CA AuthMinder™

Installation and Deployment Guide for UNIX Platforms

r7.1.01

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.

Chapter 1: Understanding the Basics

With the exponential increase in cases of Internet-based fraud over the last few years, relying on user names and passwords for authentication is no longer sufficient. The need for stronger authentication can either be to protect end users or to comply with government-mandated security requirements, internal policies, or best practices.

However, adding stronger authentication often creates conflict between compliance requirements and user convenience. Organizations want to increase the security of their authentication processes by reducing complexity.

Organizations also want to reduce the risk of financial losses or brand damage while increasing customer and partner access to applications and data.

Strong Authentication is a strong authentication service that enables your application to verify and protect the identity of your end users by:

- Not transmitting passwords (either in clear or encrypted form) over the network.
- Enabling you to select the authentication method that best suits the security and convenience of different types of users.
- Using CA AuthID® and CA AuthID OTP, which are based on the patented Cryptographic Camouflage technique to protect keys.

  In *Cryptographic Camouflage*, the keys are not encrypted with a password that is too long for exhaustive attacks. Instead, keys are encrypted such that only one password can decrypt it correctly, but many passwords can decrypt it to produce a key that looks valid enough to fool an attacker. This method protects a user's private key against dictionary attacks and Man-in-the-Middle (MITM) attacks, as a smartcard does, but entirely in the software format.

  See "How Cryptographic Camouflage Works" (see page 14) for more information.

This guide provides information for planning the deployment of Strong Authentication based on different solution requirements. Each solution consists of multiple components that interact with each other and other systems in an enterprise or multiple-network systems.

Note: Strong Authentication still contains the terms Arcot and WebFort in some of its code objects and other artifacts. Therefore, you will find occurrences of Arcot and WebFort in the documentation. In addition, some of the topics in this guide do not follow the standard formatting guidelines. These inconsistencies will be fixed in a future release.

Strong Authentication as a Versatile Authentication Server

Strong Authentication is a *Versatile Authentication Server* (*VAS*) due to support of the implementation of a wide range of proprietary and open authentication mechanisms. In addition to supporting authentication by using Public Key Infrastructure (PKI) and one-time password (OTP/Activation Code), it is also designed to plug in any existing authentication methods. This enables your organization to handle changes to critical systems and partner applications seamlessly.

The VAS functionality of Strong Authentication provides your organization the flexibility to select the authentication method that best suits the needs of your end users. You can choose to:

- Integrate with a variety of standard authentication interfaces.
- Implement standards-based hardware or software authentication methods.
- Add new authentication methods, such as CA AuthID, while continuing to support legacy technology, such as OTP/Activation Code tokens.
- Extend Strong Authentication VAS through plug-ins to perform proprietary authentication.

Strong Authentication Plug Ins

Strong Authentication provides the following authentication methods out-of-the-box:

- **CA AuthID**

  CA AuthID is a CA-proprietary secure software credential that provides two-factor authentication. The CA AuthID 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).

  See "CA AuthID Key Concepts" (see page 11) for more information about CA AuthID.

- **Password**

  A regular credential, where the user is issued a username and a password to log in to the system.

- **One-Time Password**

  One-time password is another credential generated by Strong Authentication Server. An OTP/Activation Code can be numeric or an alpha-numeric string. It is also possible to configure the number of times it can be used.

- **OATH-Compliant One-Time Password**

  One-time passwords that are complaint to Open Authentication (OATH) standards. Strong Authentication supports both counter-based OATH OTP/Activation Codes (HOTPs) and time-based OATH OTP Tokens (TOTPs).

- **Question and Answer**

  Question and Answer (also known as *QnA*) is a challenge-response authentication mechanism. Users authenticate to Strong Authentication Server by providing correct answers for the questions they are asked. These Questions and Answers are set by the users themselves during registration.

- **CA MobileOTP**

  CA AuthID OTP is compliant to the OATH, Europay, MasterCard, and VISA (EMV) standards. If your application is integrated with CA AuthID OTP, then it accepts the user's password as an input and generates passwords (also known as *passcodes*) on the users' device. The user, then, submits this generated passcode to authenticate to your Web application. Based on the authentication result, the user is granted access to the protected application or denied access.

  Passcode generation is an offline process, which means that your application need not be connected to Strong Authentication for generating passcodes.

- **LDAP Username-Password**

  Strong Authentication supports LDAP authentication, where the user credentials in the directory service are used to authenticate users.

You can issue one or more of these credentials to your users. You can also issue multiple credentials of the same type. For example, you can issue two password credentials, an CA AuthID credential, and a QnA credential for a single user.

If you want to extend the default authentication mechanisms, then Strong Authentication provides you the flexibility to do so by writing Plug-Ins (see page 8).

Plug-Ins

A *plug-in* is a custom server-side component, written in C or C++, that enables you to extend the functionality of AuthMinder VAS. A plug-in is implemented as a custom event handler library within the context of the AuthMinder Server.

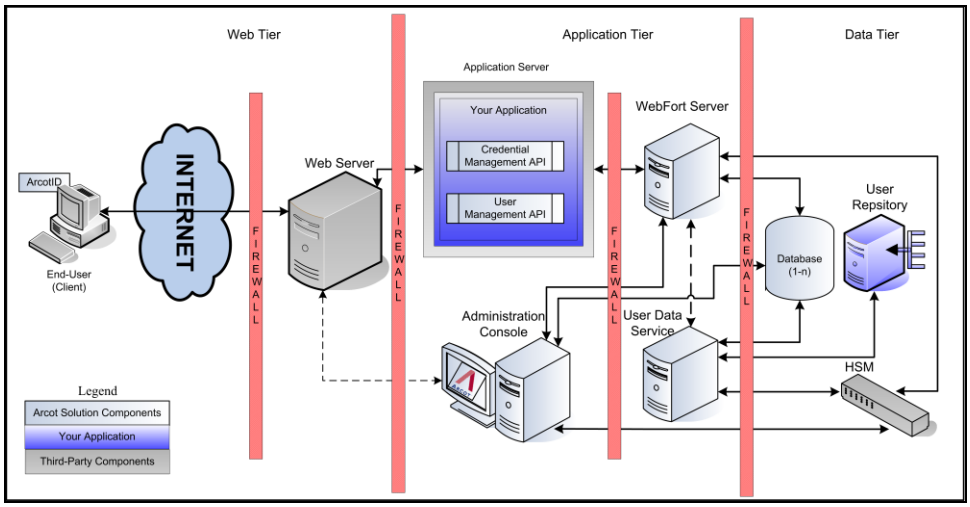
You register your plug-in (by using the AuthMinder Administration Console) to a published set of events. The plug-in is invoked when the specified event occurs.

You can configure multiple plug-ins for one organization, and you can also configure the same plug-in for multiple organizations.

Strong Authentication Architecture

You can install Strong Authentication on a single system or distribute its components across multiple systems. However, to ensure maximum security of transactions, we recommended that you implement the architecture that is shown in the following figure:

- Web Tier (see page 9)
- Application Tier (see page 10)
- Data Tier (see page 11)



Web Tier

This layer comprises the static (HTML) content and interacts directly with the user over a network or the Internet.

