. **Note:** If you add an attribute that already exists, then the current attributes will be overwritten by the new value.

Element	Mandatory	Description
fetchAttrbutes	No	The flag that indicates whether to fetch ArcotID PKI unsigned attributes in the response.

One-Time Password (OTP) Input Elements

One-Time Password is a credential that is generated by AuthMinder Server, and it does *not* require any credential-specific information.

OATH OTP Input Elements

AuthMinder supports hardware tokens that conform to *Open Authentication* (OATH) standards. To use these tokens, you need to set the oathInput element to generate OATH OTP. The following table lists the OATH OTP-specific elements:

Element	Mandatory	Description
tokenID	No	Specifies the unique identifier of the OATH token. The token identifier is used to generate the OATH OTP.
reUseToken	No	Specifies whether the token can be issued to different user, if unused for a long time. For example, if an employee leaves an organization, then their token can be reused for a new employee.

ArcotID OTP Input Elements

ArcotID OTP is a One-Time Password compliant to OATH standards. The client application that you build takes the user’s password as an input and generates passwords (also known as passcodes). The user uses this generated passcode at the Web application that is protected by ArcotID OTP authentication. Based on the authentication result, the user is granted access to the protected application.

The arcotOTPInput element contains the elements that are required for creating ArcotID OTP credential. The following table lists the ArcotID OTP-specific elements:

Element	Mandatory	Description
password	No	Specifies the ArcotID OTP password.

EMV OTP Input Elements

The emvInput element contains the elements that are required for supporting OTPs that are compliant to *Europay MasterCard VISA* (EMV) standards. The following table lists the EMV OTP-specific elements:

Element	**Mandatory**	**Description**
password	No	Specifies the EMV OTP password.

Questions and Answers (QnA) Input Elements

The qnaInput element contains the elements that are required for creating QnA credential. The following table lists the ArcotID OTP-specific elements:

Element	**Mandatory**	**Description**
qna	No	The questions and answers that are required for QnA authentication. ■ question The questions for which the user must provide answers. ■ answer Answers corresponding to the questions set.
qnaReset		The **qnaReset** element provides the following options: ■ deleteList Indicates that the specified question list will be deleted. ■ addList The list of questions and answers that have to be added. ■ changeAnswerList The list of questions and answers that have to be changed. ■ changeQuestionList The list of current questions that have to be changed and the list of newQuestions to which the old questions have to be changed to.

Password Input Elements

The upInput element contains the elements that are required for creating Password credential. The following table lists the elements for Password:

Element	Mandatory	Description
password	No	The password that the user needs to use for Password authentication.

Invoking the Web Service

To create credentials for a user:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the CreateCredential operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. Use the userName and orgName elements to fetch the details of the user for whom the credentials must be created.
3. Depending on the type of credential you want to create, use the respective <*CredentialName*>Input element to obtain the credential information.

   The input required for each credential is different. For example, password is needed for Password as well as ArcotID PKI, while questions and corresponding answers are required for QnA credentials.
4. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.

   This type provides the additional information that is set as a name-value pair.
5. Use CreateCredentialRequestMessage and construct the input message by using the details obtained in preceding steps.
6. Invoke the CreateCredential operation of the ArcotWebFortIssuanceSvc service to create the credentials.

   This operation returns an instance of the CreateCredentialResponseMessage that includes the credential and transaction details.

Interpreting the Response Message

For successful transactions, the response message, CreateCredentialResponseMessage returns the elements explained in the following table. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Method	Description
User and Common Credential Information	

Method	Description
userName	The unique identifier of the user.
orgName	The organization to which the user belongs to.
status	The status of the credential. Following are the supported values: ■ ACTIVE (1) The credential is active and can be used for authentication. ■ DISABLED (2) The credential is disabled by the administrator. ■ LOCKED (3) The credential is locked when the user consecutively fails to authenticate for the maximum number of negative attempts configured. For example if the maximum attempts configured is 3, then the third attempt with wrong credential will lock the credential. ■ DELETED (4) The credential of the user is deleted from the database. ■ EXPIRED (5) The credential of the user has expired. ■ VERIFIED (50) The credential is verified when the OTP submitted by the user is authenticated by AuthMinder Server successfully. **Note:** This status is applicable only for OTP.
remainingUsageCount	The number of times the credential can be used.
createTime	The time when the credential was created.
lastUpdatedTime	The time when the credential was updated last time.
validityStartTime	The timestamp from when the credential is valid.
validityEndTime	The date after which the credential expires.
disableStartTime	The time when the credential has to be disabled.
disableEndTime	The time when the disabled credential has to be enabled.
numberOfFailedAuthAttempts	The total number of failed authentication attempts permitted for the user.
lastSuccessAuthAttemptTime	The time when the last authentication attempt succeeded.
lastFailedAuthAttemptTime	The time when the last authentication attempt failed.
profileName	The profile name with which the credential was created.

Method	Description
profileVersion	The version number of the profile.
notes	The custom attributes that are set for the credential.
ArcotID Output	
unsignedAttributes	The unsigned attributes of the ArcotID PKI.
arcotID	The ArcotID PKI that is created for the user.
OTP Output	
otp	The OTP of the user.
usageCount	Number of times the OTP can be used.
OATH OTP Output	
tokenID	The unique identifier of the OATH token.
type	The type of OATH OTP, whether it is HOTP or TOTP.
counterOffSet	The OATH OTP count on the server.
ArcotOTP Output	
card	The ArcotID OTP of the user.
type	The type of ArcotID OTP, whether it is HOTP or TOTP.
counterOffSet	The ArcotID OTP count on the server.
EMV OTP Output	
card	The EMV OTP of the user.
counterOffSet	The EMV OTP count on the server.
QnA Output	
questions	The questions set for the user.
Transaction Details	
message	Indicates the status of the transaction.
reasonCode	Unique code that is sent by AuthMinder Server if the operation fails.
responseCode	Code returned by the SDK in case of errors.
transactionID	The unique identifier of the transaction.
additionalOutput	The output for the additionalInput that was passed to AuthMinder Server.

Disabling Credentials

User credentials can be disabled for a specified time interval. For example, if an employee goes for a long vacation, then the credentials of this user can be disabled to prevent any unauthorized access during their absence.

This section walks you through:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The DisableCredentialRequestMessage is used to disable credentials. The input elements for disabling the credentials are same as that explained in the "Creating Credentials" (see page 206) section. For more information about each element, refer to the tables listed in the "Creating Credentials (see page 206)" section.

Invoking the Web Service

To disable user credentials:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the DisableCredential operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. Use the userName and orgName elements to fetch the details of the user whose credentials must be disabled.
3. Depending on the type of credential you want to disable, use the respective <*CredentialName*>Input element to obtain the credential information.

   The input required for each credential is different. For example, password is needed for Password as well as ArcotID PKI, while questions and corresponding answers are required for QnA credentials.
4. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.

   This type provides the additional information that is set as a name-value pair.
5. Use DisableCredentialRequestMessage and construct the input message by using the details obtained in preceding steps.
6. Invoke the DisableCredential operation of the ArcotWebFortIssuanceSvc service to disable the credentials.

   This operation returns an instance of the DisableCredentialResponseMessage that includes the credential and transaction details.

Interpreting the Response Message

For successful transactions, the response message, DisableCredentialResponseMessage returns the elements explained in the table containing information about the elements that the response message, CreateCredentialResponseMessage, returns. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Enabling Credentials

The EnableCredential operation is used to activate the disabled or locked credential of a user. For example, a credential can be disabled or locked if a user tries to authenticate by using the wrong credential or exceeds the configured maximum number of allowed attempts.

This section walks you through:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The EnableCredentialRequestMessage is used to enable credentials. The input elements for enabling the credentials are same as that explained in the "Creating Credentials" (see page 206) section. For more information about each element, refer to the tables in the "Creating Credentials (see page 206)" section.

Invoking the Web Service

To enable user credentials:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the EnableCredential operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. Use the userName and orgName elements to fetch the details of the user whose credentials must be enabled.
3. Depending on the type of credential you want to enable, use the respective <*CredentialName*>Input element to obtain the credential information.

   The input required for each credential is different. For example, password is needed for Password as well as ArcotID PKI, while questions and corresponding answers are required for QnA credentials.
4. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.

   This type provides the additional information that is set as a name-value pair.
5. Use EnableCredentialRequestMessage and construct the input message by using the details obtained in preceding steps.
6. Invoke the EnableCredential operation of the ArcotWebFortIssuanceSvc service to enable the credentials.

   This operation returns an instance of the EnableCredentialResponseMessage that includes the credential and transaction details.

Interpreting the Response Message

For successful transactions, the response message, EnableCredentialResponseMessage returns the elements explained in the table containing information about the elements that the response message, CreateCredentialResponseMessage, returns. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Resetting Credentials

The Issuance Web service enables you to reset the credential. For example, you can reset the ArcotID PKI password or questions and answers.

This section walks you through:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The ResetCredentialRequestMessage is used to reset credentials. The input elements for enabling the credentials are same as that explained in the "Creating Credentials" (see page 206) section. For more information about each element, refer to the tables in the "Creating Credentials (see page 206)" section.

Invoking the Web Service

To reset user credentials:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the ResetCredential operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. Use the userName and orgName elements to fetch the details of the user whose credentials information you want to reset.
3. Depending on the type of credential you want to reset, use the respective <*CredentialName*>Input element to obtain the credential information.

   The input required for each credential is different. For example, password is needed for Password as well as ArcotID PKI, while questions and corresponding answers are required for QnA credentials.
4. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.

   This type provides the additional information that is set as a name-value pair.
5. Use ResetCredentialRequestMessage and construct the input message by using the details obtained in preceding steps.
6. Invoke the ResetCredential operation of the ArcotWebFortIssuanceSvc service to reset the credentials.

   This operation returns an instance of the ResetCredentialResponseMessage that includes the credential and transaction details.

Interpreting the Response Message

For successful transactions, the response message, ResetCredentialResponseMessage returns the elements explained in the table containing information about the elements that the response message, CreateCredentialResponseMessage, returns. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Fetching Credential Details

To read the details of the user credentials, you need to implement the FetchCredential operation. This section walks you through:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The FetchCredentialRequestMessage is used to read the credential details. The input elements for enabling the credentials are same as that explained in the "Creating Credentials" (see page 206) section. For more information about each element, refer to the tables listed in the "Creating Credentials (see page 206)" section.

Invoking the Web Service

To read a user's credential information:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the FetchCredential operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. Use the userName and orgName elements to fetch the details of the user whose credentials must be fetched.
3. Depending on the type of credential you want to fetch, use the respective <*CredentialName*>Input element to obtain the credential information.

   The input required for each credential is different. For example, password is needed for Password as well as ArcotID PKI, while questions and corresponding answers are required for QnA credentials.
4. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.

   This type provides the additional information that is set as a name-value pair.
5. Use FetchCredentialRequestMessage and construct the input message by using the details obtained in preceding steps.
6. Invoke the FetchCredential operation of the ArcotWebFortIssuanceSvc service to fetch the credentials.

   This operation returns an instance of the FetchCredentialResponseMessage that includes the credential and transaction details.

Interpreting the Response Message

For successful transactions, the response message, FetchCredentialResponseMessage returns the elements explained in the table containing information about the elements that the response message, CreateCredentialResponseMessage, returns. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Reissuing Credentials

The Issuance Web service enables you to re-create the credentials for the user. If the credential has been reissued for the user, then the user cannot log in by using their old credential. This section walks you through:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The ReissueCredentialRequestMessage is used to reissue the credentials. The input elements for reissuing the credentials are same as that explained in the "Creating Credentials" (see page 206) section. For more information about each element, refer to the tables listed in the "Creating Credentials (see page 206)" section.

Invoking the Web Service

To reissue credentials for a user:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the ReissueCredential operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.

2. Use the userName and orgName elements to fetch the details of the user whose credentials must be reissued.

3. Depending on the type of credential you want to reissue, use the respective <*CredentialName*>Input element to obtain the credential information.

   The input required for each credential is different. For example, password is needed for Password as well as ArcotID PKI, while questions and corresponding answers are required for QnA credentials.

4. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.

   This type provides the additional information that is set as a name-value pair.

5. Use ReissueCredentialRequestMessage and construct the input message by using the details obtained in preceding steps.

6. Invoke the ReissueCredential operation of the ArcotWebFortIssuanceSvc service to reissue the credentials.

   This operation returns an instance of the ReissueCredentialResponseMessage that includes the credential and transaction details.

Interpreting the Response Message

For successful transactions, the response message, ReissueCredentialResponseMessage returns the elements explained in the table containing information about the elements that the response message, CreateCredentialResponseMessage, returns. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Resetting Credential Validity

Issued credentials are valid for the period that is specified at the time they are created. The ResetCredentialValidity operation enables you to reset the validity period of the credential before it expires. This operation is used to either extend or reduce the validity period of the credential, but it does *not* reset the password or any other credential attributes.

This section walks you through:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The ResetCredentialValidityRequestMessage is used to reset the validity of the credentials. The input elements for resetting the credentials validity are same as that explained in the "Creating Credentials" (see page 206) section. For more information about each element, refer to the tables listed in the "Creating Credentials (see page 206)" section.

Invoking the Web Service

To reset the validity of user credentials:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the ResetCredentialValidity operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. Use the userName and orgName elements to fetch the details of the user whose credential validity must be reset.
3. Depending on the type of credential that has to be reset, use the respective <*CredentialName*>Input element to obtain the credential information.

   The input required for each credential is different. For example, password is needed for Password as well as ArcotID PKI, while questions and corresponding answers are required for QnA credentials.
4. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.

   This type provides the additional information that is set as a name-value pair.
5. Use ResetCredentialValidityRequestMessage and construct the input message by using the details obtained in preceding steps.
6. Invoke the ResetCredentialValidity operation of the ArcotWebFortIssuanceSvc service to reset the credential validity.

This operation returns an instance of the ResetCredentialValidityResponseMessage that includes the credential and transaction details.

Interpreting the Response Message

For successful transactions, the response message, ResetCredentialValidityResponseMessage returns the elements explained in the table containing information about the elements that the response message, CreateCredentialResponseMessage, returns. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Resetting Custom Attributes

The custom attributes associated with the credentials can be reset. The ResetCredentialNotes operation enables you to reset the custom attributes of the credential.

This section walks you through:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The ResetCredentialNotesRequestMessage is used to reset custom attributes. The input elements for resetting the custom attributes associated with the credentials are same as that explained in the "Creating Credentials" (see page 206) section. For more information about each element, refer to the tables listed in the "Creating Credentials (see page 206)" section.

Invoking the Web Service

To reset the custom attributes of user credentials:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the ResetCredentialNotes operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. Use the userName and orgName elements to fetch the details of the user whose credential custom attributes must be reset.
3. Depending on the type of credential for which the attributes have to be reset, use the respective <*CredentialName*>Input element to obtain the credential information.

   The input required for each credential is different. For example, password is needed for Password as well as ArcotID PKI, while questions and corresponding answers are required for QnA credentials.
4. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.

   This type provides the additional information that is set as a name-value pair.
5. Use ResetCredentialNotesRequestMessage and construct the input message by using the details obtained in preceding steps.
6. Invoke the ResetCredentialNotes operation of the ArcotWebFortIssuanceSvc service to reset the credential custom attributes.

This operation returns an instance of the ResetCredentialNotesResponseMessage that includes the credential and transaction details.

Interpreting the Response Message

For successful transactions, the response message, ResetCredentialNotesResponseMessage returns the elements explained in the table containing information about the elements that the response message, CreateCredentialResponseMessage, returns. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Fetching QnA Configuration

The number of questions that the user must set for QnA authentication might vary for every organization. This section explains how to use Web services to fetch the questions that are set for each organization. It covers the following topics:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The FetchQnAConfigurationRequestMessage is used to get the number of questions that must be set by the user. It contains the elements listed in the following table:

Element	**Mandatory**	**Description**
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
orgName	No	Specifies the organization name to which the user belongs to.
profileName	No	Specifies the profile name of the credential. If it is not passed, then the default profile for the organization is used.
fetchQuestions	No	Specifies whether to fetch the user questions. Following are the possible values: ■ 0: If you do not want to fetch the questions. ■ 1: If you want to fetch the questions.

Element	Mandatory	Description
additionalInput/pairs	No	AuthMinder's additionalInput element enables you to set additional inputs if you want to augment AuthMinder's authentication capability by specifying additional information. In such cases, you need to set the extra information in *name-value* pairs. ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information.

Invoking the Web Service

Perform the following steps to fetch QnA configuration:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the FetchQnAConfiguration operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. Use the orgName element to fetch the QnA details configured for the organization.
3. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.

   This type provides the additional information that is set as a name-value pair.
4. Use FetchQnAConfigurationRequestMessage and construct the input message by using the details obtained in preceding steps.
5. Invoke the FetchQnAConfiguration operation of the ArcotWebFortIssuanceSvc service to fetch the number of questions.

   This operation returns an instance of the FetchQnAConfigurationResponseMessage that includes the credential and transaction details.

Interpreting the Response Message

For successful transactions, the response message, FetchQnAConfigurationResponseMessage returns the elements explained in the following table. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See , "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	**Description**
orgName	The organization name to which the user belongs to.
profileName	The profile name of the credential.
minQuestions	The minimum number of questions that must be set by the user.
maxQuestions	The maximum number of questions a user can set.
questions	The questions for authenticating the users. **Note:** The questions are returned if fetchQuestions flag is enabled in the input.

Element	Description
transactionDetails	The transactions details include the following: ■ message A string that defines the status of the operation. ■ reasonCode Unique code that is sent by AuthMinder Server if the operation fails. ■ responseCode Unique code that is sent by AuthMinder Server if the operation fails. ■ transactionID Unique identifier of the transaction. ■ additionalOutput The output for the additionalInput that was passed to AuthMinder Server.

Setting Unsigned Attributes

This sections explains the Web service that must be used to set the ArcotID PKI unsigned attributes. It walks you through following topics:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: This operation is applicable *only* for ArcotID PKI credential.

Preparing the Request Message

The SetArcotIDUnsignedAttributesRequestMessage is used to set the unsigned attributes of the ArcotID PKI. It contains the elements listed in the following table.

Element	Mandatory	Description
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
userName	Yes	Specifies the unique identifier of the user for whom the ArcotID PKI unsigned attributes have to be defined.
orgName	No	Specifies the organization name to which the user belongs to.

Element	Mandatory	Description
unsignedAttributes	Yes	Specifies the ArcotID PKI unsigned attributes in name-value pairs. ■ name Indicates the name with which you want to create the unsigned attribute. ■ value Indicates the corresponding value for the name.
additionalInput	No	Specifies the extra information that must be sent to AuthMinder Server in name-value pairs. ■ name Indicates the name with which you want to pass additional information to the server. ■ value Indicates the corresponding value for the name.

Invoking the Web Service

To set the unsigned attributes for the ArcotID PKI of user, you need to implement the SetArcotIDUnsignedAttributes operation.

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the SetArcotIDUnsignedAttributes operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. Use the SetArcotIDUnsignedAttributes element to set the ArcotID PKI unsigned attributes.
3. Use the userName and orgName elements to fetch the details of the user whose ArcotID PKI unsigned attributes you want to set.
4. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.

   This type provides the additional information that is set as a name-value pair.
5. Use SetArcotIDUnsignedAttributesRequestMessage and construct the input message by using the details obtained in preceding steps.
6. Invoke the SetArcotIDUnsignedAttributes operation of the ArcotWebFortIssuanceSvc service to set the ArcotID PKI unsigned attributes.

   This operation returns an instance of the SetArcotIDUnsignedAttributesResponseMessage that specifies the transaction details.

Interpreting the Response Message

For successful transactions, the response message, SetArcotIDUnsignedAttributesResponseMessage returns the elements explained in the following table. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	**Description**
message	A string that defines the status of the operation.
reasonCode	Unique code that is sent by AuthMinder Server if the operation fails.
responseCode	Unique code that is sent by AuthMinder Server if the operation fails.
transactionID	Unique identifier of the transaction.
additionalOutput	The output for the additionalInput that was passed to AuthMinder Server.

Deleting Unsigned Attributes

This sections explains the Web service that must be used to delete the ArcotID PKI unsigned attributes. It walks you through following topics:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: This operation is applicable *only* for ArcotID PKI credential.

Preparing the Request Message

The DeleteArcotIDUnsignedAttributesRequestMessage is used to delete the unsigned attributes of the ArcotID PKI. It contains the elements listed in the following table.

Element	Mandatory	Description
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
userName	Yes	Specifies the unique identifier of the user whose ArcotID PKI unsigned attributes have to be deleted.
orgName	No	Specifies the organization name to which the user belongs to.
unsignedAttributeNames	Yes	Specifies the ArcotID PKI unsigned attribute name.
additionalInput	No	Specifies the extra information that must be sent to AuthMinder Server in name-value pairs. ■ name Indicates the name of the unsigned attribute that you want to delete. ■ value Indicates the corresponding value for the name.

Invoking the Web Service

To delete the unsigned attributes for the ArcotID PKI of a user, you need to implement the DeleteArcotIDUnsignedAttributes operation.

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the DeleteArcotIDUnsignedAttributes operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. Use the userName and orgName elements to fetch the details of the user whose ArcotID PKI unsigned attributes you want to delete.
3. Use the DeleteArcotIDUnsignedAttributes element to fetch the unsigned attributes that you want to delete.
4. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.

   This type provides the additional information that is set as a name-value pair.
5. Use DeleteArcotIDUnsignedAttributesRequestMessage and construct the input message by using the details obtained in preceding steps.
6. Invoke the DeleteArcotIDUnsignedAttributes operation of the ArcotWebFortIssuanceSvc service to delete the ArcotID PKI unsigned attributes.