This layer serves the CA AuthID Client (Java, Flash, or Native) to the end user's browser. CA AuthID Client interacts with Strong Authentication Server for user authentication. It collects the CA AuthID password, signs the challenge, and then sends the signed challenge to the Strong Authentication Server for verification.

Note: See the *CA CA AuthID Client Reference Guide* for information about the CA AuthID Client.

Application Tier

This layer constitutes Strong Authentication Server, your application that use the SDKs and the application servers where the Administration Console and the User Data Service (UDS) reside.

Note: All components in this layer can be installed on one system or can be distributed across multiple systems.

- **Strong Authentication Server**

  Server component that processes issuance and authentication requests from your application through Strong Authentication SDKs.

- **Administration Console**

  Web-based console for configuring server instances, communication mode between Strong Authentication components, authentication policies, credential profiles, managing credentials, and for managing organizations, administrators, and users.

- **User Data Service**

  The abstraction layer that provides access to user- and organization-related data from different types of user repositories, such as Relational Database Management Systems (RDBMSs) and directory servers (LDAPs).

- **Authentication API**

  Java APIs that can be invoked by your application to forward authentication requests to Strong Authentication Server.

- **Credential Management API**

  Java APIs that can be invoked by your application to forward issuance requests to Strong Authentication Server for creating and managing user credentials in Strong Authentication.

- **User Management API**

  Web Services client that can be invoked by your application to forward issuance requests to User Data Service for creating and managing users in Strong Authentication.

- **Sample Application**

  Sample Application demonstrates the use of Strong Authentication Java APIs and how your application can integrate with Strong Authentication.

Data Tier

This tier comprises the RDBMSs that Strong Authentication uses to store configurations, credential information, and user data if other user repositories are not configured.

If Hardware Security Module (HSM) is used to encrypt the user sensitive data, then it is part of this tier.

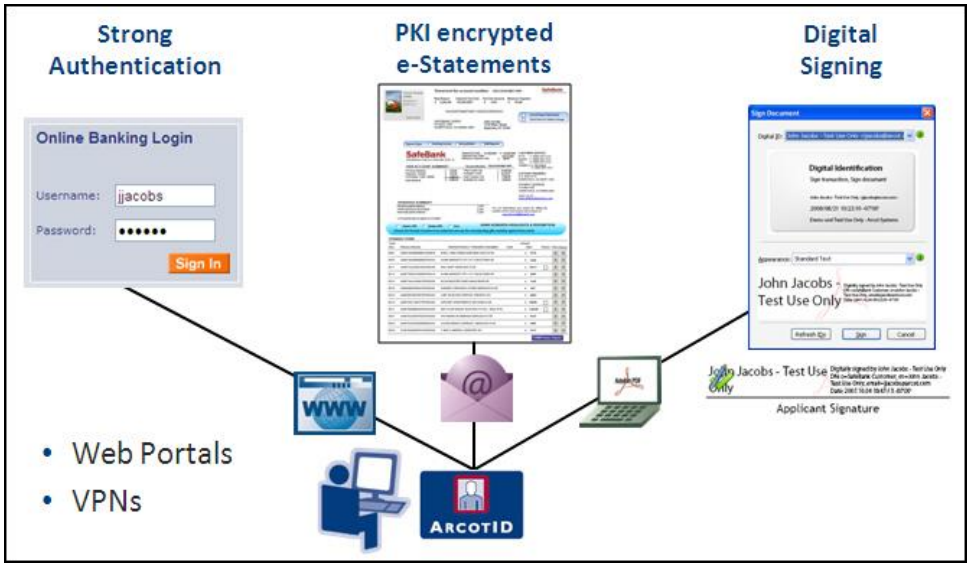
CA AuthID Key Concepts

This section introduces the key concepts of CA AuthID, which is a prime credential that Strong Authentication supports.

Introduction to CA AuthID

The CA AuthID offers the same capabilities as a physical smartcard for authentication, digital signing, encryption, and decryption for PKI-enabled applications, without requiring any end-user hardware. The CA AuthID can authenticate to any web application, even if that application does *not* support PKI-based authentication.

The following figure illustrates the use cases for CA AuthID.



The CA AuthID is a data file that is saved on an end user's computer, USB drive, or downloaded remotely for secure on-demand authentication. Unlike the simple password, an CA AuthID is *not* vulnerable to brute force password attacks. Additionally, the CA AuthID is not vulnerable to man-in-middle attacks, which, in turn, protects users from phishing attacks.

The CA AuthID can be used for strong authentication with a variety of applications, such as the Web or Virtual Private Networks (VPNs).

The CA AuthID is a configurable solution that bridges the gap between simple-but-insecure username-password-authentication and expensive-difficult-to-deploy, but very secure smartcard and USB token solutions.

The CA AuthID is based on industry standards and CA-patented Cryptographic Camouflage technology to provide software-only, strong authentication that is protected against brute force attacks.

Although an CA AuthID is protected by a password, it supports the following features to provide strong authentication:

- Only the correct CA AuthID password can access an CA AuthID.
- A plausible response is generated for every CA AuthID password entered, even if it is incorrect. As a result, preventing offline identification of the CA AuthID password is not easy.
- The CA AuthID authentication is a challenge-response authentication protocol, which ensures the user’s password is only used locally and never transmitted or verified at the server end.
- Repeated incorrect CA AuthID password entries locks out CA AuthID depending on the maximum authentication attempts configured.
- Is valid only in the domain that issued the CA AuthID.
- CA AuthIDs can only be used online, which means the user *must* connect to Strong Authentication Server to validate their CA AuthID password.

CA AuthID File Structure

The CA AuthID contains the following main components:

1. The standard X.509v3 digital certificate with a CA-specific extension.
2. A second pair of public and private keys that is generated for authenticating to Strong Authentication Server. It is not used for general signing or encryption.

   The public key is stored in the encrypted format. It is encrypted using the *Domain Key*, which is used to create and authenticate CA AuthIDs. You can configure a domain key at the global-level or at the organization-level. The CA AuthID issued with the organization-specific domain key *cannot* be used across organizations.

   The private key is cryptographically camouflaged by using the CA AuthID password.
3. A section to store the user's Open PKI keys and certificates, which they can use for signing, encrypting, and decrypting. See "CA AuthID as a Secure Container (Key Authority)" (see page 16) for more information.

How Cryptographic Camouflage Works

With the advent of support for public key cryptography in Web browsers, the use of public key cryptographic signatures and authentication protocols is becoming more common.

The security of the private key, however, remains a problem. The most basic threat is the theft of a private key that is stored on a disk. Usually such a key is stored in a software key container, a file, wherein the keys are encrypted by using a password.

An attacker that steals the container can try to guess the password using a dictionary attack.

To overcome such problems, Strong Authentication provides a method for secure storage of private keys in software, using cryptographic camouflage, where attacks on the key container are inherently supervised.

The key container embeds the user's private key among spurious private keys. An attacker who tries to crack the key container will recover many plausible private keys, but will not be able to distinguish the correct private key from the spurious decoys until they use the keys to sign the challenge and send it to the Strong Authentication Server. In such cases, Strong Authentication Server notices the multiple authentication failures and suspends the user's access.

Support Roaming Download

CA AuthID can be downloaded by using any device while the user is traveling. This feature is known as *Roaming*. Strong Authentication Server offers roaming capabilities to enable the user to securely download the CA AuthID and authenticate from any system when the need arises. This approach provides instant access to critical data and services whenever required, while keeping data safe from unauthorized access.

A roaming user can be authenticated using QnA, OTP/Activation Code, or any customized third-party solutions.

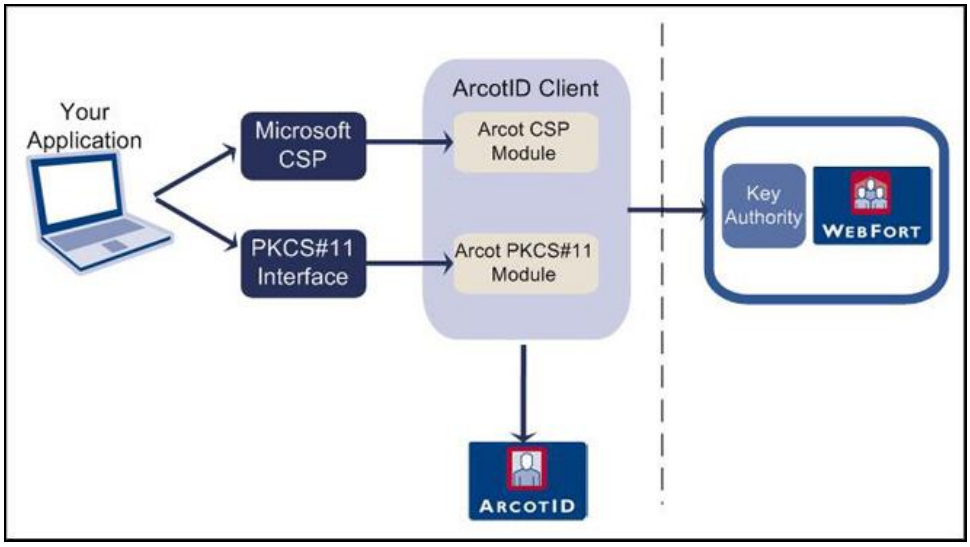
CA AuthID as a Secure Container (Key Authority)

In addition to providing strong authentication, CA AuthID can also be used as a secure container to store digital certificates and private keys that can be used for different applications or operations such as, email signing (S/MIME), document signing, certificate-based authentication (open PKI). This process of managing private key storage in the CA AuthID is performed by *Key Authority* (KA).

An unsigned attribute is created in the CA AuthID to store these credentials and this attribute is referred to as *Key Bag* or *Key Vault*. The digital certificates are stored in an unencrypted format in the Key Bag, but the private keys are encrypted using a key called *Key Authority* key, which is stored in the Strong Authentication database.

To use the private keys that are stored in a Key Bag, the CA AuthID Client (see "CA AuthID Client" (see page 17)) makes a request for the KA key to Strong Authentication Server by signing the request with the user's private key. The Strong Authentication Server authenticates the incoming request and sends the KA key to the client, which then uses this key to open the Key Bag and access the private keys.

The following figure illustrates how to use CA AuthID as an open PKI container.



CA AuthID Client

The CA AuthID Client software is used with Strong Authentication Server. The CA AuthID Client enables an end user to use an CA AuthID in a web browser to authenticate to a website, VPN, or other online resources.

To support a wide variety of application environments (operating systems, browsers, JVMs), the CA AuthID Client is available in a variety of flavors, such as:

- Native Client
- Flash Client
- Java Signed Applet
- Java Unsigned Applet
- JavaScript Client

Note: See the *CA AuthID Client Reference Guide* for more information about these client types.

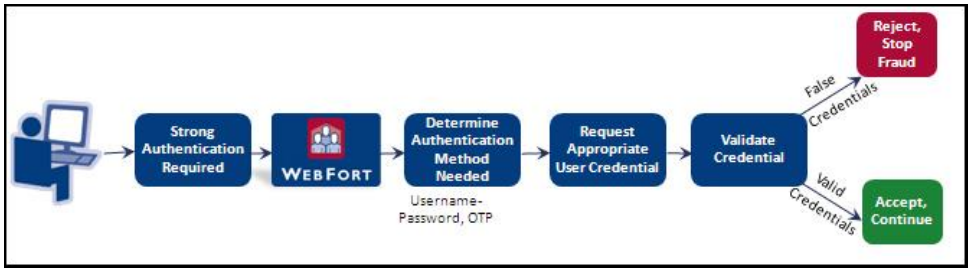
User Authentication

A user that attempts to access the web application protected by Strong Authentication is authenticated using any of the out-of-the-box credentials.

In all the authentication mechanisms, the client is provided with an authentication token after every successful authentication. The authentication token is further used to prove that the client is already authenticated to the server. The authentication token is valid only for a certain interval, after which the client has to re-authenticate to the server.

All password type credentials namely, password, OTP/Activation Code, CA AuthID OTP, and OATH OTP Token are based on the single-step authentication model, which means the credentials are passed by the client to the user and the server verifies the user credentials.

The following figure illustrates the typical authentication flow.



However, CA AuthID and QnA are based on the challenge-response authentication model. These authentication mechanisms include multiple steps to authenticate users.

How CA AuthID Authenticates Users

Authentication using CA AuthID is a PKI-based challenge-response mechanism. The client obtains an authentication token by proving the private key of the user. The client-server interactions during authentication are as follows:

1. **Get User Credentials**

Your application or the resource that is protected by Strong Authentication obtains the user credentials. For example, if the user's CA AuthID is not available on the system or the USB.

1. **Get Appropriate Challenge**

   Your application requests for a challenge used to authenticate the user.

   Strong Authentication Server prepares a unique challenge and sends it to your application.

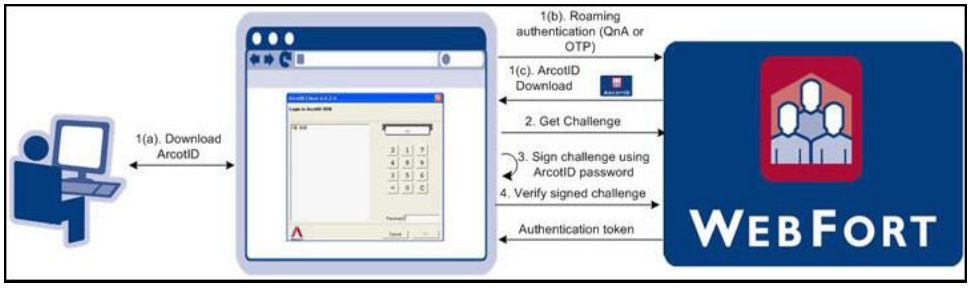
2. **Generate Signature**

   The user enters the correct CA AuthID password to uncover the CA AuthID. The client signs this challenge with the user's private key that is available as a result of uncover. The challenge can either be pre-loaded on the client machine or can be downloaded from the server.