   This operation returns an instance of the DeleteArcotIDUnsignedAttributesResponseMessage that specifies the transaction details.

Interpreting the Response Message

For successful transactions, the response message, DeleteArcotIDUnsignedAttributesResponseMessage returns the elements explained in the following table. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	**Description**
message	A string that defines the status of the operation.
reasonCode	Unique code that is sent by AuthMinder Server if the operation fails.
responseCode	Unique code that is sent by AuthMinder Server if the operation fails.
transactionID	Unique identifier of the transaction.
additionalOutput	The output for the additionalInput that was passed to AuthMinder Server.

Adding Elements to ArcotID PKI Key Bag

ArcotID PKI can *also* be used to securely store the Open PKI keys and certificates. These keys are typically used for different applications or operations such as, email signing (S/MIME), document signing, and certificate-based authentication (open PKI).

The location where the open PKI keys and certificates are stored in the ArcotID PKI is called *key bag* or *key vault*.

This sections explains the Web service that must be used to add keys and certificates to the ArcotID PKI key bag. It walks you through following topics:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: This operation is applicable *only* for ArcotID PKI credential.

Preparing the Request Message

The ArcotIDKeyBagAddElementsRequestMessage is used to add keys and certificates to the ArcotID PKI key bag. It contains the elements listed in the following table.

Element	Mandatory	Description
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
userName	Yes	Specifies the unique identifier of the user whose certificates have to be added to their ArcotID PKI key bag.
orgName	No	Specifies the organization name to which the user belongs to.
profileName	No	Specifies the profile name of the credential. If it is not passed, then the default profile for the organization is used. If it is passed, then the profile name must be available at the organization level.
elementSelection	No	Contains the following elements: ■ selectCertEncoding If you enable this option, then AuthMinder Server returns the certEncoding in response. ■ selectCertsDetails If you enable this option, then AuthMinder Server returns the certDetails in response.

Element	Mandatory	Description
additionalInput	No	Specifies the extra information that must be sent to AuthMinder Server in name-value pairs. ■ name Indicates the name of the unsigned attribute that you want to delete. ■ value Indicates the corresponding value for the name.
elements	No	Specifies the PKCS#12 file elements: ■ certEncoding: The PKCS#12 file is base-64 encoded format. ■ password:The passsword for the PKCS#12 file.

Invoking the Web Service

To add the keys and certificates to the ArcotID PKI key bag, you need to implement the ArcotIDKeyBagAddElements operation.

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the ArcotIDKeyBagAddElements operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. Use the userName and orgName elements to fetch the details of the user whose ArcotID PKI unsigned attributes you want to add.
3. Use the elementSelection and elements element to fetch the certificate information.
4. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.

   This type provides the additional information that is set as a name-value pair.
5. Use ArcotIDKeyBagAddElementsRequestMessage and construct the input message by using the details obtained in preceding steps.
6. Invoke the ArcotIDKeyBagAddElements operation of the ArcotWebFortIssuanceSvc service to add the certificates.

   This operation returns an instance of the ArcotIDKeyBagAddElementsResponseMessage that specifies the transaction details.

Interpreting the Response Message

For successful transactions, the response message, ArcotIDKeyBagAddElementsResponseMessage returns the elements explained in the following table. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	Description
transactionDetails/ message	A string that defines the status of the operation.
transactionDetails/ reasonCode	Unique code that is sent by AuthMinder Server if the operation fails.
transactionDetails/ responseCode	Unique code that is sent by AuthMinder Server if the operation fails.
transactionDetails/ transactionID	Unique identifier of the transaction.
transactionDetails/ additionalOutput	The output for the additionalInput that was passed to AuthMinder Server.

Element	Description
certificates/certEncoding	The encoding details that were requested in the input.
certificates/certsDetails	Includes the following certificate details: ■ elementId: The identifier that denotes the unsigned attribute. ■ issuerName: The name of the issuer whose issued the certificate. ■ serialNumber: The serial number of the certificate. ■ certSubject: The subject of the certificate. ■ certValidFrom: The date from when the certificate is valid. ■ certValidTo: The date when the certificate expires. ■ hasPrivatekey: Indicates if the certificate contains the private key.

Fetching ArcotID PKI Key Bag Elements

This section walks you through the following topics for fetching the certificate details that are stored in the key bag:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: This operation is applicable *only* for ArcotID PKI credential.

Preparing the Request Message

The ArcotIDKeyBagGetElementsRequestMessage is used to fetch the details of keys and certificates that are stored in the ArcotID PKI key bag. It contains the elements listed in the following table.

Element	Mandatory	Description
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
userName	Yes	Specifies the unique identifier of the user whose ArcotID PKI key bag elements have to be fetched.
orgName	No	Specifies the organization name to which the user belongs to.

Element	Mandatory	Description
profileName	No	Specifies the profile name of the credential. If it is not passed, then the default profile for the organization is used. If it is passed, then the profile name must be available at the organization level.
elementSelection	No	Contains the following elements: ■ selectCertEncoding If you enable this option, then AuthMinder Server returns the certEncoding in response. ■ selectCertsDetails If you enable this option, then AuthMinder Server returns the certDetails in response.
additionalInput	No	Specifies the extra information that must be sent to AuthMinder Server in name-value pairs. ■ name Indicates the name of the unsigned attribute that you want to delete. ■ value Indicates the corresponding value for the name.

Invoking the Web Service

To fetch the keys and certificates stored in the ArcotID PKI key bag, you need to implement the ArcotIDKeyBagGetElements operation.

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the ArcotIDKeyBagGetElements operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. Use the userName and orgName elements to fetch the details of the user whose ArcotID PKI unsigned attributes you want to delete.
3. Use the elementSelection element to identify the certificate elements that you want to fetch.
4. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.

   This type provides the additional information that is set as a name-value pair.
5. Use ArcotIDKeyBagGetElementsRequestMessage and construct the input message by using the details obtained in preceding steps.
6. Invoke the ArcotIDKeyBagGetElements operation of the ArcotWebFortIssuanceSvc service to add the certificates.

   This operation returns an instance of the ArcotIDKeyBagGetElementsResponseMessage that specifies the transaction details.

Interpreting the Response Message

For successful transactions, the response message, ArcotIDKeyBagGetElementsResponseMessage returns the elements explained in the following table. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	**Description**
transactionDetails/ message	A string that defines the status of the operation.
transactionDetails/ reasonCode	Unique code that is sent by AuthMinder Server if the operation fails.
transactionDetails/ responseCode	Unique code that is sent by AuthMinder Server if the operation fails.
transactionDetails/ transactionID	Unique identifier of the transaction.
transactionDetails/ additionalOutput	The output for the additionalInput that was passed to AuthMinder Server.

Element	Description
certificates/certEncoding	The encoding details that were requested in the input.
certificates/certsDetails	Includes the following certificate details: ■ elementId: The identifier that denotes the unsigned attribute. ■ issuerName: The name of the issuer whose issued the certificate. ■ serialNumber: The serial number of the certificate. ■ certSubject: The subject of the certificate. ■ certValidFrom: The date from when the certificate is valid. ■ certValidTo: The date when the certificate expires. ■ hasPrivatekey: Indicates if the certificate contains the private key.

Deleting ArcotID PKI Key Bag Elements

The certificates are valid for a certain period, after which they expire. The expired certificates cannot be used for any operation. In such cases, you can delete the expired certificates stored in the key bag, and import new certificates.

This section walks you through the topics for deleting certificates stored in the ArcotID PKI key bag:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: This operation is applicable *only* for ArcotID PKI credential.

Preparing the Request Message

The ArcotIDKeyBagDeleteElementsRequestMessage is used to delete the keys and certificates that are stored in the ArcotID PKI key bag. It contains the elements listed in the following table.

Element	Mandatory	Description
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Element	**Mandatory**	**Description**
userName	Yes	Specifies the unique identifier of the user whose key bag elements have to be deleted.
orgName	No	Specifies the organization name to which the user belongs to.
profileName	No	Specifies the profile name of the credential. If it is not passed, then the default profile for the organization is used. If it is passed, then the profile name must be available at the organization.
elementSelection	No	Contains the following elements: ■ selectCertEncoding If you enable this option, then AuthMinder Server returns the certEncoding in response. ■ selectCertsDetails If you enable this option, then AuthMinder Server returns the certDetails in response.
additionalInput	No	Specifies the extra information that must be sent to AuthMinder Server in name-value pairs. ■ name Indicates the name of the unsigned attribute that you want to delete. ■ value Indicates the corresponding value for the name.
elementIds	Yes	The unique identifier of the elements that you want to delete.

Invoking the Web Service

To delete the keys and certificates stored in the ArcotID PKI key bag, you need to implement the ArcotIDKeyBagDeleteElements operation.

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the ArcotIDKeyBagDeleteElements operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. Use the userName and orgName elements to fetch the details of the user whose ArcotID PKI unsigned attributes you want to delete.
3. Use the elementSelection and elementIds element to identify the certificate elements that you want to delete.
4. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.

   This type provides the additional information that is set as a name-value pair.
5. Use ArcotIDKeyBagDeleteElementsRequestMessage and construct the input message by using the details obtained in preceding steps.
6. Invoke the ArcotIDKeyBagDeleteElements operation of the ArcotWebFortIssuanceSvc service to add the certificates.

   This operation returns an instance of the ArcotIDKeyBagDeleteElementsResponseMessage that specifies the transaction details.

Interpreting the Response Message

For successful transactions, the response message, ArcotIDKeyBagDeleteElementsResponseMessage returns the elements explained in the following table. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	Description
message	A string that defines the status of the operation.
reasonCode	Unique code that is sent by AuthMinder Server if the operation fails.
responseCode	Unique code that is sent by AuthMinder Server if the operation fails.
transactionID	Unique identifier of the transaction.
additionalOutput	The output for the additionalInput that was passed to AuthMinder Server.

Downloading Credentials

To download the credential to your device, you need to use the DownloadCredential operation. This section walks you through:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Note: This operation is valid *only* for ArcotID PKI, ArcotID OTP-OATH, and ArcotID OTP-EMV credentials.

Preparing the Request Message

The DownloadCredentialRequestMessage is used to download the credentials. The input elements for downloading the credentials are same as that explained in the "Creating Credentials" (see page 206) section. For more information about each element, refer to the tables listed in the "Creating Credentials (see page 206)" section.

Invoking the Web Service

To delete the credentials of a user:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the DownloadCredential operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.

2. Use the userName and orgName elements to fetch the details of the user whose credential you want to download.

3. Depending on the type of credential you want to delete, use the respective <*CredentialName*>Input element to obtain the credential information.

   The input required for each credential is different. For example, password is needed for ArcotID PKI.

4. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.

   This type provides the additional information that is set as a name-value pair.

5. Use DownloadCredentialRequestMessage and construct the input message by using the details obtained in preceding steps.

6. Invoke the DownloadCredential operation of the ArcotWebFortIssuanceSvc service to delete the credential.

   This operation returns an instance of the DownloadCredentialResponseMessage that includes the credentials and the transaction details.

Interpreting the Response Message

For successful transactions, the response message, DownloadCredentialResponseMessage returns the elements explained in the table containing information about the elements that the response message, CreateCredentialResponseMessage, returns. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Deleting Credentials

This section walks you through:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The DeleteCredentialRequestMessage is used to delete the credentials. The input elements for deleting the credentials are same as that explained in the "Creating Credentials" (see page 206) section. For more information about each element, refer to the tables listed in the "Creating Credentials (see page 206)" section.

Invoking the Web Service

To delete the credentials of a user:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the DeleteCredential operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. Use the userName and orgName elements to fetch the details of the user whose credential you want to delete.
3. Depending on the type of credential you want to delete, use the respective <*CredentialName*>Input element to obtain the credential information.

   The input required for each credential is different. For example, password is needed for Password as well as ArcotID PKI, while questions and corresponding answers are required for QnA credentials.
4. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.

   This type provides the additional information that is set as a name-value pair.
5. Use DeleteCredentialRequestMessage and construct the input message by using the details obtained in preceding steps.
6. Invoke the DeleteCredential operation of the ArcotWebFortIssuanceSvc service to delete the credential.

   This operation returns an instance of the DeleteCredentialResponseMessage that includes the credentials and the transaction details.

Interpreting the Response Message

For successful transactions, the response message, DeleteCredentialResponseMessage returns the elements explained in the table containing information about the elements that the response message, CreateCredentialResponseMessage, returns. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Chapter 9: Integrating ArcotID PKI Client with Your Application

The **ArcotID PKI Client** is a software that is used by the end user to sign the challenge provided by AuthMinder Server. If you are planning to implement ArcotID PKI-based authentication, then you must integrate ArcotID PKI Client with your application before you call ArcotID PKI authentication APIs. This chapter provides information on different client types, details on how to integrate them with your application, and lists the APIs provided by ArcotID PKI Client. It covers the following topics:

- ArcotID PKI Client Overview (see page 249)
- Copying ArcotID PKI Client Files (see page 250)
- ArcotID PKI Client APIs (see page 251)

ArcotID PKI Client Overview

The ArcotID PKI Client is used for signing the AuthMinder-issued challenge at the user end, but it also facilitates the download of the user's ArcotID PKI. To support a wide variety of end user environments, the ArcotID PKI Client is available as a Flash client and as a signed Java applet. Each client type offers different levels of convenience and capabilities. The degree of user interaction and administration rights required for configuration vary depending on the client selected.

Flash Client

This implementation of ArcotID PKI Client runs in any Web browser that has Adobe Flash Player (version 9 or higher) installed.

Note: If you are using ArcotID PKI Flash Client for ArcotID PKI operations, then the application serving the Flash client must be enabled for HTTPS.

Signed Java Applet

This implementation of the ArcotID PKI Client can run in any Web browser that has Sun Java Runtime Environment (JRE) installed.

Note: When using the signed Java applet, the user will be presented with a security message that requires the user to accept the signed applet before it is invoked.

Copying ArcotID PKI Client Files

ArcotID PKI Client is an end-user system component. Therefore based on the client type that you are planning to use, you must include the relevant files to the correct locations on the system where the application is running.

This section discusses the files that needs to be packaged with the application:

- For Flash Client (see page 250)
- For Java Signed Applet (see page 251)

For Flash Client

The Flash client package contains the following files:

- arcotclient.js

  Contains the ArcotID PKI Flash Client APIs.

- ArcotIDClient.swf

  Contains the ArcotID PKI Flash Client implementation.

To configure a Flash Client:

1. Copy arcotclient.js and ArcotIDClient.swf files to an appropriate directory within your application home.
2. Include the following JavaScript code in the Web page of your application from where the APIs will be invoked:

   ```
   <script type="text/javascript"
   src="location_to_arcotclient.js"></script>
   ```

   In the preceding code snippet, replace location_to_arcotclient.js with the path to arcotclient.js.
3. Ensure that in all application pages, ArcotIDClient.swf is referred with same URL.

For Java Signed Applet

The Java Signed Applet client package contains the following files:

- arcotclient.js

  Contains the Java Signed Applet client APIs.

- ArcotApplet.jar (for Sun JRE)

  Contains the Java Signed Applet client implementation.

To configure the Java Signed Applet Client:

1. Copy arcotclient.js and ArcotApplet.jar to an appropriate directory within your application home.
2. Include the following JavaScript code in the relevant Web page of your application:

   ```
   <script type="text/javascript"
   src="location_to_arcotclient.js"></script>
   ```

   In the preceding code snippet, replace location_to_arcotclient.js with the path to arcotclient.js.
3. Ensure that in all application pages, the Java Applet is referred with same URL.

ArcotID PKI Client APIs

If you are implementing ArcotID PKI authentication, then your application must integrate with ArcotID PKI Client APIs for:

- Downloading ArcotID PKI (see page 252)
- Signing the Challenge (see page 252)

Downloading ArcotID PKI

To download the ArcotID PKI from the application to the end-user system, you must use the ImportArcotID() function. This function takes the base-64 encoded string of the ArcotID PKI that has to be downloaded and the storage mode as the input parameters.

The ArcotID PKI can be temporarily downloaded for the current session or can be downloaded permanently. This storage mode is specified by the storage medium selected for storing the ArcotID PKI. An ArcotID PKI can be stored in any of the following:

- Hard Disk
- Universal Serial Bus (USB)
- Memory

The downloaded ArcotID PKI is saved with the .aid extension. The name of the ArcotID PKI file is derived from the hash value of user name, organization name, and domain name.

Signing the Challenge

The challenge from AuthMinder Server must be signed by using the SignChallengeEx() function of the client API.

Note: Refer to *CA ArcotID Client Reference Guide* for more information on the API details.

Chapter 10: Authenticating Users

This chapter describes the operations that are used for different authentication methods supported by AuthMinder. This chapter covers the following topics:

- ArcotID PKI Authentication (see page 254)
- Questions and Answers Authentication (see page 264)
- Password Authentication (see page 269)
- One-Time Password Authentication (see page 277)
- OATH One-Time Password Authentication (see page 280)
- OATH One-Time Password Synchronization (see page 283)
- ArcotID OTP (ArcotID OTP-OATH) Authentication (see page 286)
- ArcotID OTP (ArcotID OTP-OATH) Synchronization (see page 289)
- EMV OTP (ArcotID OTP-EMV) Authentication (see page 292)
- EMV OTP (ArcotID OTP-EMV) Synchronization (see page 295)
- Verifying Password Type Credentials (see page 298)
- Verifying the Authentication Tokens (see page 301)
- Fetching the PAM (see page 304)

To perform the operations discussed in this chapter, you need to use the ArcotWebFortAuthSvc.wsdl file.

ArcotID PKI Authentication

ArcotID PKI is a challenge-response type of authentication, where AuthMinder Server provides a challenge. The signed challenge is sent by the ArcotID PKI Client to AuthMinder Server through your application. This section explains how to download the ArcotID PKI and then use it for authentication:

- Step 1: ArcotID PKI Download (see page 254)
- Step 2: ArcotID PKI Authentication (see page 258)

For successful ArcotID PKI authentication, you must ensure that you have integrated ArcotID PKI Client with your application, as discussed in chapter, "Integrating ArcotID PKI Client with Your Application" (see page 249).

Note: The ArcotID PKI download and authentication can be achieved in multiple ways, see chapter, "Understanding AuthMinder WorkFlows" (see page 21) for more information. This section focuses on the operations that are used to complete these tasks.

Step 1: ArcotID PKI Download

To perform ArcotID PKI authentication, the ArcotID PKI of the user has to be present on the system from where the authentication request is originating. If the ArcotID PKI is not present, then it needs to be downloaded to the system. In such a case the user must perform a secondary authentication before the ArcotID PKI is downloaded.

The ArcotWebFortAuthSvc provides the GetArcotID operation that contains the elements to download the ArcotID PKI of the users.

This section covers the following topics for downloading ArcotID PKI of the users:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The GetArcotIDRequestMessage is used to send the ArcotID PKI download request to AuthMinder Server. The following table lists the elements of this message:

Element	Mandatory	Description
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Element	Mandatory	Description
userName	Yes	The unique identifier of the user whose ArcotID PKI has to be downloaded.
orgName	No	The organization name to which the user belongs to.
additionalInput /pairs	No	AuthMinder's additionalInput element enables you to set additional inputs if you want to augment AuthMinder's authentication capability by specifying additional information. In such cases, you need to set the extra information in name-value *pairs.* ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information. **Note:** The additionalInput element is available at the end of the request message. You can add more than one of these elements.

Invoking the Web Service

To download the ArcotID PKI:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the GetArcotID operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.
3. Use GetArcotIDRequestMessage and construct the input message. See the table in the preceding section.
4. Invoke the GetArcotID operation of the ArcotWebFortAuthSvc service to fetch the ArcotID PKI of the user to your application.

   This operation returns an instance of the GetArcotIDResponseMessage, which provides the ArcotID PKI of the user and transaction details. For more information, see the table containing information about the elements that the response message, GetArcotIDResponseMessage, returns.
5. The user's ArcotID PKI is set in the HTML or Java Server Page (JSP).
6. Invoke the ImportArcotID client-side API to download the ArcotID PKI from your application to the end user's system.

Note: Refer to *CA ArcotID Client Reference Guide* for more information on the ImportArcotID function. ArcotID PKI Client provides the SDK in JavaScript programming language.

Interpreting the Response Message

For successful transactions, the response message, GetArcotIDResponseMessage returns the elements explained in the following table. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	**Description**
arcotID	The ArcotID PKI of the user in the base-64 encoded format.

Element	Description
transactionDetails	Contains the following details of the transaction: ■ message A string that defines the status of the operation. ■ reasoncode Unique code that is sent by AuthMinder Server if the operation fails. ■ responseCode Unique code that is sent by AuthMinder Server if the operation fails. ■ transactionID Unique identifier of the transaction. ■ additionalOutput The output for the additionalInput that was passed to AuthMinder Server.

Step 2: ArcotID PKI Authentication

ArcotID PKI is based on challenge response authentication. The ArcotID PKI Client first fetches the challenge from AuthMinder Server, signs it, and then sends a request to AuthMinder Server to verify the signed challenge.

This section walks you through:

- Preparing Request Messages
- Invoking the Web Service
- Interpreting Response Messages

Preparing Request Messages

For ArcotID PKI authentication, you must prepare the following request messages:

- Fetching Challenge
- Verifying Signed Challenge

Fetching Challenge

The GetArcotIDChallengeRequestMessage is used to fetch the challenge from AuthMinder Server. The following table lists the elements of this message:

Element	Mandatory	Description
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Element	Mandatory	Description
additionalInput/pairs	No	AuthMinder’s additionalInput element enables you to set additional inputs if you want to augment AuthMinder’s authentication capability by specifying additional information. In such cases, you need to set the extra information in name-value *pairs.* ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information.