3. **Verify Signed Challenge**

   The signed challenge is sent to the Strong Authentication Server for verification. If the signature is verified successfully, the user can login or access your protected resource. For every successful transaction, Strong Authentication also returns an authentication token for a user.

The following figure illustrates the CA AuthID authentication flow.



What's New in this Release

See the release notes for information about the key features and enhancements that have been introduced in release 7.1.01.

Chapter 2: How to Plan the Deployment

Use the information in this chapter to select a deployment model and to determine which Strong Authentication components and prerequisite software must be installed on each system. Architecture diagrams for each deployment model are also provided to assist you with planning.

Note: In this guide, *system* refers to a physical device and *server* refers to the software that is run on the system.

The chapter covers the following topics:

How to Deploy Strong Authentication

This process outlines the steps you follow to deploy Strong Authentication:

1. Choose a deployment model that suits your business needs.

   See "Choosing a Deployment Model" (see page 22) for more information.

2. Verify that the system where you plan to install Strong Authentication and its components meets all hardware requirements.

   See "Hardware Requirements" (see page 38) for more information.

3. Install the prerequisite software.

   See Software Requirement (see page 39)s for more information.

4. Configure the database server:

   - "Configuring Microsoft SQL Server" (see page 49) for more information about configuring the Microsoft SQL Server database.
   - "Configuring Oracle Database" (see page 50) for more information about configuring the Oracle Database.
   - "Configuring IBM DB2 Universal Database" (see page 54) for more information about configuring the IBM DB2 database.
   - "Configuring MySQL" (see page 57) for more information about configuring the MySQL database.

5. Install Strong Authentication:

   - "Deploying Strong Authentication on a Single System" (see page 66) for single-system deployment.
   - "Deploying Strong Authentication on Distributed Systems" (see page 100) for distributed-system deployment.

6. Run SQL scripts in the database to create the schema and set initial configuration values:

   - "Running Database Scripts" (see page 75) for single-system deployment.
   - "Running Database Scripts" (see page 112) for distributed-system deployment.

7. Copy required files and JARs on your application server. The Administration Console and the User Data Service use these files for proper functioning:

   - "Preparing Your Application Server" (see page 76) for more information about copying the files for single-system deployments.
   - "Preparing Your Application Server" (see page 113) for more information about copying the files for distributed deployments.

8. Deploy the Administration Console:

   - "Deploying Administration Console" (see page 84) for more information about deploying Administration Console for single-system deployments.

- "Deploying Administration Console" (see page 120) for more information about deploying Administration Console for distributed-system deployments.

9. Log in to Administration Console as Master Administrator:
   - "Logging In to Administration Console" (see page 86) and "Bootstrapping the System" (see page 87) for more information about initializing the Administration Console for single-system deployments.
   - "Logging In to Administration Console" (see page 122) and "Bootstrapping the System" (see page 87) for more information about initializing the Administration Console for distributed-system deployments.
10. Start the Strong Authentication Server and verify that the service has coming up correctly:
    - "Starting Strong Authentication Server" (see page 90) and "Verifying the Installation" (see page 90) for more information about starting the Server for single-system deployments.
    - See "Starting Strong Authentication Server" (see page 90) and "Verifying the Installation" (see page 90) for more information about starting the Server for distributed-system deployments.
11. Deploy and run the Sample Application to test the installation:
    - "Deploying Sample Application" (see page 94) and "Using Sample Application" (see page 94) for more information about doing this in a single-system environments.
    - "Deploying Sample Application" (see page 131), "Configuring Sample Application for Communication with Strong Authentication Server" (see page 131), and "Using Sample Application" (see page 132) for more information about doing this in a distributed environment.
12. **(Optional)** Deploy User Data Service (UDS). You need to do so *only if* you want to use directory service as user repository:
    - "Deploying User Data Service" (see page 92) for more information about deploying and starting UDS for single-system deployments.
    - "Deploying User Data Service" (see page 128) for more information about deploying and starting UDS for distributed deployments.

Deployment Overview for Upgrade

This section briefly outlines the steps for upgrading AuthMinder:

1. Check for the latest hardware and software support:
   - See "Hardware Requirements" (see page 38) for more information.
   - See Software Requirements (see page 39) for more information.
2. Upgrade your setup.

   See chapter, "Upgrading to AuthMinder 7.1.01" (see page 145) for more information about upgrade instructions.

Choosing a Deployment Model

As a part of Strong Authentication deployment, Strong Authentication Server is the primary component that you must install. You integrate your application with Strong Authentication Server by using the Java SDKs or Web Services that are shipped with Strong Authentication.

Strong Authentication also requires an SQL database for storing server configuration data, user-specific preferences, and usage data.

Typically, all Strong Authentication components are installed on a single system for development and simple testing. However, in production deployments and staging environments, Strong Authentication Server should be installed on its own system. The shipped SDKs or Web Services are installed on a different system or systems that contain the application that users log in to.

Strong Authentication is also shipped with Sample Application, which can be used to verify that Strong Authentication is installed properly and is able to perform user authentication. Sample Application also serves as a code sample for integrating Strong Authentication with your existing applications.

The high-level deployment types that are supported by Strong Authentication are:

- **Single-System Deployment** - For development or testing

  See "Deploying on a Single System" (see page 23) for more information.
- **Distributed-System Deployment** - For production or staging environments

  See "Deploying on Distributed Systems" (see page 27) for more information.
- **High-Availability Deployment** - For high availability and scalability, production, or staging environments

  See "Deploying in High-Availability Environment" (see page 32) for more information.

Single System Deployment

In a single-system deployment, all components of Strong Authentication and the web applications are installed on a single system. This deployment model is typically used for development, proof of concept, or initial testing.

It is possible to use both Java SDKs and Web Services in a single-system deployment, see "Software Requirements" for the prerequisite software for these components.

To deploy Strong Authentication on a single system, select the *Complete* option during installation. See chapter, "Deploying Strong Authentication on a Single System" (see page 66) for more information about the installation and post-installation steps.

Component Diagrams

The component diagrams depict a few of the possible deployment options for prerequisite software and Strong Authentication components. If you perform a Complete install, then both Java SDKs and Web Services are installed on the system. You can use any of these methods for integrating Strong Authentication with your Web application.