Verifying Signed Challenge

The VerifySignedChallengeRequestMessage is used to verify the signed challenge. The following table lists the elements of this message:

Element	Mandatory	Description
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
orgName	No	The organization name to which the user belongs to.
signedChallenge	Yes	The challenge that is signed by the ArcotID PKI Client by using user’s ArcotID PKI password.
tokenType	No	The type of authentication token that is expected from AuthMinder Server after successful authentication. See "Verifying the Authentication Tokens" (see page 301) for more information.

Element	**Mandatory**	**Description**
additionalInput/pairs	No	AuthMinder's additionalInput element enables you to set additional inputs if you want to augment AuthMinder's authentication capability by specifying additional information. In such cases, you need to set the extra information in *name-value* pairs. ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information.

Invoking the Web Service

To perform ArcotID PKI authentication:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the GetArcotIDChallenge operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. Use GetArcotIDChallengeRequestMessage and construct the input message.
3. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.
4. Invoke the GetArcotIDChallenge operation of the ArcotWebFortAuthSvc service to retrieve the challenge from AuthMinder Server.

   This operation returns GetArcotIDChallengeResponseMessage, which has the transaction details and also the challenge from the server.
5. The challenge is sent to the end user through HTML Page.
6. Invoke the ArcotID PKI Client-side method, SignChallengeEx() to sign the challenge.

   The application collects the ArcotID PKI password, and the challenge is signed by the ArcotID PKI Client using the ArcotID PKI password.
7. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the VerifyArcotIDSignedChallenge operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
8. Use VerifySignedChallengeRequestMessage and construct the input message.
9. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.
10. Use VerifyArcotIDSignedChallenge operation of the ArcotWebFortAuthSvc service to verify the signed challenge. Optionally, you can also specify the token type that must be returned to the user *after* successful authentication by using the tokenTpe element.

    This operation returns an instance of the VerifySignedChallengeResponseMessage, which provides the transaction details, credential details, and token information.

Interpreting Response Messages

Following are the response messages that are returned as part of ArcotID PKI authentication:

- Fetch Challenge Response Message
- Verify Signed Challenge Response Message

Fetch Challenge Response Message

For successful transactions, the response message, GetArcotIDChallengeResponseMessage returns the elements explained in the following table. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	Description
challenge	Challenge returned by AuthMinder Server.
transactionDetails	Contains the following transaction details: ■ message A string that defines the status of the operation. ■ reasoncode Unique code that is sent by AuthMinder Server if the operation fails. ■ responseCode Unique code that is sent by AuthMinder Server if the operation fails. ■ transactionID Unique identifier of the transaction. ■ additionalOutput The output for the additionalInput that was passed to AuthMinder Server.

Verify Signed Challenge Response Message

For successful transactions, the response message, VerifySignedChallengeResponseMessage returns the elements explained in the following table. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Method	Description
userName	The name of the authenticating user.
orgName	The organization to which the user belongs to.
authToken	The authentication token returned by AuthMinder Server after successful authentication.
tokenType	The type of authentication token that is returned by AuthMinder Server after successful authentication. See "Verifying the Authentication Tokens" (see page 301) for more information.
daysLeftToExpire	The number of days after which the credential expires.
status	The status of the credential.

Method	Description
remainingUsageCount	The number of times the credential can be used.
createTime	The time when the credential was created.
lastUpdatedTime	The time when the credential was updated last time.
validityStartTime	The date from when the credential is valid.
validityEndTime	The date after which the credential expires.
disableStartTime	The time when the credential has to be disabled.
disableEndTime	The time when the disabled credential has to be enabled.
numberOfFailedAuthAttempts	The total number of failed authentication attempts for the user.
lastSuccessAuthAttemptTime	The time when the last authentication attempt succeeded.
lastFailedAuthAttemptTime	The time when the last authentication attempt failed.
profileName	The profile name with which the credential was created.
profileVersion	The version number of the profile.
notes	The custom attributes that are set for the credential.
transactionDetails	Contains the following transaction details: ■ message A string that defines the status of the operation. ■ reasoncode Unique code that is sent by AuthMinder Server if the operation fails. ■ responseCode Unique code that is sent by AuthMinder Server if the operation fails. ■ transactionID Unique identifier of the transaction. ■ additionalOutput The output for the additionalInput that was passed to AuthMinder Server.

Questions and Answers Authentication

The *Question and Answer* (*QnA*) authentication mechanism can either be used as a secondary authentication method for Step 1: ArcotID PKI Download **(see page 254),** or Forgot Your Password (FYP) authentication, or can be used as an independent authentication type.

In this mechanism, the user can either set their own set of questions and answers during the QnA creation stage, or your application can choose to ask pre-defined questions to the user. The maximum number of questions to be set, the number of questions to be asked to the user, and the minimum correct answers to be collected during authentication are all configurable parameters and can be set by using the Administration Console.

This section walks you through the following topics for QnA authentication:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting Response Messages

Preparing the Request Message

For QnA authentication, you must prepare the following request messages:

- Fetching Questions
- Verifying Answers

Fetching Questions

The GetQuestionsRequestMessage is used to fetch the questions from AuthMinder Server. The following table lists the elements of this message:

Element	**Mandatory**	**Description**
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
userName	Yes	The unique identifier of the user.
orgName	No	The organization to which the user belongs to.

Element	Mandatory	Description
fetchAnswers	Yes	The flag, which indicates whether to fetch the answers. Following are the supported values: ■ 0: Indicates that the answers will not be fetched. Use this option, if you want AuthMinder Server to verify the answers. ■ 1: Indicates that the answers must be fetched. You *must* enable this option if QnA authentication is performed using the caller verification mode. This mode enables you to collect the answers from the user, verify the answers, and then send the verification result to AuthMinder Server.
additionalInput /pairs	No	AuthMinder's additionalInput element enables you to set additional inputs if you want to augment AuthMinder's authentication capability by specifying additional information. In such cases, you need to set the extra information in *name-value* pairs. ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information.

Verifying Answers

The VerifyAnswersRequestMessage is used to verify the answers provided by the user. The following table lists the elements of this message:

Element	Mandatory	Description
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
orgName	No	The organization name to which the user belongs to.

Element	Mandatory	Description
qna	Yes	Contains the following information: ■ questionID The unique identifier of the question asked to the user. ■ answer The answer provided by the user. ■ verificationStatus Indicates whether the answer provided is correct or wrong. **Note:** The verificationStatus element is valid *only* if caller verification is enabled.
tokenType	No	The type of authentication token that is expected from AuthMinder Server after successful authentication. See "Verifying the Authentication Tokens" (see page 301) for more information.
additionalInput /pairs	No	AuthMinder's additionalInput element enables you to set additional inputs if you want to augment AuthMinder's authentication capability by specifying additional information. In such cases, you need to set the extra information in *name-value* pairs. ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information. **Note:** The additionalInput element is available at the end of the request message. You can add more than one of these elements.

Invoking the Web Service

The following procedure outlines the QnA authentication steps:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the GetQuestions operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. Use GetQuestionsRequestMessage and construct the input message.

   Note: In the request message, you *must* set the fetchAnswers element, if you want to enable caller verification mode.
3. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.
4. Invoke the GetQuestions operation of the ArcotWebFortAuthSvc service to retrieve the user's questions and answers from AuthMinder Server.

   This operation returns GetQuestionsResponseMessage, which includes the questions to be asked, answers for each question, transaction ID, message, response code, and reason code.
5. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the VerifyAnswers operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
6. Use VerifyAnswersRequestMessage and construct the input message.
7. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.
8. Invoke the VerifyAnswers operation of the ArcotWebFortAuthSvc service to verify the answers provided by the user. Optionally, you can also specify the token type that must be returned to the user *after* successful authentication by using the tokenType element.

   This operation returns VerifyAnswersResponseMessage, which provides the transaction details, credential details, and token information.

Interpreting Response Messages

Following are the response messages that are returned as part of QnA authentication:

- Fetch Questions Response Message
- Verify Answer Response Message

Fetch Questions Response Message

For successful transactions, the response message, GetQuestionsResponseMessage returns the elements explained in the following table. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	Description
qna	Contains the following question and answer details: ■ questionID The unique identifier of the question asked to the user. ■ question The question that has to be asked to the user. ■ answer The answer for the corresponding question to be asked.
transactionDetails	Contains the following transaction details: ■ message A string that defines the status of the operation. ■ reasoncode Unique code that is sent by AuthMinder Server if the operation fails. ■ responseCode Unique code that is sent by AuthMinder Server if the operation fails. ■ transactionID Unique identifier of the transaction. ■ additionalOutput The output for the additionalInput that was passed to AuthMinder Server.

Verify Answer Response Message

For successful transactions, the response message, VerifyAnswersResponseMessage returns the elements explained in . These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix. "Error Codes" (see page 353) for more information on the SOAP error messages.

Password Authentication

The authentication Web service provides the VerifyPassword interface to perform the traditional *Password* authentication. In this authentication mechanism, the user specifies the user name and the corresponding password for authentication. The password entered by the user is then verified.

AuthMinder supports the following types of Password authentication:

- Complete Password Authentication (see page 269)
- Partial Password Authentication (see page 272)

Complete Password Authentication

In this method, the security application prompts the user for complete password. If the password entered by the user is correct, then the authentication is considered successful.

The following topics for performing the complete password authentication are covered in this section:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The VerifyPasswordRequestMessage is used to verify the password provided by the users. The following table lists the elements of this message.

Element	Mandatory	Description
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
userName	Yes	The unique identifier of the user.
orgName	No	The organization name to which the user belongs to.
password	Yes	The password provided by the user.

Element	**Mandatory**	**Description**
tokenType	No	The type of authentication token that is expected from AuthMinder Server after successful authentication. See "Verifying the Authentication Tokens" (see page 301) for more information.
challengeID	No	The unique identifier of the challenge returned by AuthMinder Server. **Note:** The challengeID is *not* required for complete password verification.
additionalInput /pairs	No	AuthMinder's additionalInput element enables you to set additional inputs if you want to augment AuthMinder's authentication capability by specifying additional information. In such cases, you need to set the extra information in *name-value* pairs. ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information.

Invoking the Web Service

To perform regular password authentication:

1. Implement the logic to collect the user's password.
2. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the VerifyPassword operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
3. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.
4. Use VerifyPasswordRequestMessage and construct the input message.
5. Invoke the VerifyPassword operation of the ArcotWebFortAuthSvc service to verify the password provided by the user. Optionally, you can also specify the token type that must be returned to the user *after* successful authentication by using the tokenType element.

   This operation returns VerifyPasswordResponseMessage, which provides the transaction details, credential details, and token information.

Interpreting the Response Message

For successful transactions, the response message, VerifyPasswordResponseMessage returns the elements explained in Verify Signed Challenge Response Message in Step 2: ArcotID PKI Authentication (see page 258). These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Partial Password Authentication

AuthMinder supports partial password authentication, if you enable this feature, then the user will be challenged to enter the characters in different positions of the password. For example, if the password is *casablanca!*, then the user can be asked to enter the characters in positions 1, 3, and 8, which would be *csn*.

The following topics for performing partial password authentication are covered in this section:

- Preparing Request Messages
- Invoking the Web Service
- Interpreting Response Messages

Preparing Request Messages

For partial password authentication, you must prepare the following request messages:

- Fetching Challenge
- Verifying Password

Fetching Challenge

The GetPasswordChallengeRequestMessage is used to fetch the password challenge for the user from AuthMinder Server. The following table lists the elements of this message.

Element	**Mandatory**	**Description**
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
userName	Yes	The unique identifier of the user.
orgName	No	The organization name to which the user belongs to.

Element	Mandatory	Description
additionalInput /pairs	No	AuthMinder's additionalInput element enables you to set additional inputs if you want to augment AuthMinder's authentication capability by specifying additional information. In such cases, you need to set the extra information in *name-value* pairs. ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information.

Verifying Password

The VerifyPasswordRequestMessage is used to verify the password provided by the users. The following table lists the elements of this message.

Element	Mandatory	Description
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
userName	Yes	The unique identifier of the user.
orgName	No	The organization name to which the user belongs to.
password	Yes	The password provided by the user.
tokenType	No	The type of authentication token that is expected from AuthMinder Server after successful authentication. See "Verifying the Authentication Tokens" (see page 301) for more information.

Element	Mandatory	Description
challengeID	No	The unique identifier of the challenge returned by AuthMinder Server. **Note:** The challengeID is required for complete partial password verification.
additionalInput /pairs	No	AuthMinder's additionalInput element enables you to set additional inputs if you want to augment AuthMinder's authentication capability by specifying additional information. In such cases, you need to set the extra information in *name-value* pairs. ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information.

Invoking the Web Service

To perform partial password authentication:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the GetPasswordChallenge operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.
3. Use GetPasswordChallengeRequestMessage and construct the input message.
4. Invoke the GetPasswordChallenge operation of the ArcotWebFortAuthSvc service to obtain the challenge from AuthMinder Server. The challenge contains the password positions that the user has to answer.
5. Implement the logic to collect the user's password.
6. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the VerifyPassword operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
7. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.
8. Use VerifyPasswordRequestMessage and construct the input message.
9. Invoke the VerifyPassword operation of the ArcotWebFortAuthSvc service to verify the password provided by the user. Optionally, you can also specify the token type that must be returned to the user *after* successful authentication by using the tokenType element.

   This operation returns VerifyPasswordResponseMessage, which provides the transaction details, credential details, and token information.

Interpreting Response Messages

Following are the response messages that are returned as part of QnA authentication:

- Fetch Password Challenge Response Message
- Verify Password Response Message

Fetch Password Challenge Response Message

For successful transactions, the response message, GetPasswordChallengeResponseMessage returns the elements explained in the following table. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	Description
positions	The password positions for which the user has to provide the password characters.
challengeID	The unique identifier of the challenge returned by AuthMinder Server.
transactionDetails	Contains the following transaction details: ■ message A string that defines the status of the operation. ■ reasoncode Unique code that is sent by AuthMinder Server if the operation fails. ■ responseCode Unique code that is sent by AuthMinder Server if the operation fails. ■ transactionID Unique identifier of the transaction. ■ additionalOutput The output for the additionalInput that was passed to AuthMinder Server.

Verify Password Response Message

For successful transactions, the response message, VerifyPasswordResponseMessage returns the elements explained in Verify Signed Challenge Response Message in Step 2: ArcotID PKI Authentication (see page 258). These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

One-Time Password Authentication

One-Time Password (*OTP*) is a numeric or an alpha-numeric string that is generated by AuthMinder Server. AuthMinder supports OTPs that can be reused pre-configured number of times. You can specify this setting by using Administration Console. The OTP lifetime depends on the duration for which it is valid and number of times it can be used.

The following topics for performing OTP authentication are covered in this section:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The VerifyOTPRequestMessage is used to verify the OTP provided by the users. The following table lists the elements of this message.

Element	**Mandatory**	**Description**
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
userName	Yes	The unique identifier of the user.
orgName	No	The organization name to which the user belongs to.
otp	Yes	The OTP provided by the user.
tokenType	No	The type of authentication token that is expected from AuthMinder Server after successful authentication. See "Verifying the Authentication Tokens" (see page 301) for more information.

Element	Mandatory	Description
additionalInput /pairs	No	AuthMinder's additionalInput element enables you to set additional inputs if you want to augment AuthMinder's authentication capability by specifying additional information. In such cases, you need to set the extra information in *name-value* pairs. ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information.

Invoking the Web Service

To perform OTP authentication:

1. Implement the logic to collect the OTP from the user.
2. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the VerifyOTP operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
3. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.
4. Use VerifyOTPRequestMessage and construct the input message.
5. Invoke the VerifyOTP operation of the ArcotWebFortAuthSvc service to verify the OTP of the user. Optionally, you can also specify the token type that must be returned to the user *after* successful authentication by using the tokenType element.

   This operation returns VerifyOTPResponseMessage, which provides the transaction details, credential details, and token information.

Interpreting the Response Message

For successful transactions, the response message, VerifyOTPResponseMessage returns the elements explained in Verify Signed Challenge Response Message in Step 2: ArcotID PKI Authentication (see page 258). These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

OATH One-Time Password Authentication

The following topics for performing OATH-based OTP authentication are covered in this section:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The VerifyOATHOTPRequestMessage is used to verify the OATH OTP provided by the users. The following table lists the elements of this message.

Element	Mandatory	Description
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
userName	Yes	The unique identifier of the user.
orgName	No	The organization name to which the user belongs to.
otp	Yes	The OATH OTP provided by the user.
tokenType	No	The type of authentication token that is expected from AuthMinder Server after successful authentication. See "Verifying the Authentication Tokens" (see page 301) for more information.

Element	Mandatory	Description
additionalInput /pairs	No	AuthMinder's additionalInput element enables you to set additional inputs if you want to augment AuthMinder's authentication capability by specifying additional information. In such cases, you need to set the extra information in name-value *pairs.* ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information.

Invoking the Web Service

To authenticate the OTPs that are OATH compliant:

1. Implement the logic to collect the OATH OTP from the user.
2. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the VerifyOATHOTP operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
3. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.
4. Use VerifyOATHOTPRequestMessage and construct the input message.
5. Invoke the VerifyOATHOTP operation of the ArcotWebFortAuthSvc service to verify the OATH OTP of the user. Optionally, you can also specify the token type that must be returned to the user *after* successful authentication by using the tokenType element.

   This operation returns VerifyOATHOTPResponseMessage, which provides the transaction details, credential details, and token information.

Interpreting the Response Message

For successful transactions, the response message, VerifyOATHOTPResponseMessage returns the elements explained in Verify Signed Challenge Response Message in Step 2: ArcotID PKI Authentication (see page 258). These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

OATH One-Time Password Synchronization

The following topics for synchronizing the OATH OTPs are covered in this section:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The SynchronizeOATHOTPRequestMessage is used to synchronize the client and server OATH OTPs. The following table lists the elements of this message.

Element	Mandatory	Description
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
userName	Yes	The unique identifier of the user.
orgName	No	The organization name to which the user belongs to.
otpList	Yes	The subsequent client OATH OTPs to which the Server OTP has to be synchronized.
tokenType	No	The type of authentication token that is expected from AuthMinder Server after successful authentication. See "Verifying the Authentication Tokens" (see page 301) for more information.

Element	Mandatory	Description
additionalInput /pairs	No	AuthMinder's additionalInput element enables you to set additional inputs if you want to augment AuthMinder's authentication capability by specifying additional information. In such cases, you need to set the extra information in name-value *pairs.* ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information.

Invoking the Web Service

To synchronize the client and server OATH OTPs:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the SynchronizeOATHOTP operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.
3. Use SynchronizeOATHOTPRequestMessage and construct the input message.
4. Invoke the SynchronizeOATHOTP operation of the ArcotWebFortAuthSvc service to synchronize the server OTP with the client OTP. Optionally, you can also specify the token type that must be returned to the user *after* successful authentication by using the tokenType element.

   This operation returns SynchronizeOATHOTPResponseMessage, which provides the transaction details, credential details, and token information.

Interpreting the Response Message

For successful transactions, the response message, SynchronizeOATHOTPResponseMessage returns the elements explained in Verify Signed Challenge Response Message in Step 2: ArcotID PKI Authentication (see page 258). These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

ArcotID OTP (ArcotID OTP-OATH) Authentication

The following topics for performing ArcotID OTP authentication are covered in this section:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The VerifyArcotOTPRequestMessage is used to verify the ArcotID OTP provided by the users. The following table lists the elements of this message.

Element	**Mandatory**	**Description**
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
userName	Yes	The unique identifier of the user.
orgName	No	The organization name to which the user belongs to.
otp	Yes	The ArcotID OTP provided by the user.
tokenType	No	The type of authentication token that is expected from AuthMinder Server after successful authentication. See "Verifying the Authentication Tokens" (see page 301) for more information.

Element	Mandatory	Description
additionalInput /pairs	No	AuthMinder's additionalInput element enables you to set additional inputs if you want to augment AuthMinder's authentication capability by specifying additional information. In such cases, you need to set the extra information in name-value *pairs.* ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information. ■ AR_WF_OTP_TXN_SIGN_DATA Specifies the transaction data that the end user enters in the Challenge field of the ArcotID OTP client to generate a passcode in the Sign mode. The maximum length of the signed data is 64 bytes. This implementation of the Transaction Signing feature conforms to the OATH Challenge-Response Algorithm (OCRA) as defined by RFC 6287.

Invoking the Web Service

To authenticate the ArcotID OTPs:

1. Implement the logic to collect the ArcotID OTP from the user.
2. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the VerifyArcotOTP operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
3. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.
4. Use VerifyArcotOTPRequestMessage and construct the input message.
5. Invoke the VerifyArcotOTP operation of the ArcotWebFortAuthSvc service verify the ArcotID OTP of the user. Optionally, you can also specify the token type that must be returned to the user *after* successful authentication by using the tokenType element.

   This operation returns VerifyArcotOTPResponseMessage, which provides the transaction details, credential details, and token information.

Interpreting the Response Message

For successful transactions, the response message, VerifyArcotOTPResponseMessage returns the elements explained in Verify Signed Challenge Response Message in Step 2: ArcotID PKI Authentication (see page 258). These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

ArcotID OTP (ArcotID OTP-OATH) Synchronization

The following topics for synchronizing the ArcotID OTPs are covered in this section:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The SynchronizeArcotOTPRequestMessage is used to synchronize the client and server ArcotID OTPs. The following table lists the elements of this message.

Element	**Mandatory**	**Description**
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
userName	Yes	The unique identifier of the user.
orgName	No	The organization name to which the user belongs to.
otpList	Yes	The subsequent client ArcotID OTPs to which the Server OTP has to be synchronized.
tokenType	No	The type of authentication token that is expected from AuthMinder Server after successful authentication. See "Verifying the Authentication Tokens" (see page 301) for more information.