- Deploying Java SDKs
- Deploying Web Services

If you plan to perform a single-system deployment, then decide on the following:

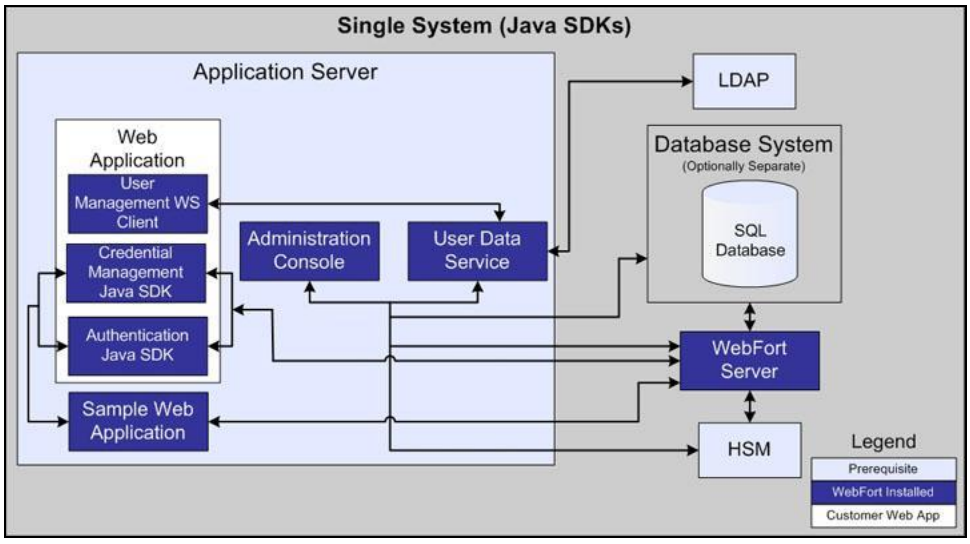
- Install a database server on the system which has Strong Authentication Server, or use an existing database on a separate system.
- Use Sample Application or write my own web application.

Important! Sample Application must *not* be used in production deployments. It is recommended that you build your own web application by using Sample Application as a code-reference.

- Use Java SDKs or Web Services to integrate with your own web application.

The following sections will help you achieve your deployment decision.

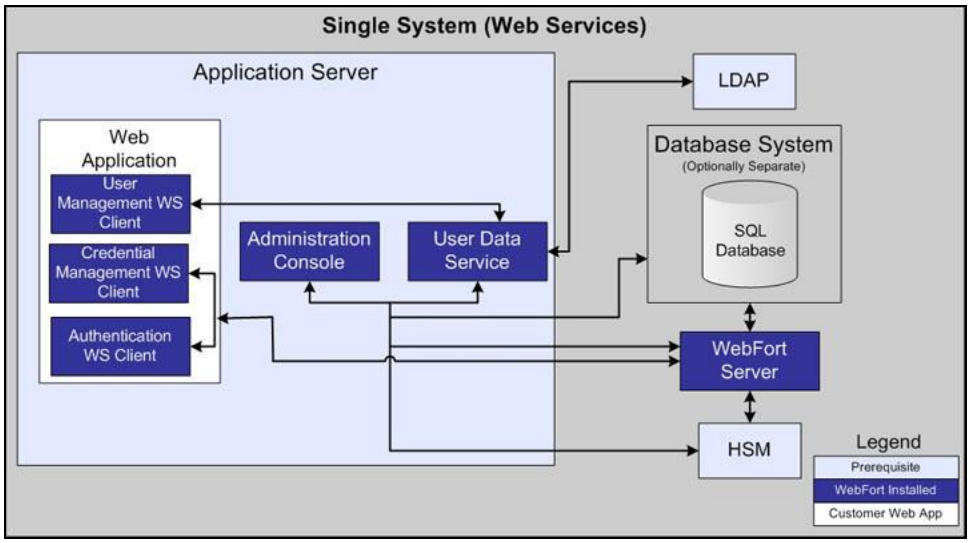
The following figure illustrates deployment of Strong Authentication Server and Java SDKs on a single system:



Note: The use of a web server to deliver HTML pages for the application server is optional and is transparent. In production deployments, this approach is generally used to improve Application Server performance and security. See the documentation of your Application Server for detailed information.

If you plan to deploy Web Services, the following figure illustrates Strong Authentication Server and Web Services on a single system.

Note: Because all web wervices are now built into the Strong Authentication Server itself, you install the Server on the target system and generate the requisite client stubs.



Decision Points:

- Install a database server on the system which has AuthMinder Server, or use an existing database on a separate system.
- Use Sample Application or write my own web application.

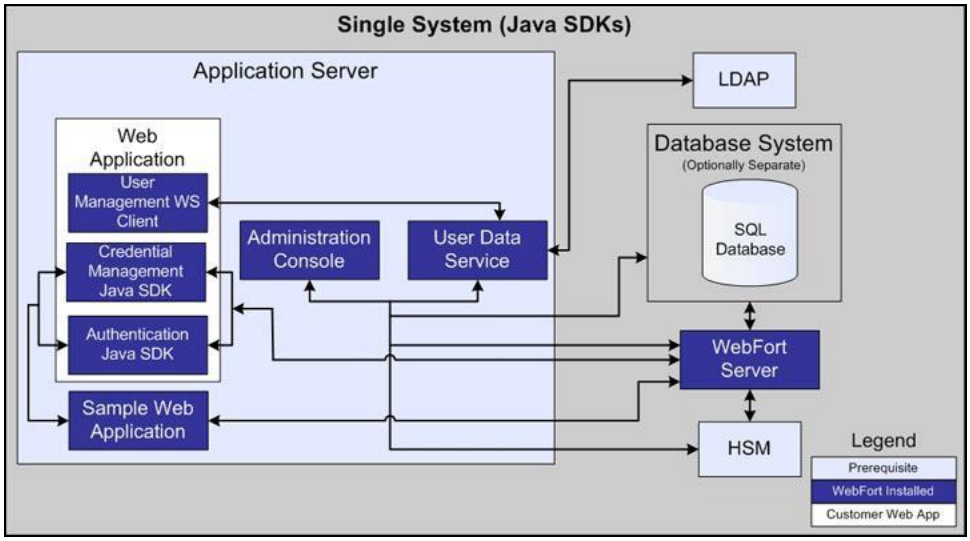
Important! Sample Application must *not* be used in production deployments. It is recommended that you build your own web application by using Sample Application as a code-reference.

- Use Java SDKs or Web Services to integrate with your own web application.

The following sections will help you achieve your deployment decision.

Deploying Java SDKs

The following figure illustrates deployment of AuthMinder Server and Java SDKs on a single system:

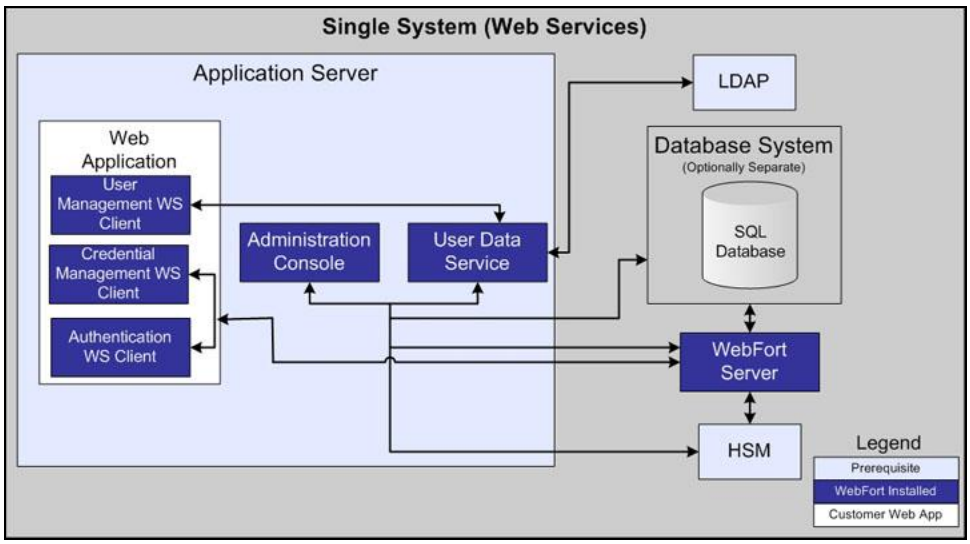


Note: The use of a web server to deliver HTML pages for the application server is optional and is transparent to AuthMinder. In production deployments, this approach is generally used to improve Application Server performance and security. See the documentation of your Application Server for detailed information.

Deploying Web Services

If you plan to deploy Web Services, then the following figure illustrates AuthMinder Server and Web Services on a single system.

Note: Because all web wervices are now built into the AuthMinder Server itself, you just need to install the AuthMinder Server on the target system and generate the requisite client stubs. No further configuration is required.



Distributed Systems Deployment

In a distributed-system deployment, you install the Strong Authentication components on different systems. This type of deployment increases the security and performance. This model is typically used for production deployments or staging environments.

The most common deployment is to install the Server on one system and one or more web applications on additional systems. To deploy Strong Authentication on distributed systems, select the *Custom* option during installation. See chapter, "Deploying Strong Authentication on Distributed Systems" (see page 100) for more information about the installation and post-installation steps.

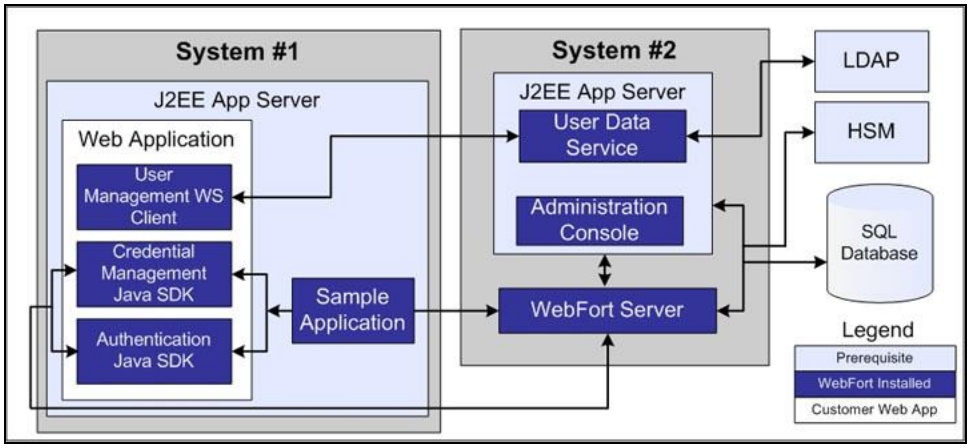
Component Diagrams

The diagrams in this section depict several possible options, where prerequisites and Strong Authenticationcomponents you can install on multiple systems:

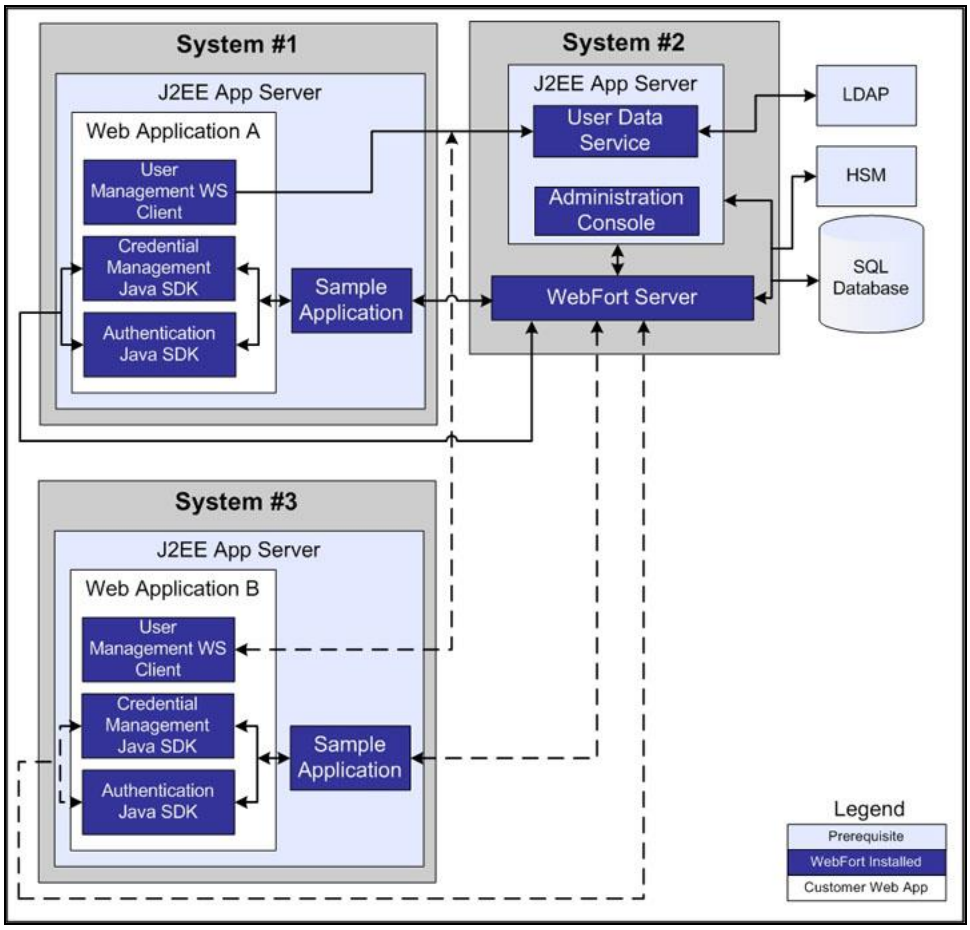
- Deploying Single Application with Java SDKs
- Deploying Multiple Applications with Java SDKs
- Deploying Single Application with Web Services
- Which Strong Authentication components will be installed on each system?

The following sections will help you achieve your deployment decision.