Element	**Mandatory**	**Description**
additionalInput /pairs	No	AuthMinder’s additionalInput element enables you to set additional inputs if you want to augment AuthMinder’s authentication capability by specifying additional information. In such cases, you need to set the extra information in name-value *pairs.* ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information.

Invoking the Web Service

To synchronize the client and server ArcotID OTPs:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the SynchronizeArcotOTP operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.
3. Use SynchronizeArcotOTPRequestMessage and construct the input message.
4. Invoke the SynchronizeArcotOTP operation of the ArcotWebFortAuthSvc service to synchronize the server ArcotID OTP with the client ArcotID OTP. Optionally, you can also specify the token type that must be returned to the user *after* successful authentication by using the tokenType element.

   This operation returns SynchronizeArcotOTPResponseMessage, which provides the transaction details, credential details, and token information.

Interpreting the Response Message

For successful transactions, the response message, SynchronizeArcotOTPResponseMessage returns the elements explained in Verify Signed Challenge Response Message in Step 2: ArcotID PKI Authentication (see page 258). These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

EMV OTP (ArcotID OTP-EMV) Authentication

The following topics for performing EMV OTP authentication are covered in this section:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The VerifyEMVRequestMessage is used to verify the EMV OTP provided by the users. The following table lists the elements of this message.

Element	**Mandatory**	**Description**
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
userName	Yes	The unique identifier of the user.
orgName	No	The organization name to which the user belongs to.
otp	Yes	The EMV OTP provided by the user.
tokenType	No	The type of authentication token that is expected from AuthMinder Server after successful authentication. See "Verifying the Authentication Tokens" (see page 301) for more information.

Element	Mandatory	Description
additionalInput /pairs	No	AuthMinder’s additionalInput element enables you to set additional inputs if you want to augment AuthMinder’s authentication capability by specifying additional information. In such cases, you need to set the extra information in name-value *pairs.* ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information.

Invoking the Web Service

To authenticate the EMV OTPs:

1. Implement the logic to collect the EMV OTP from the user.
2. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the VerifyEMV operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
3. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.
4. Use VerifyEMVRequestMessage and construct the input message.
5. Invoke the VerifyEMV operation of the ArcotWebFortAuthSvc service verify the EMV OTP of the user. Optionally, you can also specify the token type that must be returned to the user *after* successful authentication by using the tokenType element.

   This operation returns VerifyEMVResponseMessage, which provides the transaction details, credential details, and token information.

Interpreting the Response Message

For successful transactions, the response message, VerifyEMVResponseMessage returns the elements explained in Verify Signed Challenge Response Message in Step 2: ArcotID PKI Authentication (see page 258). These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

EMV OTP (ArcotID OTP-EMV) Synchronization

The following topics for synchronizing the EMV OTPs are covered in this section:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The SynchronizeEMVRequestMessage is used to synchronize the client and server EMV OTPs. The following table lists the elements of this message.

Element	**Mandatory**	**Description**
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
userName	Yes	The unique identifier of the user.
orgName	No	The organization name to which the user belongs to.
otpList	Yes	The subsequent client EMV OTPs to which the Server OTP has to be synchronized.
tokenType	No	The type of authentication token that is expected from AuthMinder Server after successful authentication. See "Verifying the Authentication Tokens" (see page 301) for more information.

Element	Mandatory	Description
additionalInput /pairs	No	AuthMinder's additionalInput element enables you to set additional inputs if you want to augment AuthMinder's authentication capability by specifying additional information. In such cases, you need to set the extra information in name-value *pairs.* ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information.

Invoking the Web Service

To synchronize the client and server EMV OTPs:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the SynchronizeEMV operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.
3. Use SynchronizeEMVRequestMessage and construct the input message.
4. Invoke the SynchronizeEMV operation of the ArcotWebFortAuthSvc service to synchronize the server EMV OTP with the client EMV OTP. Optionally, you can also specify the token type that must be returned to the user *after* successful authentication by using the tokenType element.

   This operation returns SynchronizeEMVResponseMessage, which provides the transaction details, credential details, and token information.

Interpreting the Response Message

For successful transactions, the response message, SynchronizeEMVResponseMessage returns the elements explained in Verify Signed Challenge Response Message in Step 2: ArcotID PKI Authentication (see page 258). These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Verifying Password Type Credentials

The authentication requests that are presented to the AuthMinder Server must specify the type of credential that has to be used to process the requests. In case of RADIUS and ASSP authentication requests, the input requests do not have the provision to specify the type of credential, and by default RADIUS uses One-Time Password and ASSP uses password credential for authentication.

To support any password-based authentication mechanisms for RADIUS and ASSP, or to map any input request with an unknown credential type to a particular password-based authentication mechanism you must create the *Credential Type Resolution* configuration. You can map the input request to any of the following credentials that AuthMinder supports:

- Password
- OTP
- OATH OTP
- ArcotID OTP-OATH
- ArcotID OTP-EMV
- RADIUS OTP
- LDAP Password
- Native Token

If a particular input request uses the credential resolution configuration, then the VerifyPlain operation is invoked to process that request. Based on the configuration, the incoming user credential will be mapped to the credential that it is configured to.

Note: To use this feature, you should have configured the created credential type resolution, as discussed in chapter, "Creating Configurations" (see page 152).

This section walks you through the following topics for verifying any password type credential:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The VerifyPlainRequestMessage is used to verify any password type credentials that AuthMinder supports. The following table lists the elements of this message.

Element	Mandatory	Description

Element	**Mandatory**	**Description**
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
userName	Yes	The name of the user whose credentials have to be verified.
orgName	No	The name of the organization to which the authenticating user belongs to.
password	Yes	The mapped password type credential with which the user has to be authenticated.
tokenType	No	The type of authentication token that is returned to the user after successful authentication.
additionalInput /pairs	No	AuthMinder’s additionalInput element enables you to set additional inputs if you want to augment AuthMinder’s authentication capability by specifying additional information. In such cases, you need to set the extra information in *name-value* pairs. ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information.

Invoking the Web Service

To verify a password type credential:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the VerifyPlain operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.
3. Use VerifyPlainRequestMessage and construct the input message.
4. Invoke the VerifyPlain operation of the ArcotWebFortAuthSvc service to verify the user's credential.

   This operation returns VerifyPlainResponseMessage, which provides the credential and transaction details.

Interpreting the Response Message

For successful transactions, the response message, VerifyPlainResponseMessage returns the elements explained in Verify Signed Challenge Response Message in Step 2: ArcotID PKI Authentication (see page 258). These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Verifying the Authentication Tokens

The AuthMinder Authentication Web service provides an appropriate token to the end user after they authenticate successfully. The token is then presented to AuthMinder Server, indicating that the user is authenticated and can be provided access to the protected resources.

By using the Authentication Web service, you can specify whether the token has to be returned after authentication or not. In addition, you can also specify the type of the token that must be returned after authentication. The tokenType element specifies the return token type and supports the following types of tokens:

- **Native Tokens**

  Specify this type when CA-proprietary (or Native) token is required after successful authentication. This token can be used multiple times before it expires.

- **One-Time Tokens**

  Specify this type when one-time token is required after successful authentication. This token can be used only one time before it expires.

- **SAML Tokens**

  Secure Assertion Markup Language (*SAML*) is an open standard, which specifies the format of the authentication data exchanged between security domains. The Native, Default, and One-Time tokens issued by AuthMinder can only be interpreted by the AuthMinder Server, but the SAML tokens issued by the AuthMinder Server can be interpreted by any other authentication system. AuthMinder supports **1.1** and **2.0** versions of SAML:

  - SAML 1.1 Tokens

    Specify this type of token when you are using custom (non-AuthMinder) authentication mechanism that needs SAML 1.1 tokens after successful authentication.

  - SAML 2.0 Tokens

    Specify this type of token when you are using custom (non-AuthMinder) authentication mechanism that needs SAML 2.0 tokens after successful authentication.

- **Default Tokens**

  Specify this type of token when the default token configured at the server is to be requested after successful authentication.

AuthMinder Server can verify *only* the Native and One-Time tokens that are issued to the users. The authentication token must be verified in cases when you use the token for Single Sign-On, wherein you authenticate the user once and allow them to use multiple resources using the same authentication token.

This section walks you through the following topics for verifying authentication tokens:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The VerifyAuthTokenRequestMessage is used to verify the authentication token returned by AuthMinder Server. The following table lists the elements of this message.

Element	Mandatory	Description
clientTxnId	No	Specifies the unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.
token	No	The authentication token that is returned to the user after successful authentication.
additionalInput /pairs	No	AuthMinder's additionalInput element enables you to set additional inputs if you want to augment AuthMinder's authentication capability by specifying additional information. In such cases, you need to set the extra information in *name-value* pairs. ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information.

Invoking the Web Service

To verify if a token is valid or not:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the VerifyAuthToken operation. See chapter, "Managing Web Services Security" (see page 35) for more information on these details.
2. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.
3. Use VerifyAuthTokenRequestMessage and construct the input message.
4. Invoke the VerifyAuthToken operation of the ArcotWebFortAuthSvc service to verify the token of the user.

   This operation returns VerifyAuthTokenResponseMessage, which provides the credential and transaction details.

Interpreting the Response Message

For successful transactions, the response message, VerifyAuthTokenResponseMessage returns the elements explained in the following table. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Method	**Description**
userName	The name of the user to whom the authentication token belongs to.
orgName	The organization to which the user belongs to.

Method	Description
transactionDetails	Contains the following transaction details: ■ message A string that defines the status of the operation. ■ reasoncode Unique code that is sent by AuthMinder Server if the operation fails. ■ responseCode Unique code that is sent by AuthMinder Server if the operation fails. ■ transactionID Unique identifier of the transaction. ■ additionalOutput The output for the additionalInput that was passed to AuthMinder Server.

Fetching the PAM

Personal Assurance Message (*PAM*) is a security feature that reassures the end users that they are accessing the genuine site of your organization, and not a phished site. To fetch the user’s PAM, you need to use the GetPAM operation.

The getPAM operation in the ArcotUserRegistrySvc service is used to perform the same task. It is recommended that you use the getPAM operation in the ArcotUserRegistrySvc service instead of GetPAM in the ArcotWebFortAuthSvc service to fetch the user’s PAM. See "Fetching the Personal Assurance Message" (see page 137) for more information.

Chapter 11: Performing Bulk Operations

This chapter discusses the following AuthMinder operations that you can perform in bulk:

- Assigning Credentials to Users (see page 306)
- Uploading OATH Tokens (see page 307)
- Fetching OATH Tokens (see page 311)

To perform these operations, you need to use the ArcotWebFortBulkOperationsSvc.wsdl file.

Assigning Credentials to Users

The Web services that are discussed in , "Performing Credential Operations" (see page 199) are used to create the out-of-box credentials for a single user. Using bulk operations Web services (ArcotWebFortBulkOperationsSvc), you can create and assign credentials to users in bulk. This section walks you through the following topics for creating credentials in bulk:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The AssignCredentialsRequest message is used to set the information required to create credentials for users in bulk. The elements of this request message are same as CreateCredential operation. Refer to the tables in the "Creating Credentials (see page 206)" section for more information on the input elements. You can either pass these elements or include them in an XML file an upload it using inputXML element.

Invoking the Web Service

To assign the credentials to the users in bulk:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the AssignCredentials operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.

   This type provides the additional information that is set as a name-value pair.
3. Use AssignCredentialsRequest and construct the input message by using the details obtained in preceding steps.
4. Invoke the AssignCredentials operation of the ArcotWebFortBulkOperationsSvc service to upload the OATH tokens.

   This operation returns an instance of the AssignCredentialsResponse that includes the credential and transaction details.

Interpreting the Response Message

For successful transactions, the response message, UploadOATHTokenResponse returns the elements explained in the following table. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	**Description**
message	A string that defines the status of the operation.
reasonCode	Unique code that is sent by AuthMinder Server if the operation fails.
responseCode	Unique code that is sent by AuthMinder Server if the operation fails.
transactionID	Unique identifier of the transaction.
additionalOutput	The output for the additionalInput that was passed to AuthMinder Server.
batchID	The unique identifier that helps to identify the batch of the uploaded token.

Uploading OATH Tokens

This section walks you through the following topics for uploading the OATH tokens in bulk:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The UplodaOATHTokensRequestMessage is used to bulk upload the OATH tokens in the AuthMinder database.The following table lists the elements of this request message.

Element	**Mandatory**	**Description**
Common Elements		
clientTxId	No	The unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Element	Mandatory	Description
additionalInput/pairs	No	AuthMinder’s additionalInput element enables you to set additional inputs if you want to augment AuthMinder’s authentication capability by specifying additional information. In such cases, you need to set the extra information in *name-value* pairs. ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information.
Common Key Elements		
orgDetails/orgName	No	Indicates the name of the organization for which you want to upload the OATH tokens.
orgDetails/isGlobal	No	Indicates whether you want to apply the OATH tokens at the global level. If you choose this option, then the OATH tokens will be available to all the organizations present in the system.
remark	No	A text message to identify the OATH tokens.
key	Yes	The key that is used to generate the OTP.
OATH Token (keyContainer) Elements **Note:** You can pass the OATH token information using the following elements or include this information in an XML file and upload that file using the "inputXml" element.		
version	Yes	The version of the schema that defines the OATH token information. The supported value for this element is 1.0.
EncryptionMethod/ algorithm	Yes	The encryption method that is used to encrypt the sensitive information. For example, the Secret element.

Element	Mandatory	Description
EncryptionMethod/IV	Yes	Base64-encoded value of the Initialization Vector that is used in the encryption scheme. This is required only for "AES128-CBC" among other supported algorithms.
TokenInfo/Manufacturer	No	The manufacturer information of the OATH token.
TokenInfo/SerialNo	No	The unique serial number of the OATH token.
TokenInfo/Model	No	The unique model number that provides information about the make of the OATH token.
TokenInfo/AdditionalInfo	No	Extra information that you want to set for the OATH tokens. This information is set in name-value pairs. ■ Name Indicates the name with which you want to create the key pair. ■ Value Indicates the corresponding value for the name.
Key/KeyAlgorithm	Yes	The algorithm that is used to generate the OTP. The supported values are: ■ HOTP: Indicates that the event-based OTPs are supported. ■ TOTP: Indicates that the time-based OTPs are supported.
Key/KeyId	Yes	The unique identifier of the token.
Key/OTPFormat/Length	Yes	Indicates the character length of the OTPs that are generated using the OATH token.
Key/Secret	Yes	The shared secret that is used to generate OTPs.
Key/Counter	No	The counter that is used to generate OTPs in case of the HOTPs. This counter defines the number of times the user can use their OTP.
Key/Time	No	The time (in seconds) to start counting time steps in case of TOTPs.
Key/TimeInterval	No	The time-step window that is used to generate TOTPs. OTPs generated within a window are same. This value is in seconds.
XML Information		

Element	Mandatory	Description
inputXml	No	The XML file that defines the key container for One Time Passwords that have to be issued by AuthMinder Server.

Invoking the Web Service

To upload OATH tokens for users in bulk:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the UploadOATHTokens operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.

   This type provides the additional information that is set as a name-value pair.
3. Use UploadOATHTokensRequestMessage and construct the input message by using the details obtained in preceding steps.
4. Invoke the UploadOATHTokens operation of the ArcotWebFortBulkOperationsSvc service to upload the OATH tokens.

   This operation returns an instance of the UploadOATHTokensResponseMessage that includes the transaction details and batch identifier.

Interpreting the Response Message

For successful transactions, the response message, UploadOATHTokenResponse returns the elements explained in the following table. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	Description
message	A string that defines the status of the operation.
reasonCode	Unique code that is sent by AuthMinder Server if the operation fails.
responseCode	Unique code that is sent by AuthMinder Server if the operation fails.
transactionID	Unique identifier of the transaction.
additionalOutput	The output for the additionalInput that was passed to AuthMinder Server.

Element	Description
batchID	The unique identifier that helps to identify the batch of the uploaded token.

Fetching OATH Tokens

This section walks you through the following topics for fetching the OATH tokens that are uploaded for an organization:

- Preparing the Request Message
- Invoking the Web Service
- Interpreting the Response Message

Preparing the Request Message

The FetchOATHTokensRequestMessage is used to fetch the OATH tokens from the AuthMinder database.The following table lists the elements of this request message.

Element	Mandatory	Description
Common Elements		
clientTxId	No	The unique transaction identifier that the calling application can include. This identifier helps in tracking the related transactions.

Element	Mandatory	Description
additionalInput /pairs	No	AuthMinder's additionalInput element enables you to set additional inputs if you want to augment AuthMinder's authentication capability by specifying additional information. In such cases, you need to set the extra information in *name-value* pairs. ■ name (The name with which you want to create the key pair.) ■ value (The corresponding value for name.) **Note:** You can add more than one of these elements. Some of the pre-defined additional input parameters include: ■ AR_WF_LOCALE_ID Specifies the locale that AuthMinder will use while returning the messages back to your calling application. ■ AR_WF_CALLER_ID This is useful in tracking transactions. You can use session ID or client transaction ID (clientTxnId) for specifying this information.
Organization Detail (orgDetails) Elements		
fetchGlobal	No	Indicates whether you want to fetch the OATH tokens that are assigned at the global level. Following are the supported values: ■ yes: The OATH tokens that are uploaded for all organizations are fetched. ■ no: The OATH tokens that are assigned to the organizations listed in the orgList element are fetched.
Or		
orgList	No	Indicates the name of the organization for which you want to upload the OATH tokens.
Token Elements		
tokenID	No	The unique identifier of the OATH token.
batchID	No	The identifier that denotes the batch in which the OATH token is manufactured.
Search Filter (tokenStatus) Elements		
tokenStatusFilter/free	No	The filter to fetch the tokens that are free and not yet assigned to the users.

Element	Mandatory	Description
tokenStatusFilter/assigned	No	The filter to fetch the tokens that are assigned to the users.
tokenStatusFilter/abandoned	No	The filter to fetch the tokens that are no longer used.
tokenStatusFilter/failed	No	The filter that is used to fetch the tokens that failed during upload. Token upload might fail in the following cases: ■ If the seed decryption operation fails. ■ If the token has already been assigned to the user.

Invoking the Web Service

To fetch the OATH tokens assigned for users of an organization:

1. (**Optional**) Include the authentication and authorization details in the SOAP header or in the additionalInput element of the FetchOATHTokens operation. See chapter, "Managing Web Services Security" (see page 35) for more information on the header elements.
2. (**Optional**) If you are implementing a plug-in, then invoke the additionalInput element type to fill the additional input.

   This type provides the additional information that is set as a name-value pair.
3. Use FetchOATHTokensRequestMessage and construct the input message by using the details obtained in preceding steps.
4. Invoke the FetchOATHTokens operation of the ArcotWebFortBulkOperationsSvc service to upload the OATH tokens.

   This operation returns an instance of the FetchOATHTokensResponseMessage that includes the credential and transaction details.

Interpreting the Response Message

For successful transactions, the response message, FetchOATHTokenResponse returns the elements explained in the following table and the token information that is uploaded. These elements are included in the SOAP body. If there are any errors, then the Fault response is included in the SOAP body. See appendix, "Error Codes" (see page 353) for more information on the SOAP error messages.

Element	Description
message	A string that defines the status of the operation.
reasonCode	Unique code that is sent by AuthMinder Server if the operation fails.

Element	Description
responseCode	Unique code that is sent by AuthMinder Server if the operation fails.
transactionID	Unique identifier of the transaction.
additionalOutput	The output for the additionalInput that was passed to AuthMinder Server.
batchID	The unique identifier that helps to identify the batch of the uploaded token.

Appendix A: Input Data Validations

To ensure that the system does not process invalid data, to enforce business rules, and to ensure that user input is compatible with internal structures and schemas, AuthMinder Server validates the data that it receives from the Web services. These validations can be grouped as:

Note: Attribute length mentioned in the following table corresponds to the character length.

AuthMinder Validation Checks

The following table lists the validation checks that are performed by AuthMinder.

Attribute	Parameter Name	Validation Criteria
Protocol Status	PROTOCL_STATUS	Checks for the following values: ■ PROTOCOL_STATUS_ACTIVE ■ PROTOCOL_STATUS_DISABLED
Port Number	PORT_NUMBER	Length is between 1 and 65535 characters.
Port Type	PORT_TYPE	■ Is non-empty ■ Checks for the following values: ■ TCP ■ SSL ■ UDP
Client Root ID	CLIENT_ROOT_ID	Checks with a set of client root IDs
Server Certificate chain encoding	SERVER_CERT_CHAIN_ENCODING	■ Server certificate chain encoding is non-empty. ■ Checks for the PEM format.
Server Certificate Chain	SERVER_CERT_CHAIN	Server certificate chain is valid.
Client Certificate Chain	CLIENT_CERT_CHAIN	Client certificate chain is valid.
Client Root CA Certificate	CLIENT_ROOT_CA_CERT	Client root CA certificate is valid.
Server Root CA Certificate	SERVER_ROOT_CA_CERT	Server root CA certificate is valid.