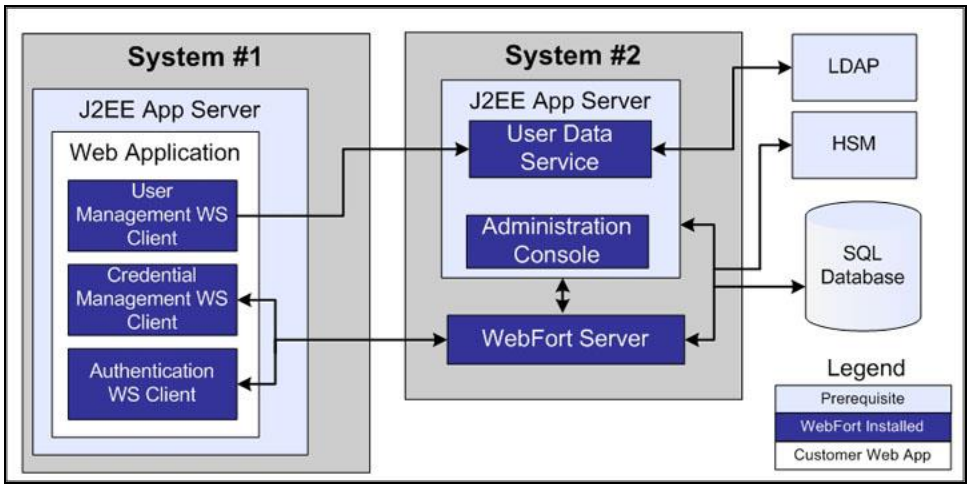
The following figure illustrates deployment of Strong Authentication using Java SDKs on a single application:



The following figure illustrates Strong Authentication deployment using Java SDK with multiple applications:



The following figure illustrates Strong Authentication deployment using Web Services on a single application:



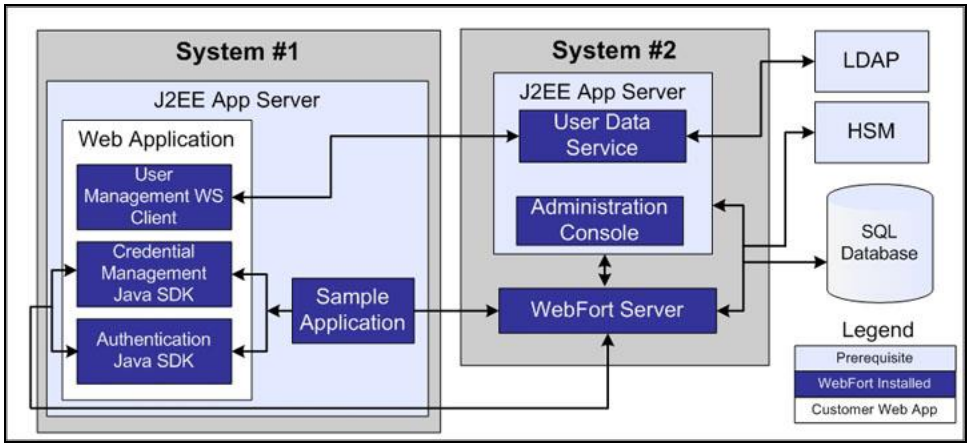
Decision Point

- Which AuthMinder components will be installed on each system?

The following sections will help you achieve your deployment decision.

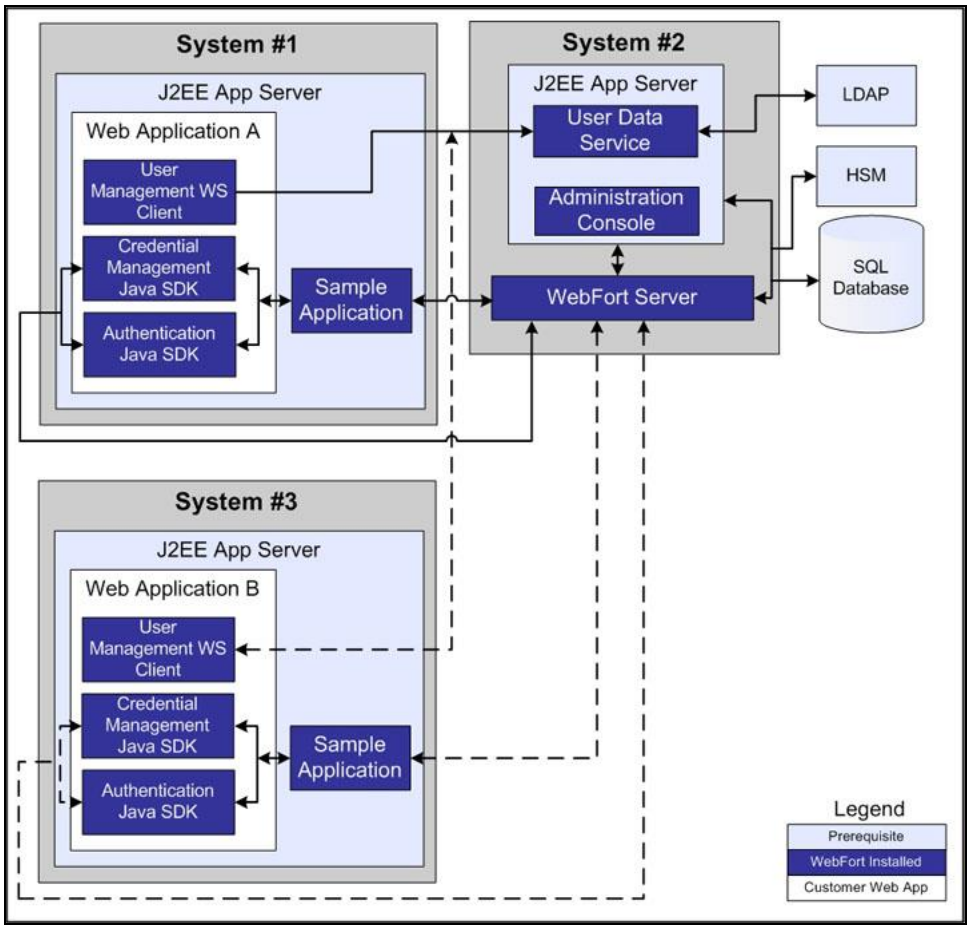
Deploying Single Application with Java SDKs

The following figure illustrates deployment of AuthMinder using Java SDKs on a single application:



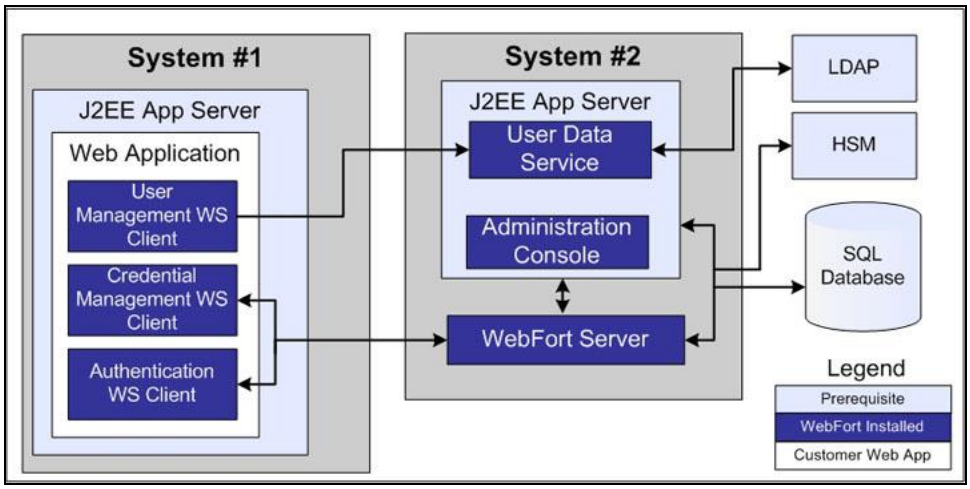
Deploying Multiple Applications with Java SDKs

The following figure illustrates AuthMinder deployment using Java SDK with multiple applications:



Deploying Single Application with Web Services

The following figure illustrates AuthMinder deployment using Web Services on a single application:



High-Availability Deployment

In a high-availability deployment, Strong Authentication components are installed on more than one server to provide high availability and scalability.