Attribute	Parameter Name	Validation Criteria
Client Root CA Certificates Count	CLIENT_ROOT_CA_CERT	Checks the count of CA certificate is non-zero.
Client Root ID	CLIENT_ROOT_ID	Checks with a set of client root IDs.
Server Certificate Chain Encoding	SERVER_CERT_CHAIN_ENCODING	■ Server certificate chain encoding is non-empty. ■ Checks for the PEM format.
Server Certificate Chain	SERVER_CERT_CHAIN	Server certificate chain is valid.
Client Certificate Chain	CLIENT_CERT_CHAIN	Client certificate chain is valid.
Client Root CA Certificate	CLIENT_ROOT_CA_CERT	Client root CA certificate is valid.
Server Root CA Certificate	SERVER_ROOT_CA_CERT	Server root CA certificate is valid.
Client Root CA Certificate count	CLIENT_ROOT_CA_CERT	Checks the count of CA certificates is non-zero.
Server Private Key Encoding	SERVER_PRIVATE_KEY_ENCODING	■ Server private key encoding is non-empty. ■ Checks for the PEM format.
Locale Name	LOCALE_NAME	■ Locale name is non-empty. ■ Checks locale name with the ISO set of locales.
Client Root CA Path	CLIENT_ROOT_CA_PATH	Client root CA path is non-empty.
Server ID	SERVER_ID	■ Port number > 1. ■ Checks with a set of server identifiers.
Client Root CA Certificate Encoding	CLIENT_ROOT_CA_CERT_ENCODING	■ Client root CA certificate encoding is non-empty. ■ Checks for the PEM format.
Certificate Common Name	CERT_COMMON_NAME	■ Certificate common name is non-empty. ■ Certificate common name length is between 1 and 256. ■ Does not contain invalid characters (ASCII 0-31).

Attribute	**Parameter Name**	**Validation Criteria**
Certificate Country Name	COUNTRY_NAME	■ Certificate country name is non-empty. ■ Certificate country name length is between 1 and 64. ■ Does not contain invalid characters (ASCII 0-31).
Certificate Organization Name	ORG_NAME	■ Certificate organization name is non-empty. ■ Certificate organization name length is between 1 and 64. ■ Does not contain invalid characters (ASCII 0-31).
Certificate Organization Unit Name	ORG_UNIT_NAME	■ Certificate organization unit name is non-empty. ■ Certificate organization unit name length is between 1 and 64. ■ Does not contain invalid characters (ASCII 0-31).
Certificate State Name	STATE_NAME	■ Certificate state name is non-empty. ■ Certificate state name length is between 1 and 64. ■ Does not contain invalid characters (ASCII 0-31).
Certificate Locality Name	LOCALITY_NAME	■ Certificate locality name is non-empty. ■ Certificate locality name length is between 1 and 64. ■ Does not contain invalid characters (ASCII 0-31).
Certificate Start Date	START_TIME	Checks for valid date format.
Certificate End Date	END_TIME	Checks for valid date format.
PKI Certificate	PKI_CERTIFICATE	PKI certificate is valid.
PKI Key	PKI_KEY	PKI key is valid.

Attribute	**Parameter Name**	**Validation Criteria**
Certificate Chain and Key Pair	PRIVATE_KEY_PAIR	Certificate chain and key pair are valid.
PKCS12 Certificate Chain	PKCS12_CERT_CHAIN_KEY	PKCS12 certificate chain is valid.
PKCS7 Certificate Chain	PKCS12_CERT_CHAIN_KEY	PKCS7 certificate chain is valid.
User ID	USER_ID	Minimum value of user ID must be greater than 1.
Group ID	GROUP_ID	Minimum value of group ID must be greater than 1.
Create Time	CREATE_TIME	Checks for valid date format.
Last Modified Time	LAST_MODIFIED_TIME	Checks for valid date format.
Start and End Date	START_END_DATES	Start date < End date.
User Attribute Name	USER_ATTR_NAME	User Attribute Name is non-empty.
WebFort organization name (checks for the organization name is '\n', else go for the validation)	ORG_NAME	■ Organization name is non-empty. ■ Organization name length is between 1 and 64. ■ Does not contain invalid characters (ASCII 0-31).
User Existence Check	USER_EXISTENCE_CHECK	Value of user existence check is 0 or 1.
User Active Check	USER_ACTIVE_CHECK	Value of user active check is 0 or 1.
Kerberos User Name	KERBEROS_USER_NAME	■ Kerberos user name is non-empty. ■ Kerberos user name length is between 1 and 64. ■ Does not contain invalid characters (ASCII 0-31).
Kerberos Domain Name	KERBEROS_DOMAIN_NAME	■ Kerberos domain name is non-empty. ■ Kerberos domain name length is between 1 and 64. ■ Does not contain invalid characters (ASCII 0-31).

Attribute	Parameter Name	Validation Criteria
Kerberos Password	KERBEROS_PASSWORD	■ Kerberos password is non-empty. ■ Kerberos password length is between 1 and 64. ■ Does not contain invalid characters (ASCII 0-31).
Authentication User Password	AUTH_USER_PASSWORD	■ User password is non-empty. ■ User password is between 1 and 64. ■ Checks user password against to a set of strings. ■ Does not contain invalid characters (ASCII 0-31).
Password Maximum Length	PWD_MAX_LENGTH	■ Minimum value of password maximum length must be greater than 4. ■ Maximum value of password maximum length must be less than 64.
Password Minimum Length	PWD_MIN_LENGTH	■ Minimum value of password minimum length must be greater than 4. ■ Maximum value of password minimum length must be less than 64.
Password Minimum Special Character Length	PWD_SPECIAL_CHAR_MIN_LENGTH	■ Minimum value of password special character length must be greater than 0. ■ Maximum value of password special character minimum length must be less than 64.
Password Minimum Alphabetic Character Length	PWD_ALPHA_CHAR_MIN_LENGTH	■ Minimum value of password alphabetic character length must be greater than 0. ■ Maximum value of password alphabetic character length must be less than 64.

Attribute	Parameter Name	Validation Criteria
Password Minimum Numeric Character Length	PWD_NUMERIC_CHAR_MIN_LENGTH	■ Minimum value of password numeric character length must be greater than 0. ■ Maximum value of password numeric character length must be less than 64.
Password Strength Configuration	PASSWORD_STRENGTH	Password strength attribute length must be less than the password length.
Question	AUTH_QUESTIONS	■ Question is non-empty. ■ Question length is between 1 and 64. ■ Does not contain invalid characters (ASCII 0-31).
Answer	AUTH_ANSWERS	■ Answer is non-empty. ■ Answer length is between 1 and 64. ■ Does not contain invalid characters (ASCII 0-31).
Number of Questions	NUM_OF_QNA	■ Number of questions must be greater than the minimum number of questions. ■ Number of questions must be lesser than the maximum number of questions.
Number of Questions to Ask	QNA_NUM_QUESTION_TO_ASK	■ Minimum questions to ask must be greater than 1. ■ Maximum questions to ask must be lesser than 10.
Minimum Number of Correct Answers Required	QNA_MIN_ANS_REQUIRED	■ Minimum correct answers must be greater than 1. ■ Minimum correct answers must be less than 10.
QnA Maximum Questions	MAX_QUESTIONS	■ Minimum value of maximum questions must be greater than 1. ■ Maximum value of maximum questions must be less than 10.
QnA Minimum Questions	MIN_QUESTIONS	■ Minimum value of minimum questions must be greater than 2. ■ Maximum value of minimum questions must be less than 10.

Attribute	Parameter Name	Validation Criteria
QnA Challenge Timeout in Seconds	QNA_CHALLENGE_TIMEOUT_SECS	QnA challenge timeout in seconds must be between 1 and 7200.
Plain Key Type	PLAIN_KEY_TYPE	■ Plain key type is non-empty. ■ Checks for the RSA value.
Arcot Key Type	ARCOT_KEY_TYPE	■ Plain key type is non-empty. ■ Checks for the RSA value.
Plain Key Length	PLAIN_KEY_LENGTH	Plain key length value must be between 512 and 4096.
Arcot Key Length	ARCOT_KEY_LENGTH	Arcot key length is between 512 and 4096.
ArcotID Challenge Timeout in Seconds	ARCOTID_CHALLENGE_TIMEOUT_SECS	The ArcotID PKI challenge timeout in seconds is between 1 and 7200.
ArcotID Unsigned Attribute Key Check	AID_UNSIGNED_ATTRIB_KEY	Unsigned attribute key is either USERID or ORG.
Warning Period in Days	WARNING_PERIOD_DAYS	Warning period in days is greater than 0.
Grace Period in Days	GRACE_PERIOD_DAYS	Grace period in days is greater than 0.
Auto Unlock Period in Hours	AUTO_UNLOCK_PERIOD_HOURS	Auto-unlock period in hours is greater than 0.
Authentication OTT Token	AUTH_OTT_TOKEN	■ OTT token is non-empty. ■ OTT token length is between 4 and 64.
OTT Length	OTT_LENGTH	Value of OTT length is between 5 and 240.
OTT Timeout in Seconds	OTT_TIMEOUT	Value of OTT timeout in seconds is between 1 and 172800.
OTP Length	OTP_LENGTH	Value of OTP length is between 4 and 64.
OTP Type	OTP_TYPE	Checks for numeric and alphanumeric values.
OTP Multiple Usage Count	OTP_MULTIPLE_USAGE_COUNT	Multiple usage count of OTP is between 1 and 99999.

Attribute	**Parameter Name**	**Validation Criteria**
Global Authentication Token Timeout in Seconds	GLOBAL_AUTH_TOKEN_TIMEOUT_SECS	Global authentication token timeout in seconds is between 1 and 172800.
Maximum Strikes	MAX_STRIKES	Maximum strike count is between 1 and 100.
Transaction Algorithm ID	TRANSALGO_ID	Checks for the following values: ■ NATIVE_PLAIN_CS ■ NATIVE_PLAIN_CI ■ NATIVE_SHA1_CS ■ NATIVE_SHA1_CI
Organization Credential Configuration Name	ORG_CRED_CONFIG_NAME	Organization credential configuration name is non-empty.
ArcotID Credential Configuration Name	ARCOTID_CRED_CONFIG_NAME	■ ArcotID PKI credential configuration name is non-empty. ■ Checks ArcotID PKI credential configuration name with a set of strings. ■ ArcotID PKI credential configuration name length is between 1 and 64. ■ Does not contain invalid characters (ASCII 0-31).
OTP Credential Configuration Name	OTP_CRED_CONFIG_NAME	■ OTP credential configuration name is non-empty. ■ Checks OTP credential configuration name against to a set of strings. ■ OTP credential configuration name length is between 1 and 64 ■ Does not contain invalid characters (ASCII 0-31).

Attribute	Parameter Name	Validation Criteria
QnA Credential Configuration Name	QNA_CRED_CONFIG_NAME	■ QnA credential configuration name is non-empty. ■ Checks QnA credential configuration name with a set of strings. ■ QnA credential configuration name length is between 1 and 64. ■ Does not contain invalid characters (ASCII 0-31).
Password Credential Configuration Name	UP_CRED_CONFIG_NAME	■ Password credential configuration name is non-empty. ■ Checks Password credential configuration name with a set of strings. ■ Password credential configuration name length is between 1 and 64 ■ Does not contain invalid characters (ASCII 0-31).
ArcotID Authentication Policy Name	ARCOTID_AUTH_POLICY_NAME	■ ArcotID PKI authentication policy name is non-empty. ■ Checks ArcotID PKI authentication policy name with a set of strings. ■ ArcotID PKI authentication policy name length is between 1 and 64 ■ Does not contain invalid characters (ASCII 0-31).
OTP Authentication Policy Name	OTP_AUTH_POLICY_NAME	■ OTP authentication policy name is non-empty. ■ Checks OTP authentication policy name with a set of strings. ■ OTP authentication policy name length is between 1 and 64. ■ Does not contain invalid characters (ASCII 0-31).

Attribute	Parameter Name	Validation Criteria
QnA Authentication Policy Name	QNA_AUTH_POLICY_NAME	■ QnA authentication policy name is non-empty. ■ Checks QnA authentication policy name with a set of strings. ■ QnA authentication policy name length is between 1 and 64. ■ Does not contain invalid characters (ASCII 0-31).
Password Authentication Policy Name	PASSWORD_AUTH_POLICY_NAME	■ Password authentication policy name is non-empty. ■ Checks Password authentication policy name with a set of strings. ■ Password authentication policy name length is between 1 and 64. ■ Does not contain invalid characters (ASCII 0-31).
General Authentication Policy Name	GENERAL_AUTH_POLICY_NAME	■ General authentication policy name is non-empty. ■ Checks General authentication policy name with a set of strings. ■ General authentication policy name length is between 1 and 64. ■ Does not contain invalid characters (ASCII 0-31).
RADIUS Authentication Policy Name	RADIUS_AUTH_POLICY_NAME	■ RADIUS authentication policy name is non-empty. ■ Checks RADIUS authentication policy name with a set of strings. ■ RADIUS authentication policy name length is between 1 and 64 ■ Does not contain invalid characters (ASCII 0-31).

Attribute	Parameter Name	Validation Criteria
Kerberos Authentication Policy Name	KERBEROS_AUTH_POLICY_NAME	■ Kerberos authentication policy name is non-empty. ■ Checks Kerberos authentication policy name with a set of strings. ■ Kerberos authentication policy name length is between 1 and 64. ■ Does not contain invalid characters (ASCII 0-31).
Mechanism Name	MECHANISM_NAME	■ Mechanism name is non-empty. ■ Does not contain invalid characters (ASCII 0-31). ■ Checks mechanism name with a set of strings.
Mechanism Status	MECHANISM_STATUS	Checks for the following values: ■ MECHANISM_STATUS_ENABLE ■ MECHANISM_STATUS_DISABLED
Radius Client IP Address	RADIUS_CLIENT_IP	■ Radius client IP address is non-empty. ■ Radius client IP address length is between 7 and15. ■ Does the following checks: ■ It should contain integers and '.' ■ It should contain three dots
Radius Client Shared Secret	RADIUS_ClIENT_SHARED_SECRET	■ Radius client shared secret is non-empty. ■ Radius client shared secret length is between 1 and 1024.
Radius Client Description	RADIUS_CLIENT_DESC	■ Radius client description length is between 0 and 256. ■ Does not contain invalid characters (ASCII 0-31).
Radius Client Authentication Type	RADIUS_CLIENT_AUTH_TYPE	■ Radius client shared secret is non-empty. ■ Checks for the following values: ■ OTT ■ INBAND

Attribute	Parameter Name	Validation Criteria
Radius Client Maximum Chunk Size	RADIUS_CLIENT_MAX_CHUNK_SIZE	RADIUS client maximum chunk size is between 50 and 200.
Radius Version	RADIUS_VERSION	Checks for the following values: ■ 1 ■ 2
Duplicate Question and Answers	DUPLICATE_QUESTION_AND_ANSWER	■ Questions are not duplicate. ■ Answers are not duplicate. ■ Question is not same as answer.
Token Type	AUTH_TOKEN_TYPE	Checks for the following values: ■ DEFAULT_TOKEN ■ NATIVE_TOKEN ■ OTP_TOKEN ■ SAML11_TOKEN ■ SAML20_TOKEN ■ NO_TOKEN
Configuration Name	CONFIG_NAME	■ Configuration name is non-empty. ■ Configuration name length is between 1 and64. ■ Does not contain invalid characters (ASCII 0-31).
Pin	PIN	■ Pin is non-empty. ■ Pin length is between 1 and 64. ■ Does not contain invalid characters (ASCII 0-31).
OTP Maximum Length	OTP_MAX_LENGTH	OTP maximum length is between 4 and 64.
OTP Minimum Length	OTP_MIN_LENGTH	OTP minimum length is between 4 and 64.
Last Strike Time	LAST_STRIKE_TIME	Checks for valid date format.
Last Failed Time	LAST_FAILED_TIME	Checks for valid date format.
Last Succeeded Time	LAST_SUCCEEDED_TIME	Checks for valid date format.

Attribute	Parameter Name	Validation Criteria
Credential Status	CRED_STATUS	Checks for the following values: ■ ACTIVE ■ LOCKED ■ DISABLED ■ REVOKED ■ REISSUED ■ VERIFIED
Certificate Serial Number	CERT_SERIAL_NUMBE R	■ Certificate serial number is non-empty. ■ Certificate serial number length is between 1 and32. ■ Checks for the following characters: ■ 0 – 9 ■ a – f ■ A - F
Password Minimum and Maximum Length	PWD_MIN_LENGTH	Password minimum length is lesser than password maximum length.
QnA Minimum and Maximum Questions	MIN_QUESTIONS	QnA minimum questions is lesser than QnA maximum questions.
Questions and Correct Answers	QNA_NUM_QUESTIO N_TO_ASK	Number of correct answers is lesser than number of questions.
Host Name	HOST_NAME	■ Host name is non-empty. ■ Host name length is between 1 and 64 ■ Does not contain invalid characters (ASCII 0-31).
URI	URI_NAME	■ URI is non-empty. ■ URI length is between 1 and 1024. ■ Does not contain invalid characters (ASCII 0-31).
Connection Timeout	CONNECTION_TIMEO UT	Connection timeout is between 0 and 2147483647.

Attribute	**Parameter Name**	**Validation Criteria**
Read Timeout	READ_TIMEOUT	Read timeout is between 0 and 2147483647.
Idle Timeout	IDLE_TIMEOUT	Idle timeout is between 0 and 2147483647.
Minimum Connections	MIN_CONNECTIONS	Minimum connections is between 0 and 2147483647.
Maximum Connections	MAX_CONNECTIONS	Maximum connections is between 0 and 2147483647.
WebFort Event ID	WF_EVENT_ID	Checks for the set of valid events.
Instance name	INSTANCE_NAME	■ Instance name is non-empty. ■ Instance name length is between 1 and 64. ■ Does not contain invalid characters (ASCII 0-31).
Log Level	LOG_TXN_LOG_LEVEL	Minimum database connections is between 1 and 3.
Minimum DB Connections	MIN_DB_CONNECTIONS	Minimum database connections is between 1 and 128.
Maximum DB Connections	MAX_DB_CONNECTIONS	Maximum database connections is between 1 and 512.
Maximum DB Connections Against Minimum	MAX_DB_CONNECTIONS	Maximum database connections are less than minimum database connections.
Increment DB Connections	INC_DB_CONNECTIONS	■ Increment database connections must be greater than 0. ■ Increment database connections must be less than maximum database connections-minimum database connections.
ArcotID Unsigned Attribute Key (No validation on value)	AID_UNSIGNED_ATTRIB_KEY	Attributes with name USERID and ORG are not allowed because these are created by default while creating ArcotID PKI. Therefore, these values cannot be modified.
Custom Attributes	NOTES_KEY/ NOTES_VALUE/ NOTES	■ Does not contain invalid characters (ASCII 0-31). ■ Custom attribute string length must be between 0 and 1024.

Attribute	Parameter Name	Validation Criteria
SSL Trust Store Group Name	SSL_TRUST_STORE_GROUP_NAME	■ SSL trust store group name is non-empty. ■ SSL trust store group name length is between 1 and 64. ■ Does not contain invalid characters (ASCII 0-31).
Minimum Threads	MIN_THREADS	Minimum thread count is between 1 and 1024.
Maximum Threads	MAX_THREADS	Maximum thread count is between 1 and 1024.
Threads Minimum and Maximum Count	MIN_THREADS	Minimum thread count is less than maximum thread count.
Additional Input	ADDITIONAL_INPUTS_NAME	Does not contain invalid characters (ASCII 0-31).
Server Statistics Option	STATS_OPTION	Checks for the following values: ■ CONSOLIDATED ■ PER_PROTOCOL ■ DATABASE ■ UDS_CLIENT ■ MAXVAL
Numeric Instance Attribute	parameterName that is passed to the function	Checks only if the numeric instance attributes are used.
Display Name	DISPLAY_NAME	■ Display name is non-empty. ■ Display name length is between 0 and 256. ■ Does not contain invalid characters (ASCII 0-31).
Logo URL	LOGO_URL	Checks if the URL format is valid.
Password Challenge Validity	PASSWORD_CHALLENGE_TIMEOUT_SECS	Password challenge validity is between 1 and 7200.
ArcotID Card Name	AUTH_CARD_NAME	■ ArcotID PKI Card Name is non-empty. ■ ArcotID PKI Card Name length is between 1 and 8.

Attribute	Parameter Name	Validation Criteria
Duplicate Questions	DUPLICATE_QUESTIONS	Questions are not duplicate.
Duplicate Answers	DUPLICATE_ANSWERS	Answers are not duplicate.
Partial password Length	PARTIAL_PWD_LENGTH	Partial password length is between 0 and 64.
QnA Shuffle Mode	QNA_SHUFFLE_MODE	Checks for the following values: ■ RANDOM ■ ALTERNATIVE
QnA Shuffle Flag	QNA_SHUFFLE_FLAG	Checks for the following values: ■ SHUFFLE_ALWAYS ■ SHFFULE_AFTER_SUCCESS_AUTH
QnA Return Mode	QNA_RETURN_MODE	Checks for the following values: ■ STATIC ■ RANDOM
OATH One-Time Password Length	OATH_OTP_LENGTH	OATH One-Time Password length is between 4 and 32.
OATH One-Time Password Token Type	OATH_OTP_TYPE	Checks for the following values: ■ HOTP ■ TOTP
OATH One-Time Password Authentication Look Ahead Count	OATH_OTP_AUTH_LOOK_AHEAD	OATH One-Time Password Authentication look ahead count is between 0 and 99999.
OATH One-Time Password Authentication Look Back Count	OATH_OTP_AUTH_LOOK_BACK	OATH One-Time Password Authentication look back count is between 0 and 99999.
OATH One-Time Password Synchronization Look Ahead Count	OATH_OTP_RESYNC_LOOK_AHEAD	OATH One-Time Password Synchronization look ahead count is between 0 and 99999.
OATH One-Time Password Synchronization Look Back Count	OATH_OTP_RESYNC_LOOK_BACK	OATH One-Time Password Synchronization look back count is between 0 and 99999.

User Attributes Validation Checks

The following table explains the criteria that User Data Service (UDS) uses to validate the input data.

Attribute	Attribute ID	Validation Criteria
User Name	UserName	Is non-empty.
		Length is between 1 and 256 characters.
		Does not contain invalid characters (ASCII 0-31).
First Name	FirstName	Is non-empty.
		Length is between 1 and 32 characters.
		Does not contain invalid characters (ASCII 0-31).
Middle Name	MiddleName	Length is between 0 and 32 characters.
		Does not contain invalid characters (ASCII 0-31).
Last Name	LastName	Is non-empty.
		Length is between 1 and 32 characters.
		Does not contain invalid characters (ASCII 0-31).
Email	Email	Is non-empty.
		Length is between 1 and 128 characters.
		Does not contain invalid characters. All default regular expressions are allowed.
Telephone Number	TelephoneNumber	Is non-empty.
		Length is between 1 and 128 characters.
		Does not contain invalid characters (ASCII 0-31).
Personal Assurance Message	PAM	Length is between 0 and 128 characters.
		Does not contain invalid characters (ASCII 0-31).
Personal Assurance Message URL	PAM URL	Length is between 0 and 128 characters.
		Does not contain invalid characters, although alphabets, number, and + / \ \ # $ % & - _ : . are allowed.
Image	Image	Size is between 0 and 1024 KB.

Attribute	Attribute ID	Validation Criteria
		Is of one of the following formats: ■ JPEG ■ JPG ■ GIF ■ BMP ■ PNG
Account ID	AccountID	Length is between 0 and 256 characters.
		Is non-empty if the AccountType attribute is enabled.
		Does not contain invalid characters (ASCII 0-31).
Account ID Attribute1	AccountIDAttribute1	Length is between 0 and 256 characters.
		Does not contain invalid characters (ASCII 0-31).
Account ID Attribute2	AccountIDAttribute2	Length is between 0 and 256 characters.
		Does not contain invalid characters (ASCII 0-31).
Account ID Attribute3	AccountIDAttribute3	Length is between 0 and 256 characters.
		Does not contain invalid characters (ASCII 0-31).
User Custom Attributes	User Custom Attributes--	Maximum supported database column size for the field is 2000 KB. The maximum length is dependent on number of custom attributes, multi-byte character support, and encryption.
		Does not contain invalid characters (ASCII 0-31).
Organization Name	OrgName	Is non-empty.
		Length is between 0 and 64 characters.
		Does not contain invalid characters. **Note:** All keyboard characters are supported.
Display Name	DisplayName	Is non-empty.
		Length is between 0 and 128 characters.
		Does not contain invalid characters (ASCII 0-31).
Description	Description	Length is between 1 and 128 characters.

Attribute	Attribute ID	Validation Criteria
		Does not contain invalid characters (ASCII 0-31).
Account Type	AccountType	Length is between 0 and 64 characters.
		Does not contain invalid characters. **Note:** All keyboard characters are supported.
Account Type Display Name	AccountType-DisplayName	Is non-empty if the AccountType attribute is enabled.
		Length is between 0 and 128 characters.
		Does not contain invalid characters (ASCII 0-31).
Organization Custom Attributes	Org Custom Attributes	Length is between 0 and (2000 -(2 * No of custom attributes - 1)) characters.
		Does not contain invalid characters (ASCII 0-31).
Account Type Custom Attribute	Account Type Custom Attribute	Length is between 0 and (2000 -(2 * No of custom attributes - 1)) characters.
		Does not contain invalid characters (ASCII 0-31).
User Account Custom Attributes - Name	User Account Custom Attributes -Name	Length is between 0 and 64 characters.
		Does not contain invalid characters (ASCII 0-31).
User Account Custom Attributes - Value	User Account Custom Attributes - Value	Length is between 0 and 128 characters.
		Does not contain invalid characters (ASCII 0-31).
Key Label	Key Label	Is non-empty.
		Does not contain invalid characters. **Note:** All keyboard characters are supported.

Appendix B: AuthMinder Logging

To effectively manage the communication between AuthMinder Server and your application, it is necessary to get information about the activity and performance of the Server as well as any problems that have occurred.

This appendix describes the various log files supported by AuthMinder, the severity levels that you will see in these files, and the formats of these log files. It covers the following topics:

- About the Log Files (see page 336)
- Format of the AuthMinder Log Files (see page 341)
- Format of UDS and Administration Console Log Files (see page 342)
- Supported Severity Levels (see page 343)

About the Log Files

The AuthMinder log files can be categorized as:

- Installation Log File (see page 337)
- AuthMinder Server Startup Log File (see page 337)
- AuthMinder Server Log File (see page 338)
- UDS Log File (see page 339)
- Administration Console Log File (see page 340)

The parameters that control logging in these files can be configured either by using the relevant INI files (as is the case with Administration Console, UDS, and AuthMinder Server startup log files) or by using Administration Console itself (as is the case with AuthMinder log file.) The typical logging configuration options that you can change in these files include:

- **Specifying log file name and path:** AuthMinder enables you to specify the directory for writing the log files and storing the backup log files. Specifying the diagnostic logging directory allows administrators to manage system and network resources.
- **Log file size:** The maximum number of bytes the log file can contain. When the log files reach this size, a new file is created and the old file is moved to the backup directory.
- **Using log file archiving:** As AuthMinder components run and generate diagnostic messages, the size of the log files increases. If you allow the log files to keep increasing in size, then the administrator must monitor and clean up the log files manually. AuthMinder enables you to specify configuration options that limit how much log file data is collected and saved. AuthMinder lets you specify the configuration option to control the size of diagnostic logging files. This lets you determine a maximum size for the log files. When the maximum size is reached, older log information is moved to the backup file before the newer log information is saved.
- **Setting logging levels:** AuthMinder also allows you to configure logging levels. By configuring logging levels, the number of messages saved to diagnostic log files can be reduced. For example, you can set the logging level so that the system only reports and saves critical messages. See "Supported Severity Levels" (see page 343) for more information on the supported log levels.
- **Specifying time zone information:** AuthMinder enables you to either use the local time zone for time stamping the logged information or use GMT for the same.

Installation Log File

When you install AuthMinder, the installer records all the information that you supply during the installation and the actions (such as creating the directory structure and making registry entries) that it performs in the Arcot_WebFort_Install_*[assign the value for mm in your book]_<dd>_<yyyy>_<hh>_[assign the value for mm in your book]_SpectroSERVER*.log file. The information in this file is very useful in identifying the source of the problems if the AuthMinder installation did not complete successfully.

The default location of this file is:

Windows:
`<`*`install_location`*`>\`

UNIX-Based Platforms:
`<`*`install_location`*`>/`

AuthMinder Server Startup Log File

When you start AuthMinder Server, it records all start-up (or boot) actions in the arcotwebfortstartup.log file. The information in this file is very useful in identifying the source of the problems if the AuthMinder service does not start up.

The default location of this file is:

Windows:
`<`*`install_location`*`>\Arcot Systems\logs\`

UNIX-Based:
`<`*`install_location`*`>/arcot/logs/`

AuthMinder Server Log File

When you perform AuthMinder Server configurations for example, protocol configurations, profile configurations, policy configurations, and authenticate users, such configurations are written to the arcotwebfort.log file. The default location of this file is:

Windows:
`<install_location>\Arcot Systems\logs\`

UNIX-Based:
`<install_location>/arcot/logs/`

The parameters that control logging in this file can be configured by using Administration Console. To do so, you must use the instance-specific configuration sub-screen that you can access by clicking the required instance in the **Instance Management** screen.

In addition to the log file path, the maximum log file size (in bytes), backup directory, logging level, and timestamp information, you can also control whether you want to enable trace logging. See section, "Format of the AuthMinder Log Files" (see page 341) for the details of the default format used in the file.

UDS Log File

All User Data Service (UDS) information and actions are recorded in the arcotuds.log file. This information includes:

- UDS database connectivity information
- UDS database configuration information
- UDS instance information and the actions performed by this instance

The information in this file is very useful in identifying the source of the problems if Administration Console could not connect to the UDS instance. The default location of this file is:

Windows:
`<install_location>\Arcot Systems\logs\`

UNIX-Based:
`<install_location>/arcot/logs/`

The parameters that control logging in this file can be configured by using the udsserver.ini file, which is available in the conf folder in ARCOT_HOME.

In addition to the logging level, log file name and path, the maximum file size (in bytes), and archiving information, you can also control the layout of the logging pattern for UDS by specifying the appropriate values for log4j.appender.debuglog.layout.ConversionPattern. See section, "Format of UDS and Administration Console Log Files" (see page 342) for details of the default format used in the file.

Administration Console Log File

When you deploy Administration Console and subsequently start it, the details of all its actions and processed requests are recorded in the arcotadmin.log file. This information includes:

- Database connectivity information
- Database configuration information
- Instance information and the actions performed by this instance
- UDS configuration information
- Other Administration Console information specified by the Master Administrator, such as cache refresh

The information in this file is very useful in identifying the source of the problems if Administration Console does not start up. The default location of this file is:

Windows:
`<install_location>\Arcot Systems\logs\`

UNIX-Based:
`<install_location>/arcot/logs/`

The parameters that control logging in this file can be configured by using the adminserver.ini file, which is available in the conf folder in ARCOT_HOME.

In addition to the logging level, log file name and path, the maximum log file size (in bytes), and log file archiving information, you can also control the layout of the logging pattern for the console by specifying the appropriate values for log4j.appender.debuglog.layout.ConversionPattern. See section, "Format of UDS and Administration Console Log Files" (see page 342) for the details of the default format used in the file.

Format of the AuthMinder Log Files

The following table describes the format of the entries in the following AuthMinder loggers:

- arcotwebfort.log (AuthMinder Server Log File (see page 338))
- arcotwebfortstartup.log (AuthMinder Server Startup Log File (see page 337))

Column	Description
Time Stamp	The time when the entry was logged, translated to the time zone you configured. The format of logging this information is: mm/dd/yy HH:MM:SS.mis Here, mis represents milliseconds.
Log Level (LEVEL) (or Severity)	The severity level of the logged entry. See"Supported Severity Levels" (see page 343) for more information. **Note:** AuthMinder also provides trace logging, which contains the flow details. The trace logs are logged in the arcotwebfort.log file. The entries for the trace messages start with TRACE:.
Protocol Name (PROTOCOLNAME)	The protocol used for the transaction. Possible values are: ■ AUTH_NATIVE ■ ADMIN_WS ■ ASSP_WS ■ RADIUS ■ SVRMGMT_WS ■ TXN_WS In case the server is starting up, shutting down, or is in the monitoring mode, then no protocol is used and the following values are displayed, respectively: ■ STARTUP ■ SHUTDOWN ■ MONITOR
Thread ID (THREADID)	The ID of the thread that logged the entry.
Transaction ID (000TXNID)	The ID of the transaction that logged the entry.

Column	Description
Message	The message logged by the Server in the log file in the free-flowing format. **Note:** The granularity of the message depends on the **Log Level** that you set in the log file.

Format of UDS and Administration Console Log Files

The table describes the format of the entries in the following loggers:

- arcotuds.log (UDS Log File (see page 339))
- arcotadmin.log (Administration Console Log File (see page 340))

Column	Associated Pattern (In the Log File)	Description
Time Stamp	%d{yyyy-MM-dd hh:mm:ss,SSS z} :	The time when the entry was logged. This entry uses the application server time zone. The format of logging this information is: yyyy-MM-dd hh:mm:ss,SSS z Here, SSS represents milliseconds.
Thread ID	[%t] :	The ID of the thread that logged the entry.
Log Level (or Severity)	%-5p :	The severity level of the logged entry. See Supported Severity Levels (see page 343) for more information.
Logger Class	%-5c{3}(%L) :	The name of the logger that made the log request.
Message	%m%n :	The message logged by the Server in the log file in the free-flowing format. **Note:** The granularity of the message depends on the **Log Level** that you set in the log file.

Refer to the following URL for customizing the **PatternLayout** parameter in the UDS and Administration Console log files:

http://logging.apache.org/log4j/1.2/apidocs/org/apache/log4j/PatternLayout.html

Supported Severity Levels

A *log level* (or *severity level*) enables you to specify the level of detail of the information stored in the AuthMinder logs. This also enables you to control the rate at which the log file will grow.

Server Log File Security Levels

The following table describes the log levels that you see in all log files, in the *decreasing* order of severity.

Log Level		Description
0	FATAL	Use this log level for serious, non-recoverable errors that can cause the abrupt termination of the AuthMinder service.
1	WARNING	Use this log level for undesirable run-time exceptions, potentially harmful situations, and recoverable problems that are not yet FATAL.
2	INFO	Use this log level for capturing information on run-time events. In other words, this information highlights the progress of the application, which might include changes in: ■ Server state, such as start, stop, and restart. ■ Server properties. ■ State of services. ■ State of a processes on the Server.
3	DEBUG	Use this log level for logging detailed information for debugging purposes. This might include process tracing and changes in Server states.

Note: For AuthMinder Server (arcotwebfort.log) you can set the logging to any of these levels and also enable TRACE logging to capture the flow details.

Note: When you specify a log level, messages from all other levels of *higher* significance are reported as well. For example if the LogLevel is specified as 3, then messages with log levels of FATAL, WARNING, and INFO level are also captured.

Administration Console and UDS Log File Severity Levels

The following table describes the log levels that you see in Administration Console and UDS log files, in the *decreasing* order of severity.

Log Level		**Description**
0	OFF	Use this level to disable all logging.
1	FATAL	Use this log level for serious, non-recoverable errors that can cause the abrupt termination of Administration Console or UDS.
2	WARNING	Use this log level for undesirable run-time exceptions, potentially harmful situations, and recoverable problems that are not yet FATAL.
3	ERROR	Use this log level for recording error events that might still allow the application to continue running.
4	INFO	Use this log level for capturing information on run-time events. In other words, this information highlights the progress of the application, which might include changes in: ■ Server state, such as start, stop, and restart. ■ Server properties. ■ State of services. ■ State of a processes on the Server.
5	TRACE	Use this log level for capturing finer-grained informational events than DEBUG.
6	DEBUG	Use this log level for logging detailed information for debugging purposes. This might include process tracing and changes in Server states.
7	ALL	Use this log level to enable all logging.

Note: When you specify a log level, messages from all other levels of *higher* significance are reported as well. For example if the LogLevel is specified as 4, then messages with log levels of FATAL, WARNING, ERROR, and INFO are also captured.

Sample Entries for Each Log Level

The following subsections show a few sample entries (based on the Log Level) in the **WebFort log file.**

FATAL

```
07/17/09 11:49:20.404  FATAL  STARTUP  00002872  00WFMAIN - Unable to initialize the
database

07/17/09 11:49:20.405  FATAL  STARTUP  00002872  00WFMAIN - Failed to load the ini
parameters

07/17/09 11:49:20.406  FATAL  STARTUP  00002872  00WFMAIN - Cannot continue due to
setConfigData failure, SHUTTING DOWN
```

WARNING

```
07/17/09 12:50:05.848  INFO  AUTH_NATIVE  00002780  00022508 - Fail to connect to
Database prdsn for 1 time(s). DbUsername system

07/17/09 12:50:05.848  INFO  AUTH_NATIVE  00002780  00022508 - ReportError: SQL Error
State:08001, Native Error Code: FFFFFFFF, ODBC Error: [Arcot Systems][ODBC Oracle Wire
Protocol driver][Oracle]TNS-12505: TNS:listener could not resolve SID given in
connect descriptor
```

INFO

```
07/17/09 11:51:20.166  INFO  MONITOR  00000424  STATSMON - mMinConnections [4]
07/17/09 11:51:20.166  INFO  MONITOR  00000424  STATSMON - mMaxConnections [128]
07/17/09 11:51:20.166  INFO  MONITOR  00000424  STATSMON - mCurrPoolSize   [4]
07/17/09 11:51:20.166  INFO  MONITOR  00000424  STATSMON - mNumDBFailure     [0]
07/17/09 11:51:20.166  INFO  MONITOR  00000424  STATSMON - mCurrNumUsed      [0]
07/17/09 11:51:20.166  INFO  MONITOR  00000424  STATSMON - mCurrNumAvailable [4]
07/17/09 11:51:20.166  INFO  MONITOR  00000424  STATSMON - idx [0]
mNumTimesConnIdxLocked   [24]
07/17/09 11:51:20.166  INFO  MONITOR  00000424  STATSMON - idx [0]
mNumTimesConnIdxReleased [24]
07/17/09 11:51:20.166  INFO  MONITOR  00000424  STATSMON - idx [1]
mNumTimesConnIdxLocked   [24]
07/17/09 11:51:20.166  INFO  MONITOR  00000424  STATSMON - idx [1]
mNumTimesConnIdxReleased [24]
07/17/09 11:51:20.166  INFO  MONITOR  00000424  STATSMON - idx [2]
mNumTimesConnIdxLocked   [24]
07/17/09 11:51:20.166  INFO  MONITOR  00000424  STATSMON - idx [2]
mNumTimesConnIdxReleased [24]
07/17/09 11:51:20.166  INFO  MONITOR  00000424  STATSMON - idx [3]
mNumTimesConnIdxLocked   [23]
```

```
07/17/09 11:51:20.166  INFO  MONITOR  00000424  STATSMON - idx [3]
mNumTimesConnIdxReleased [23]
07/17/09 11:51:20.166  INFO  MONITOR  00000424  STATSMON - ---------- logging stats
for databse [wf-test-p] : [primary] [ACTIVE] end   ----------
```

DEBUG

```
03/25/10 15:29:30.921 DEBUG SVRMGMT_WS   00000536 00000620 -
ArDBPoolManagerImpl::getLockedDBConnection: [primary] DSN [webfort] is active. Will
get the connection from this
03/25/10 15:29:30.921 DEBUG SVRMGMT_WS   00000536 00000620 -
ArDBPoolManagerImpl::getLockedDBConnection: Returning DBPool [0112FD80]
03/25/10 15:29:30.921 DEBUG SVRMGMT_WS  00000536 00000620 - ArDBM::Number of queries
being executed [1]
03/25/10 15:29:30.921 DEBUG SVRMGMT_WS  00000536 00000620 - ArDBM::Found query string
for query-id : [SSL_TRUST_STORE_FETCH_ALL].
03/25/10 15:29:30.921 DEBUG SVRMGMT_WS   00000536 00000620 - ArDBM::Executing
Query[ArWFSSLTrustStoreQuery_FetchAll]
03/25/10 15:29:30.921 DEBUG SVRMGMT_WS   00000536 00000620 - Number of rows fetched
: 0
```

(For AuthMinder Server *Only*) Trace Logs

```
03/25/10 15:23:38.515 DEBUG SVRMGMT_WS   00004396 00000596 - TRACE: Released Cache
read lock on [01129D98]
03/25/10 15:23:38.515 DEBUG SVRMGMT_WS  00004396 00000596 - TRACE: CallTrace::Leaving
: [ArDBPoolManagerImpl::selectAnActivePool]. time : 0
03/25/10 15:23:38.515 DEBUG SVRMGMT_WS   00004396 00000596 - TRACE:
CallTrace::Entering : [ArDBPool::getLockedDBConnectionConst]
03/25/10 15:23:38.515 DEBUG SVRMGMT_WS   00004396 00000596 - TRACE:
ArDBPool::getLockedDBConnection [(primary)] : GotContext [1], [3] more connections
available
03/25/10 15:23:38.515 DEBUG SVRMGMT_WS  00004396 00000596 - TRACE: CallTrace::Leaving
: [ArDBPool::getLockedDBConnectionConst]. time : 0
```

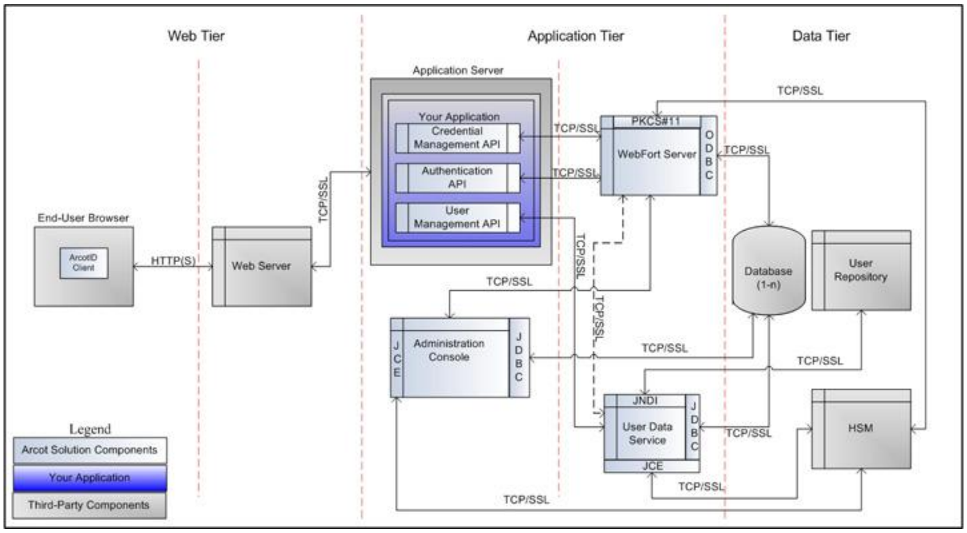
Appendix C: Enabling SSL for Web Services

This appendix provides the steps to enable SSL communication between Web services and AuthMinder Server.

Setting up SSL

To ensure integrity and confidentiality of the data being exchanged during a session, WebFort supports Secure Socket Layer (SSL) communication between Web services and WebFort Server. By default, the communication mode between all the components is through Transmission Control Protocol (TCP).

The following figure shows the communication that are supported between WebFort components:



Important! Before you enable SSL communication between Web services and WebFort Server, you must obtain a digital certificate from a trusted Certificate Authority and expose your application over an HTTPS-enabled server port.

To set up one-way SSL between Web services (Authentication and Issuance) and WebFort Server, you must first configure the Transaction Web Services protocol by using the **Protocol Management** page of Administration Console.

In case of two-way SSL, you must create the client store using the **Trusted Certificates Authorities** page, configure the client store using the **Protocol Management** (**Transaction Web Services**) page, and configure the client certificates using the **WebFort Connectivity** (**Transaction Web Services**) page of Administration Console.

The following subsections walk you through the detailed steps for configuring:

- One-Way SSL (see page 349)
- Two-Way SSL (see page 350)

Note: In this communication, your application integrated with Web services is the client and WebFort Server is the server.

One-Way SSL

Perform the following steps to set up SSL between Web services and AuthMinder Server:

1. Access Administration Console in a Web browser.
2. Log in to Administration Console as the Master Administrator (MA).
3. Activate the **Services and Server Configurations** tab in the main menu.
4. Ensure that the **WebFort** tab in the submenu is active.
5. Under the **Instance Configurations** section, click the **Protocol Management** link to display the Protocol Configuration page.
6. Select the **Server Instance** for which you want to configure the protocols.
7. In the **List of Protocols** section, click the **Transaction Web Services** protocol link

   The page to configure the protocol appears.
8. Configure the following fields:
   - Ensure that the **Protocol Status** is **Enabled**.
   - In the **Transport** field, select **SSL (1-Way)**.
   - Select **Key in HSM** if you want to store the SSL key in HSM.
   - (*Only* if you selected **Key in HSM** in the preceding step) Click the **Browse** button adjacent to the **Certificate Chain (in PEM Format)** field to select the AuthMinder root certificate.
   - Click the **Browse** button adjacent to the **P12 File Containing Key Pair** field to select the AuthMinder root certificate.
   - Enter the password for the PKCS#12 store in the **P12 File Password** field.
9. Click the **Save** button.
10. Restart the AuthMinder Server instance.

Two-Way SSL

To enable SSL communication mode between Web services and AuthMinder Server:

1. Enable the application server where your client integrated with Web services is deployed for SSL communication. Refer to your application server vendor documentation for more information on how to do this.
2. Log in to Administration Console as the MA.
3. Activate the **Services and Server Configurations** tab in the main menu.
4. Activate the **WebFort** tab in the submenu.
5. Under **Instance Configurations**, click the **Trusted Certificate Authorities** link to display the corresponding page.

   The Trusted Certificate Authorities page appears.
6. Set the following information:
   - In the **Name** field, enter the name for the SSL trust store.
   - Click the **Browse** button to select the root certificate of the application server where Web services client is deployed.
7. Click the **Save** button.
8. Under **Instance Configurations**, click the **Protocol Management** link to display the corresponding page.

   The Protocol Configuration page appears.
9. Select the **Server Instance** for which you want to configure the protocols.
10. In the **List of Protocols** section, click the **Transaction Web Services** link.

    The page to configure the protocol appears.
11. Configure the following fields:
    - Ensure that the protocol is enabled.
    - In the **Transport** field, select **SSL (2-Way)**.
    - Select **Key in HSM** if you want to store the SSL key in HSM.
    - (*Only* if you selected **Key in HSM** in the preceding step) Click the **Browse** button adjacent to the **Certificate Chain (in PEM Format)** field to select the AuthMinder root certificate.
    - Click the **Browse** button adjacent to the **P12 File Containing Key Pair** field to select the AuthMinder root certificate.
    - Enter the password for the PKCS#12 store in the **P12 File Password** field.
    - Select the **Client Store** that you created in Step 6.
12. Click the **Save** button.
13. Restart the AuthMinder Server instance.

14. Activate the **Services and Server Configurations** tab in the main menu.
15. Activate the **WebFort** tab in the submenu.
16. Under **System Configuration**, click the **WebFort Connectivity** link to display the corresponding page.

    The WebFort Connectivity page appears.
17. Set the following for the **Transaction Web Services** protocol:
    - Ensure that the **IP Address** and **Port** number of AuthMinder Server is set appropriately.
    - In the **Transport** field, select **SSL**.
    - Click the **Browse** button adjacent to the **Server CA Certificate in PEM** field to select the AuthMinder root certificate.
    - Click the **Browse** button adjacent to the **Client Certificate-Key Pair in PKCS#12** field to select the PKCS#12 file that contains the root certificate of the application server where Java SDKs are deployed.
    - Enter the PKCS#12 file password in the **Client PKCS#12 Password** field.
18. Click the **Save** button.
19. Restart the AuthMinder Server instance.
20. Verify that the AuthMinder Server is enabled for SSL communication by performing the following steps:

    a. Navigate to the following location:

    - **On Windows**:
      `<install_location>\Arcot Systems\logs`
    - **On UNIX-Based Platforms**:
      `<install_location>/arcot/logs`

    b. Open the arcotwebfortstartup.log file in a text editor.

    c. Search for the following section:

    Listing : [Successful listeners(Type-Port-FD)]

    d. In this section, you must find the following line:

    ```
    Transaction-WS.............................. :
    [SSL-9744-<Internal_listener_identifier>-[subject
    [<cert_subject>] issuer [<cert_issuer>] sn
    [<cert_serial_number>] device [<device_name>]]]
    ```

    e. Close the file.

Appendix D: Error Codes

This appendix lists the error codes that are returned by the AuthMinder 7.0 Web services. It covers the following error code types:

- User Data Service Error Codes (see page 353)
- AuthMinder Server Codes (see page 373)

User Data Service Error Codes

The following table lists the error codes and messages that returned by the Web services used for managing organizations, users, and account types.

Error Code	Error Message	Possible Cause for Failure
31201	Unable to process the database query, {0}. **Note:** This is a critical error.	**Possible Causes:** ■ Invalid input parameter was specified. ■ Error occurred during encryption. ■ Database is down. **Solution:** 1. Verify that the database information in arcotcommon.ini is correct. 2. See if there are any database-related errors in arcotadmin.log and arcotuds.log and take corrective action. 3. If the issue is not resolved, then you must contact CA Support.

Error Code	Error Message	Possible Cause for Failure
31002	UDS is not initialized. **Note:** This is a critical error, and is typically seen when UDS or Administration Console are restarted.	**Possible Causes:** ■ ARCOT_HOME is not correctly set ■ Database is down. ■ Hardware encryption initialization failed. **Solution:** 1. Verify that the database information in arcotcommon.ini is correct. 2. See if there are any database-related errors in arcotadmin.log and arcotuds.log and take corrective action. 3. If the issue is not resolved, then you must contact CA Support.
31003	Fatal error, restart UDS. This error is expected when UDS application has encountered an unexpected error.	**Possible Causes:** ■ UDS did not correctly start up. ■ Database is down. ■ Hardware encryption initialization failed. **Solution:** 1. Verify that the database information in arcotcommon.ini is correct. 2. See if there are any database-related errors in arcotadmin.log and arcotuds.log and take corrective action. 3. Restart UDS.
31006	Configuration parameter, {0} not found. This error occurs if the specified UDS configuration was not found in the ARUDSCONFIG table.	**Possible Causes:** ■ Database was manually updated. ■ Information in database tables was not correctly populated. **Solution:** 1. See if there are any database-related errors in arcotadmin.log and arcotuds.log and take corrective action. 2. If the issue is not resolved, then you must contact CA Support.

Error Code	Error Message	Possible Cause for Failure
31007	Invalid configuration parameter value, {0} for parameter name, {1}. This error occurs if the specified UDS configuration contains invalid value(s).	**Possible Causes:** ■ Database was manually updated. ■ Information in database tables was not correctly populated. **Solution:** 1. See if there are any database-related errors in arcotadmin.log and arcotuds.log and take corrective action. 2. If the issue is not resolved, then you must contact CA Support.
31008	Unknown error. This message appears if an unexpected internal error occurred.	**Possible Cause:** Unexpected internal error. **Solution:** 1. See the arcotadmin.log and arcotuds.log files and take corrective action. 2. If the issue is not resolved, then you must contact CA Support.
31009	General error: {0}. This message appears if an unexpected internal error occurred.	**Possible Cause:** Unexpected internal error. **Solution:** 1. See the arcotadmin.log and arcotuds.log files and take corrective action. 2. If the issue is not resolved, then you must contact CA Support.
35100	Error communicating with data store. **Note:** This is a critical error that occurs either when the connectivity with the database server is lost or during processing of a database query.	**Possible Causes:** ■ The database is down. ■ There was an error during encryption or decryption of data. **Solution:** 1. See the arcotadmin.log and arcotuds.log files and take corrective action. 2. If the issue is not resolved, then you must contact CA Support.

Error Code	Error Message	Possible Cause for Failure
35101	Error while loading configuration file, {0}. **Note:** This is a critical error and occurs while reading configuration files in the conf directory of ARCOT_HOME.	**Possible Cause:** The configuration files are corrupted. **Solution:** 1. Ensure that the configuration files contain the required details. 2. Retry reading from the files.
35102	Configuration file, {0} not found.	**Possible Causes:** 1. ARCOT_HOME is not set. 2. The required configuration files not found in the conf directory of ARCOT_HOME. **Solution:** 1. Verify if ARCOT_HOME points to the right location. 2. Verify if the required configuration files exist in the conf directory of ARCOT_HOME.
35103	ARCOT_HOME environment variable is not set.	**Possible Cause:** ARCOT_HOME not set. **Solution:** Set the ARCOT_HOME to point to your Arcot installation directory.
35105	Invalid input parameter.	**Possible Cause:** The input value provided for the specified parameter is not valid. **Solution:** See User Attributes Validation Checks (see page 331) for more information on valid input parameters.
35106	Missing input parameter, {0}.	**Possible Cause:** The specified input parameter is missing in the API request. **Solution:** Provide the required parameter.

Error Code	Error Message	Possible Cause for Failure
35107	Cannot update global chosen encryption set, because the organization already contains users.	**Possible Cause:** One or more organizations refer to the global encryption set, but users have already been created in these organizations. **Solution:** Global encryption set cannot be updated when there are users in the referring organizations.
35108	Error while audit logging.	**Possible Causes:** 1. Database connectivity is lost. 2. Invalid input provided for the audit logs. **Solution:** See the arcotadmin.log and arcotuds.log files and take corrective action.
35109	Field, {0} exceeded maximum length, {1}.	**Possible Cause:** The specified field exceeded the allowed length. **Solution:** Provide a value within the expected range. See User Attributes Validation Checks (see page 331) for more information on valid input parameters.
35110	Field, {0} contains invalid characters.	**Possible Causes:** The value provided for the specified field contains unsupported characters. **Solution:** Retry with valid inputs. See User Attributes Validation Checks (see page 331) for more information on valid input parameters.
31125	User, {0} not found.	**Possible Causes:** 1. The user identifier specified in the request is not valid. 2. The user does not exist in the system. 3. The user has been deleted. **Solution:** 1. Provide valid user details. 2. Also, search for deleted users to verify whether the user has been deleted.

Error Code	Error Message	Possible Cause for Failure
31126	User, {0} not unique. More than one user found.	**Possible Cause:** The specified user is not unique in the system. As a result, more than one user returned for the given UserID. **Solution:** 1. Ensure that the UserID is unique. 2. Verify if the UserID mapping attribute exists in the LDAP organization. 3. If the issue is not resolved, then you must contact CA Support.
31118	Search field, {0} not permitted.	**Possible Cause:** User search is not permitted on the specified field. For example, searching for an unmapped LDAP attribute is not allowed. **Solution:** 1. Search based on other fields. 2. Ensure that the required attributes are correctly mapped.
31127	Operation, {0} not supported. Invalid current state {1} of User, {2}.	**Possible Causes:** ■ The current operation is not supported for the given user status. ■ The status of the specified user is either INITIAL or INACTIVE. For example, authentication operations are not supported for INACTIVE users. **Solution:** Update the user status to a valid status and then perform the operation. See User Attributes Validation Checks (see page 331) for more information on valid input parameters.
31104	Operation, {0} not supported for repository.	**Possible Cause:** The current operation is not supported for the repository. For example, Write operations are not supported for LDAP repository. **Solution:** Unsupported operations must be independently performed on the repository and must not go through CA AuthMinder flows. For example, users must be created in LDAP through LDAP user interface or APIs.

Error Code	Error Message	Possible Cause for Failure
31128	User, {0} already exists. **Note:** The createUser API throws this error.	**Possible Cause:** The specified user already exists in the system. Therefore, cannot create another user with the same UserID. **Solution:** UserID must be unique in an organization. Create an user with different UserID.
31119	User identifier, {0} is mandatory.	**Possible Cause:** API was called without providing user identifier, which is mandatory for the given API call. **Solution:** Provide a valid user identifier in the API request.
31129	PAM is not set. This is a C++ error code.	**Possible Causes:** 1. Specified user was not found. 2. Database connectivity is lost. **Solution:** 1. See the arcotadmin.log and arcotuds.log files and take corrective action. 2. If the issue is not resolved, then you must contact CA Support.
31131	Invalid authentication token. **Note:** This is observed when the API is enabled for authentication and authorization.	**Possible Causes:** 1. The authentication token is not provided in the request. 2. The specified authentication token is not valid or has been tampered with. **Solution:** Provide a valid authentication token, if the API is enabled for AnA
31132	Invalid authentication request. **Note:** This is observed when the API is enabled for authentication and authorization.	**Possible Causes:** 1. The authentication token provided in the request has expired. 2. The SOAP request is invalid. **Solution:** 1. Provide a valid authentication token in the request. 2. Obtain a new authentication token, if required.

Error Code	Error Message	Possible Cause for Failure
31135	Hashing of authentication token failed. **Note:** This is observed when the API is enabled for authentication and authorization.	**Possible Causes:** 1. The authentication token provided in the request is invalid or has been tampered with. 2. The token could not be processed. 3. The securestore.enc file not found in the conf directory of ARCOT_HOME. **Solution:** 1. Provide a valid authentication token. 2. Re-create securestore.enc. **Book:** See *CA AuthMinder Administration Guide* for more information on how to create this file. 3. If the issue is not resolved, then you must contact CA Support.
70611	Authentication failed.	**Possible Causes:** 1. Incorrect username or password has been specified in the request. 2. The administrator status is not valid. 3. The Account is locked. 4. Account has expired. **Solution:** Retry with valid password or contact the administrator to unlock or activate the account.
70300	Administrator {0} (organization: {1}) does not have the privilege to perform administration operations for organization, {2}. **Note:** This is observed when the API is enabled for authentication and authorization.	**Possible Cause:** The administrator performing the operation has a limited scope, and does not have the required permissions to perform the current task. **Solution:** 1. Contact an administrator at higher level for the required permissions.

Error Code	Error Message	Possible Cause for Failure
31136	Delete operation for product {0} failed. **Note:** This error is seen when the Cascade Delete feature is enabled.	**Possible Cause:** A database error occurred during a delete operation. **Solution:** 1. Retry the operation. 2. If the error persists, then you must contact CA Support.
31137	Invalid search expression. **Note:** This error is thrown by the searchUsers API.	**Possible Cause:** The search expression that you provided is not valid. **Solution:** Provide valid search expressions. Refer to the WSDL documentation to know more about valid search expressions.
31138	Invalid start ({0}) or end ({1}) index specified. **Note:** This error is thrown by the listUsers API.	**Possible Causes:** 1.The start or end index that you provided is not vv lid (probably a a negative integer). 2. End index is less than the start index. **Solution:** Provide valid positive integers for the start and end indexes.
31139	Page size, {0} exceeded the configured default search count, {1}. **Note:** This error is thrown by the listUsers API.	**Possible Cause:** Number of users that you are trying to retrieve exceeds the configured search count. **Solution:** Limit the number of users within the search count. Or increase the search count.
31151	Start lock time and End lock time are not allowed for ACTIVE user status.	**Possible Cause:** You have provided Start lock time and End lock time as inputs along with the user status, ACTIVE. **Note:** User status cannot be updated to ACTIVE if Start lock time and End lock time are also specified in the request. **Solution:** Do not provide Start and End lock dates for ACTIVE user status. These inputs are valid only if the user status is INACTIVE.

Error Code	Error Message	Possible Cause for Failure
31152	Invalid lock period. Start lock time must be before End lock time.	**Possible Cause:** Start lock time is greater than the End lock time. **Solution:** Ensure that the Start lock time is always less than the End lock time.
31153	Invalid lock Period. Start lock time cannot be before current time.	**Possible Cause:** Start lock time is less than the current time. **Solution:** Ensure that the Start lock time is always greater than the current time.
36100	Invalid input parameter: name, {0} value, {1}.	**Possible Cause:** The input value provided for the given parameter name is not valid. For example, specified email contains multi-byte characters. **Solution:** Provide valid input values. See User Attributes Validation Checks (see page 331) for more information on valid input parameters.
36101	Unsupported encoding exception: {0}.	**Possible Cause:** Error while encoding or decoding user or organization custom attributes. Invalid custom attributes. **Solution:** Retry with valid inputs.
36102	Parser exception: {0}.	**Possible Cause:** Error occurred while parsing user details from datastore. For example, the error occurred while retrieving information from the LDAP Date field. **Solution:** 1. Retry the operation. 2. If the error persists, then you must contact CA Support.

Error Code	Error Message	Possible Cause for Failure
36103	{0}.	**Possible Causes:** Error occurred while encrypting or decrypting user data, because: 1. HSM is not reachable 2. Invalid Key Label is provided in the input. 3. Server Cache has not been updated. **Solution:** 1. Retry after cache refresh. 2. Verify if the specified Key Label exists in the HSM. 3. Ensure that the HSM connectivity is not lost.
36106	User {0} not updated.	**Possible Causes:** 1. Specified user not found. 2. Database connectivity is lost. **Solution:** 1. Retry the operation. 2. See if there are any database-related errors in arcotadmin.log and arcotuds.log and take corrective action. 3. If the issue is not resolved, then you must contact CA Support.
36107	Provided image size, {0} KB, exceeds the maximum supported size, {1} KB.	**Possible Cause:** The user image has exceeded the supported size. **Solution:** Ensure that the image size is within the supported range. See User Attributes Validation Checks (see page 331) for more information on valid input parameters.
36108	Invalid image format, {0}. Supported image formats are JPEG, GIF, BMP, and PNG only.	**Possible Cause:** The format of the user image provided in the request is not valid. **Solution:** Ensure that the image format you are using is supported by UDS. See User Attributes Validation Checks (see page 331) for more information on valid input parameters.

Error Code	Error Message	Possible Cause for Failure
31134	Invalid input parameter. {0} is not an enterprise LDAP organization.	**Possible Cause:** You passed non-LDAP organization details to the LDAP-only API, such as API=performQnaVerification **Solution:** Ensure that you use the correct non-LDAP APIs.
31108	Invalid input parameter: name, {0} and value, {1}.	**Possible Cause:** The value provided for the specified parameter is not valid. **Solution:** Provide valid inputs. See User Attributes Validation Checks (see page 331) for more information on valid input parameters.
31109	Organization with name {0} already exists.	**Possible Cause:** Organization with the specified name already exists. As a result, another organization with the same name cannot be created. **Solution:** Provide a unique organization name.
31110	Organization with the display name {0} already exists.	**Possible Cause:** Organization with the specified display name already exists. As a result, another organization with the same display name cannot be created. **Solution:** Provide a unique display name for the organization.
31114	Operation, {0} is not supported for organization {1} with status {2}.	**Possible Cause:** The specified operation is not supported for the specified status of an organization. For example, setting an INACTIVE organization as default is not allowed. **Solution:** Update the organization status and then perform the operation.
31115	Organization, {0} with status, {1} does not exist.	**Possible Cause:** The specified organization with the given status does not exist. **Solution:** Verify if the organization exists and check if its status has changed.

Error Code	Error Message	Possible Cause for Failure
31116	Organization {0} is already deleted.	**Possible Cause:** The organization that you are trying to delete cannot be deleted, because it has already been deleted. **Solution:** Ensure that you specify the correct organization details.
31117	Unable to connect to the repository, {0}.	**Possible Causes:** ■ Repository is down. ■ Specified Repository connection details are not correct. **Solution:** Check the repository connection details and retry.
31121	Invalid organization status, {0}.	**Possible Cause:** The organization status is not valid for the given operation. For example, the organization cannot be created with status INACTIVE. **Solution:** Update the organization status to a valid one.
31122	Operation, {0} not supported for default organization {1}.	**Possible Cause:** The specified operation is not supported on Default organization. For example, Default organization cannot be deleted.
31124	Organization, {0} does not exist.	**Possible Causes:** ■ The organization name provided as input is not valid. ■ The Server cache has not been updated. ■ The organization with the specified name does not exist. **Solution:** Retry the operation after cache refresh.

Error Code	Error Message	Possible Cause for Failure
31140	Attribute encryption failed.	**Possible Causes:** ■ The organization display name provided as input is not valid. ■ The Server cache has not been updated. ■ The organization with the specified display name does not exist. **Solution:** Retry the operation after cache refresh.
31142	Invalid key label. No key with alias, {0} exists. **Note:** This error is thrown if the system is configured for hardware encryption.	**Possible Causes:** ■ The Key Label specified in the operation does not exist in the HSM. ■ The HSM connection failed. **Solution:** 1. Check the HSM connectivity. 2. Verify if the Key Label is present in the HSM.
31143	Organization {0} exist with the same LDAP configuration. **Note:** The createOrg APIs throw this error.	**Possible Cause:** An LDAP organization with the same configuration already exists. **Solution:** LDAP organizations must have unique configurations. Update the existing organization or create a new organization with different details.
31146	Error saving custom attributes for user {0}. **Note:** This error is thrown while updating users' custom attributes.	**Possible Causes:** ■ The specified user was not found. ■ The database connection failed. **Solution:** 1. Retry the operation. 2. If the issue is not resolved, then you must contact CA Support.

Error Code	Error Message	Possible Cause for Failure
38100	Resource, {0} of type, {1} does not exist. **Note:** This error is thrown when the processing of an account type fails.	**Possible Causes:** ■ The specified account type does not exist. ■ The Server cache has not been updated. **Solution:** Refresh the cache and retry.
38101	Unknown or unexpected error.	**Possible Cause:** A critical internal error occurred. **Solution:** Contact CA Support.
39100	User account, {0} not found for account type, {1}.	**Possible Causes:** The specified user account was not found for the given account type, because: ■ The specified user does not have an account. ■ The account has been deleted. **Solution:** Verify if the input Account ID is valid.
39101	The custom attribute, {0} is invalid for account type, {1}.	**Possible Cause:** ■ The specified account custom attribute does not exist for the account type. ■ The Server cache has not been refreshed. **Solution:** 1. Ensure that you provide a valid input. 2. Refresh the cache and retry.
39102	User identifier, {0} not found for organization, {1}.	**Possible Cause:** The user identifier was not found for the organization during deep search. In other words, the specified identifier did *not* match any of the userid, accountid, and accountid attributes. **Solution:** Provide valid user identifier.

Error Code	Error Message	Possible Cause for Failure
39103	Account(s) creation failed for user identifier, {0} belonging to the organization, {1}. **Note:** This is a C++ error code.	**Possible Causes:** ■ The connection to the database failed. ■ An unexpected error occurred while creating account(s). **Solution:** 1. See if there are any related errors in arcotadmin.log and arcotuds.log and take corrective action. 2. Retry the operation.
39104	The specified user account already exists for user {0}.	**Possible Cause:** The specified user account already exists. **Solution:** Ensure that you specify a unique account name.
39105	Account types do not exist for organization, {0}.	**Possible Causes:** ■ The specified account type is not available for the organization. ■ The Server cache has not been refreshed. **Solution:** 1. Ensure that you specify the correct account type name. 2. Add the organization to the account type scope. 3. Retry after cache refresh.
39106	Account type already exists. **Note:** The APIs for creating and updating account types throw this error.	**Possible Cause:** The specified account type exists in the system. As a result, another accounttype with the same name or display name cannot be created. **Solution:** Create an accounttype with a unique name or display name.
39107	Account ID, {0} already created for the account type, {1}.	**Possible Cause:** The specified user account has already been created for the given account type. **Solution:** The Account ID must be unique for a given account type for an organization. Provide a different Account ID and retry.

Error Code	Error Message	Possible Cause for Failure
39201	SOAP action is null.	**Possible Cause:** The incoming SOAP request is not valid. **Solution:** 1. Provide a valid SOAP request. 2. If problem persists, then contact CA Support.
55000	Invalid input parameter, {0}.	**Possible Cause:** The input value provided for the given parameter is not valid. **Solution:** See User Attributes Validation Checks (see page 331) for more information on valid input parameters.
55001	Missing input parameter, {0}.	**Possible Cause:** The required parameter is missing from the API request. **Solution:** Provide the required parameter.
55002	Insufficient input parameters. **Note:** This is a C++ error code.	**Possible Cause:** The API inputs are incomplete. **Solution:** Provide all the required inputs.
55003	Resource bundles not found. **Note:** This is a C++ error code.	**Possible Causes:** ■ ARCOT_HOME is not set. ■ The required properties files are missing from the resourcebundles subdirectory in the conf directory. **Solution:** Ensure that the ARCOT_HOME environment variable is correctly set.
55004	Database error. **Note:** This is a C++ error code.	**Possible Causes:** ■ The connection to the database failed. ■ An unexpected error occurred while writing to or reading from the database. **Solution:** 1. Refer to arcotadmin.log and arcotuds.log for more information. 2. Retry the connection.

Error Code	Error Message	Possible Cause for Failure
55010	Error while encrypting the data.	**Possible Causes:** ■ The connection to the HSM failed. ■ An unexpected error occurred. ■ The Server cache has not been refreshed. **Solution:** 1. Refer to arcotadmin.log and arcotuds.log for more information. 2. Check HSM connection details. 3. Refresh the Server cache.

Error Code	Error Message	Possible Cause for Failure
55011	Error while decrypting the data. **Note:** This is a C++ error code.	**Possible Causes:** ■ The connection to the HSM failed. ■ An unexpected error occurred. ■ The Server cache has not been refreshed. **Solution:** 1. Refer to arcotadmin.log and arcotuds.log for more information. 2. Check HSM connection details. 3. Refresh the Server cache.
55012	Internal error occurred. **Note:** This is a C++ error code.	**Possible Cause:** An unexpected error occurred. **Solution:** Refer to arcotadmin.log and arcotuds.log for more information.
55100	Error while retrieving organization configuration data. **Note:** This is a C++ error code.	**Possible Causes:** ■ The specified organization was not found in the system. ■ The connection to the database failed. ■ The Server cache has not been refreshed. **Solution:** 1. Refer to arcotadmin.log and arcotuds.log for more information. 2. Check the database connection details. 3. Refresh the Server cache.
50030	Search size limit exceeded the maximum value. **Note:** This is a C++ error code.	**Possible Cause:** Search returned more than the maximum configured limit. **Solution:** Refine your search criteria.

Error Code	Error Message	Possible Cause for Failure
50031	Search base node context needs to be bound.	**Possible Causes:** ■ The connection to LDAP organization failed. ■ The specified LDAP connection details are not valid. **Solution:** Ensure that the LDAP connection details are correct and then retry.
50032	Search is not expected to return more than set limit values. **Note:** This error is thrown by the retrieveUser API in context of LDAP organizations.	**Possible Causes:** 1. The user you specified is not unique in the system (LDAP). As a result, more than one user details were returned for the given UserID. 2. The UserID attribute is not mapped to the correct LDAP attribute. **Solution:** Ensure that the UserID is unique. Also verify the UserID attribute mappings.
50033	Search criteria is not valid. **Note:** This is a C++ error code.	**Possible Cause:** The search input that you provided is not valid. **Solution:** Provide valid search inputs.
50034	Unable to get supporting data access class.	**Possible Cause:** ■ The connection to LDAP organization failed. ■ The specified LDAP connection details are not valid. **Solution:** Ensure that the LDAP connection details are correct and then retry.
50035	LDAP node has to be created before referencing.	**Possible Cause:** ■ The connection to LDAP organization failed. ■ The specified LDAP connection details are not valid. **Solution:** Ensure that the LDAP connection details are correct and then retry.

Error Code	Error Message	Possible Cause for Failure
50036	LDAP repository has to be initialized.	**Possible Cause:** ■ The connection to LDAP organization failed. ■ The specified LDAP connection details are not valid. **Solution:** Ensure that the LDAP connection details are correct and then retry.
50037	Context is not bound or cannot be created.	**Possible Cause:** ■ The connection to LDAP organization failed. ■ The specified LDAP connection details are not valid. **Solution:** Ensure that the LDAP connection details are correct and then retry.

AuthMinder Server Codes

The following table lists the response codes, reason codes, the cause for failure, and solution wherever applicable.

Response Code	Reason Code	Description	Possible Cause for Failure
0	0	Operation completed successfully.	N/A
	6100	Authentication succeeded, but the credential is in grace period.	**Action to Take:** Credential has already expired. Notify the user to get the credential reissued.
	6101	Authentication succeeded, but the credential is in warning period.	**Action to Take:** Credential is about to expire. Notify the user to get the credential reissued.
1000	0	Internal error.	**Possible Cause:** Unexpected internal error.

Response Code	Reason Code	Description	Possible Cause for Failure
	2000	Database is not operational.	**Possible Cause:** Database is not operational. **Solution:** Start the database. **Possible Cause:** Connection between the server and database is not complete. **Solution:** Establish the connection between server and database again using the database parameters available in arcotcommon.ini file.
	2001	Configuration is missing.	**Possible Cause:** Configuration required for processing the transaction is missing. **Solution:** Check the server transaction logs for details and ensure the required configuration is created and assigned. **Possible Cause:** Configuration required for processing the transaction is created but not available in server cache. **Solution:** Refresh server cache.

Response Code	Reason Code	Description	Possible Cause for Failure
	2002	Transaction ID generation failed.	**Possible Cause:** Transaction ID generation failed due to internal error in the server. **Solution:** Most likely cause might be because of database failure. Check the server transaction logs for details and ensure appropriate action is taken based on the server logs.
	6004	Internal error.	**Possible Cause:** Unexpected internal error.
1001	0	Access is denied.	**Possible Cause:** The operation being invoked is protected, and you need to authenticate. **Solution:** Obtain authentication credentials from your administration to include them in the call. You must send the correct credential or authorization token while making the Web Service call. See chapter, "Managing Web Services Security" (see page 35) for more information.
1050	0	Invalid parameter.	**Possible Cause:** The input parameter is invalid. **Solution:** Provide a valid parameter.

Response Code	Reason Code	Description	Possible Cause for Failure
	2050	Value of one of the parameters used in the operation is empty.	**Possible Cause:** The parameter passed to the API is empty. **Solution:** Provide a non-empty value for the parameter. See appendix, "Input Data Validations" (see page 315) for the supported parameter values.
	2051	Length of one of the parameters used in the operation has exceeded the maximum allowed value. **Tip:** Length here refers to length of the parameter, for example password length.	**Possible Cause:** The length of the parameter passed to the API has exceeded the maximum value. **Solution:** Provide the parameter such that its length is less than or equal to the maximum allowed value. See appendix, "Input Data Validations" (see page 315) for the supported parameter values.
	2052	Length of one of the parameters used in the operation is less than minimum allowed value.	**Possible Cause:** The length of the parameter passed to the API is less than minimum value. **Solution:** Provide the parameter such that the length of the parameter is greater than or equal to the minimum allowed value. See appendix, "Input Data Validations" (see page 315) for the supported parameter values.

Response Code	Reason Code	Description	Possible Cause for Failure
	2053	Value of one of the parameters used in the operation exceeded the maximum allowed value. **Tip:** VALUE here refers to value of the parameter, for example ArcotID PKI Plain Key length.	**Possible Cause:** The value of the parameter passed to the API has exceeded the maximum allowed value. **Solution:** Provide the parameter such that the value of the parameter is less than or equal to the maximum allowed value. See appendix, "Input Data Validations" (see page 315) for the supported parameter values.
	2054	Value of one of the parameters used in the operation is less than the minimum allowed value.	**Possible Cause:** The value of the parameter passed to the API is less than the minimum allowed value. **Solution:** Provide the parameter such that the value of the parameter is greater than or equal to the minimum allowed value. See appendix, "Input Data Validations" (see page 315) for the supported parameter values.

Response Code	Reason Code	Description	Possible Cause for Failure
	2055	Value of one of the parameters used in the operation is invalid.	**Possible Cause:** The value of the parameter passed to the API is invalid. For example, the allowed values for user status are 0 and 1. If you set the value of this as 5, then you will get this error. **Solution:** Provide valid value for the parameter. See appendix, "Input Data Validations" (see page 315) for the supported parameter values.
	2056	Value of one of the parameters used in the operation contains invalid characters.	**Possible Cause:** The parameter specified by ParameterKey contains invalid characters. **Solution:** Provide valid characters for the parameter that is specified by ParameterKey.
1050	2057	One of the parameters used in the operation does not meet the formatting requirements.	**Possible Cause:** The parameter specified by ParameterKey has invalid format. **Solution:** Provide valid format for the parameter that is specified by ParameterKey.

Response Code	Reason Code	Description	Possible Cause for Failure
	2058	The password has less number of alphabets than the minimum allowed value.	**Possible Cause:** The password provided contains lesser number of alphabets than the password strength policy allows. **Solution:** Refer to the relevant password policy and ensure that the password strength is set correctly.
	2059	The password has less number of numeric characters than the minimum allowed value.	**Possible Cause:** The password provided contains lesser number of numeric characters than the password strength policy allows. **Solution:** Refer to the relevant password policy and ensure that the password strength is set correctly.
	2060	The password has less number of ASCII special characters than the minimum allowed value.	**Possible Cause:** The password provided contains lesser number of ASCII special characters than the password strength policy allows. **Solution:** Refer to the relevant password policy and ensure that the password strength is set correctly.

Response Code	Reason Code	Description	Possible Cause for Failure
	2061	Parameter is not supported for this operation.	**Possible Cause:** The parameter that is passed by the plug-in is not supported by the operation. For example, if you pass SAML token configuration name in the createCredential operation. **Solution:** Change the plug-in code appropriately.
1050	2063	Password is invalid.	**Possible Cause:** The PKCS#12 files are uploaded with a wrong password. **Solution:** Ensure that you use the correct password for the PKCS#12 files.
	2064	Update operation is not supported for the parameter.	**Possible Cause:** You are trying to update a read-only parameter. **Solution:** None.
	2065	Parameter does not match.	**Possible Cause:** The organization name specified in the XML file to upload the OATH tokens does not match with organization name specified in the operation. **Solution:** Provide the correct organization name.
	6000	Duplicate questions are not supported.	**Possible Cause:** Two or more questions are same. **Solution:** Provide distinct questions.

Response Code	Reason Code	Description	Possible Cause for Failure
	6001	Duplicate answers are not supported.	**Possible Cause:** Two or more answers are same. **Solution:** Provide distinct answers.
	6002	The question cannot be same as any of the answers.	**Possible Cause:** Question might be same as any of the answers. **Solution:** Provide distinct question and answer.
1050	6007	Credential history check failed.	**Possible Cause:** The credential that you are trying to update failed the password history validation check. **Solution:** Ensure that the password that you have specified meets the history check criterion.
	6010	Question not found.	**Possible Cause:** Question that you are trying to update, delete, and for which you want to update answer does not exist. **Solution:** Ensure that you use the correct question.
	6105	Duplicate elements found.	**Possible Cause:** The PKCS12 file being uploaded in to the ArcotID PKI contains duplicate elements. **Solution:** Upload a PKCS#12 file that does not contain duplicate entries.

Response Code	Reason Code	Description	Possible Cause for Failure
	6106	Invalid element reference.	**Possible Cause:** The element that you are trying to delete does not exist in the ArcotID PKI. **Solution:** Ensure that you use the correct element identifier.
	6200	Event is already assigned.	**Possible Cause:** The event is already associated with an organization. **Solution:** Choose a different event to assign.
	6201	Duplicate events are not supported.	**Possible Cause:** The event list passed contains duplicate entries. **Solution:** Do not assign duplicate events.
1051	0	Invalid request.	**Possible Cause:** The packet received is invalid. **Solution:** 1. Ensure correct SDK is pointing to the server. 2.Ensure the port cconfigured on the client-side refers to the appropriate server protocol.

Response Code	Reason Code	Description	Possible Cause for Failure
1060	0	The request is noted.	**Possible Cause:** Caller verification of the QnA credential is successful. In this case server does not apply the authentication policy. **Solution:** NA
1100	0	Organization is not found.	**Possible Cause:** Organization specified is not present. **Solution:** 1. Check if the organization with the given name is created. 2. After creating the organization, the server might need cache refresh. Refresh the server cache. 3. Check if the name of the organization passed is correct.
1101	0	Credential configuration not found for the organization.	**Possible Cause:** The configuration for the specified credential is not present. **Solution:** 1. Check if the configuration is created for this organization. 2. Check if the configuration is assigned

Response Code	Reason Code	Description	Possible Cause for Failure
			to this organization. 3. Creating and assigning configuration might need cache refresh. Refresh the server cache.
1102	0	User not found.	**Possible Cause:** User is not present. **Solution:** Create the user or provide the user information correctly.
1103	0	Organization is not active.	**Possible Cause:** Organization is not active. **Solution:** Activate the organization using Administration Console.
1104	0	Configuration already exists.	**Possible Cause:** The configuration that you are trying to create already exists. **Solution:** If you want to create a configuration, the use a different configuration name. If you want to update an existing configuration, then use the correct operation.
1150	0	User status is not active.	**Possible Cause**: User status is not active. **Solution:** Activate the user by using Administration Console.

Response Code	Reason Code	Description	Possible Cause for Failure
1151	0	User already exists.	**Possible Cause**: User already present in the system. **Solution:** Create the user with different user name or provide the user details correctly.
1152	0	Credential is invalid.	**Possible Cause:** Credential already present for the user. **Solution:** Do not create a credential that already exists for the user.
5500	0	Processor is invalid. Note that processor refers to authentication mechanism.	**Possible Cause:** The mechanism requested is not supported by the system. **Solution:** Use mechanisms supported by AuthMinder.
5501	0	Data not found.	**Possible Cause:** There was no data found for the specified OATH token search criteria. **Solution:** Use a different search criteria.
5600	0	The RADIUS client IP is not valid.	**Possible Cause:** Client IP used in the RADIUS configuration is not valid. **Solution:** Ensure that you use an appropriate octet IP format.

Response Code	Reason Code	Description	Possible Cause for Failure
5601	0	The credential configuration is not valid.	**Possible Cause:** The configuration passed in the input is not valid. **Solution:** Based on the operation being performed there could be multiple reasons for this error. Check the parameter details in the response or check the server logs for further details.
	2003	Configuration organization does not match with the request organization.	**Possible Cause:** The organization name specified in the OATH token does not match with the organization name that you have specified in the operation. **Solution:** Ensure that you provide the correct organization name.
5601	6005	OATH token not found.	**Possible Cause:** OATH token being assigned is not uploaded to the organization or it might not be uploaded for the organization the current user belongs to. **Solution:** Check the token identifier and ensure that you upload the OATH token at the global level or for the current organization.

Response Code	Reason Code	Description	Possible Cause for Failure
	6006	OATH token is already assigned to a user.	**Possible Cause:** The OATH token has already been assigned. **Solution:** Assign a different OATH token for the user.
	6009	OATH token is abandoned.	**Possible Cause:** The OATH token has been used and abandoned. **Solution:** Assign a different OATH token for the user or reuse the same token by force-assigning the token.
	6104	Credential key is not active.	**Possible Cause:** The key with which the credential is protected is no longer ACTIVE. **Solution:** Reissue and use the new credential.
5602	0	The protocol is not valid.	**Possible Cause:** The protocol that you are trying to update or fetch is not valid. **Solution:** Use a valid protocol identifier.
5603	0	The credential configuration for the organization is not valid.	**Possible Cause:** The credential configuration name is not valid. **Solution:** You must provide a valid configuration name.

Response Code	Reason Code	Description	Possible Cause for Failure
5605	0	SSL trust store group name is invalid.	**Possible Cause:** The provided organization name is not valid. **Solution:** You must provide a valid organization name.
5606	0	SSL trust store group is invalid.	**Possible Cause:** SSL trust store with this name already exists. **Solution:** Create a trust store with a different name.
5607	0	Invalid WebFort Server instance name.	**Possible Cause:** Server instance name being set is not valid. **Solution:** Provide a valid instance name.
5608	0	A RADIUS client with the specified IP address is already configured.	**Possible Cause:** The IP address specified in the operation has already been configured. **Solution:** If the existing configuration is not correct, then delete that configuration and create a new configuration.
5700	0	Number of authentication attempts exceeded.	**Possible Cause**: Number of authentication attempts for the credential exceeded the allowed limit. **Solution:** The administrator must change the status of the credential from locked to active.

Response Code	Reason Code	Description	Possible Cause for Failure
5701	0	Authentication token has expired.	**Possible Cause**: Authentication token submitted by the user is expired. **Solution:** Authenticate again.
5702	0	Challenge has expired.	**Possible Cause:** Challenge is expired. **Solution:** Request for the challenge again.
5704	0	Credential has expired.	**Possible Cause:** The credential, which is provided by the user is expired. **Solution:** Get the new credential.
	0	The credential configured for ASSP has expired.	**Possible Cause:** The credential, which is provided by the user is expired. **Solution:** Get the new credential.
	6102	The credential validity period has not yet started.	**Possible Cause:** The credential has been created for future use. **Solution:** Use the credential that is within the validity period.
5705	0	Credential is not active.	**Possible Cause:** The credential, which is provided by the user is not active. **Solution:** The administrator must activate the credential.

Response Code	Reason Code	Description	Possible Cause for Failure
	0	Credential is not active" "ASSP" "The user ($$(USER)$$) account is inactive."	**Possible Cause:** The credential, which is provided by the user is not active. **Solution:** The administrator must activate the credential.
5706	0	Credential is reissued.	**Possible Cause:** Credential is reissued.
5707	0	The authentication credentials provided are incorrect.	**Possible Cause:** The credential details provided by the user are incorrect. **Solution:** Provide the credential details correctly.
	0	The ASSP authentication credentials provided are incorrect.	**Possible Cause:** The credential details provided by the user are incorrect. **Solution:** Provide the credential details correctly.
	6103	The authentication credentials provided are incorrect. Re-synchronize the credential.	**Possible Cause:** The OTP that is provided is not in the configured authentication window, but can be synchronized. **Solution:** Synchronize the OTP credential.

Response Code	**Reason Code**	**Description**	**Possible Cause for Failure**
5800	0	Credential not found for the user.	**Possible Cause:** The credential does not exist for the user. **Solution:** Create the credential. **Possible Cause:** The details provided by the user might be incorrect. **Solution:** Provide the correct details.
	0	ASSP credential not found for the user.	**Possible Cause:** The credential does not exist for the user. **Solution:** Create the credential. **Possible Cause:** The details provided by the user might be incorrect. **Solution:** Provide the correct details.
	6004	The credential not found for the user. It is already been deleted.	**Possible Cause:** The credential has already deleted. **Solution:** You can perform a fetch operation on the credential to understand the credential state. Reissue the credential, if required.
5801	0	Credential already present for the user.	**Possible Cause:** Credential already exists for the user.
	6008	Credential already present for the PAN.	**Possible Cause:** Credential already exists for the user.

Response Code	Reason Code	Description	Possible Cause for Failure
6500	0	The event is not supported.	**Possible Cause:** The event that being assigned to the plug-in is not supported by AuthMinder. **Solution:** Ensure that you use the supported events.
6501	0	The operation is not supported.	**Possible Cause:** The credential input provided is not valid. For example, you might have provided QnA input for the downloadCredential operation. **Solution:** Ensure that the input data that you provide is correct.