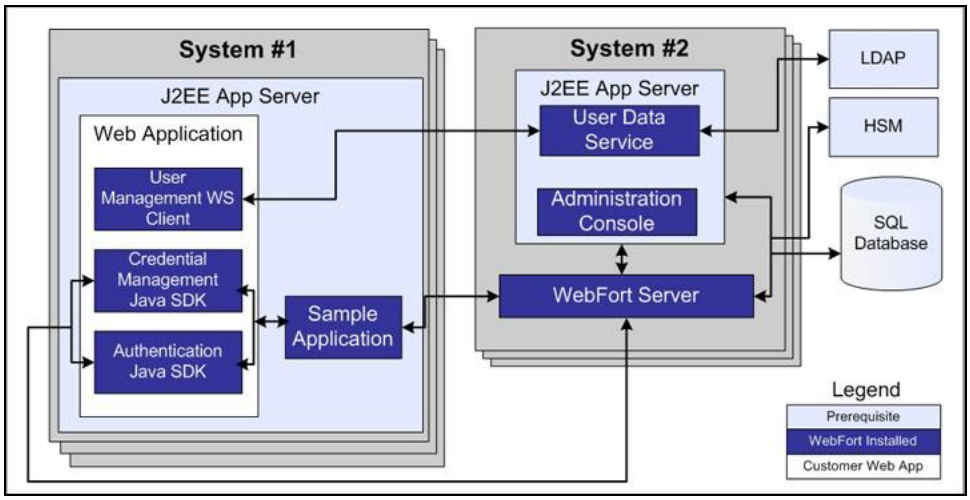
The diagrams in this section depict several possible options for which prerequisites and Strong Authentication components can be installed on multiple systems for a high-availability deployment.

- When do I add a new server instance?

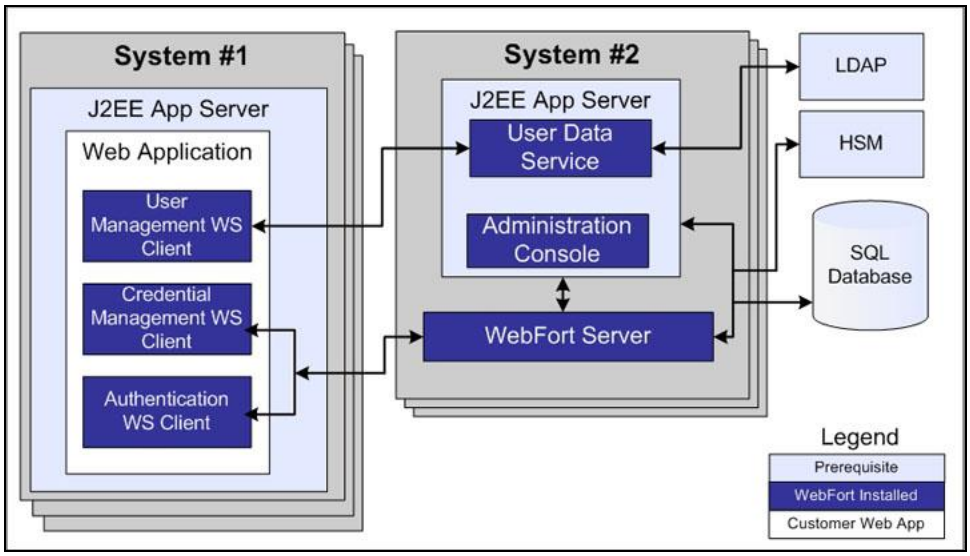
  Typically, when your transaction rate exceeds the allowable threshold (as decided by your organizational policies), then you need to add a new server instance.

- How many servers, Administration Console, UDS, and SDK instances can I have?
  - **Strong Authentication Servers:** Multiple instances are supported. The number depends on the transaction rate you want to achieve.
  - **Administration Consoles**: Multiple instances are supported. The number depends on the number of administrators in the system who log in to the Console at the same time.
  - **UDS Servers:** Currently, *only* one is supported. But if you need multiple instances of UDS, put them behind a Load Balancer. However, UDS failover is not supported.
  - **SDKs:** Multiple instances are supported. This number depends on the number of your application instances that you plan to support.

The following figure illustrates multiple-instance deployment of Strong Authentication using Java SDK.



The following figure illustrates multiple-instance deployment of Strong Authentication using Web Services.



Component Diagrams

The diagrams in this section depict several possible options for which prerequisites and AuthMinder components can be installed on multiple systems for a high-availability deployment.

Decision Points

- When do I add a new server instance?

  Typically, when your transaction rate exceeds the allowable threshold (as decided by your organizational policies), then you need to add a new server instance.

- How many AuthMinder Server, Administration Console, UDS, and SDK instances can I have?

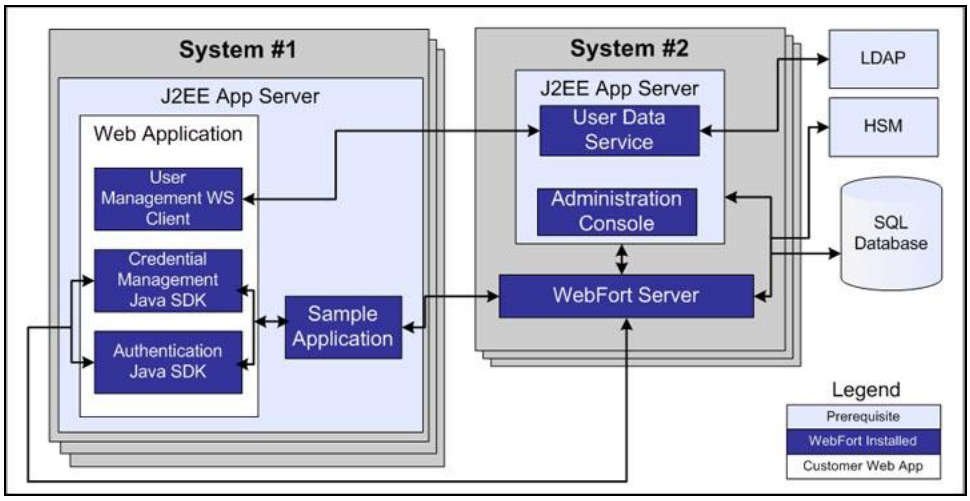
  - **AuthMinder Servers:** Multiple instances are supported. The number depends on the transaction rate you want to achieve.
  - Administration Consoles: Multiple instances are supported. The number depends on the number of administrators in the system who log in to the Console at the same time.
  - **UDS Servers:** Currently, *only* one is supported. But if you need multiple instances of UDS, put them behind a Load Balancer. However, UDS failover is not supported.
  - **SDKs:** Multiple instances are supported. This number depends on the number of your application instances that you plan to support.

The following sections will help you achieve your deployment decision.

- High-Availability Deployment Using Java SDK
- High-Availability Deployment Using Web Services

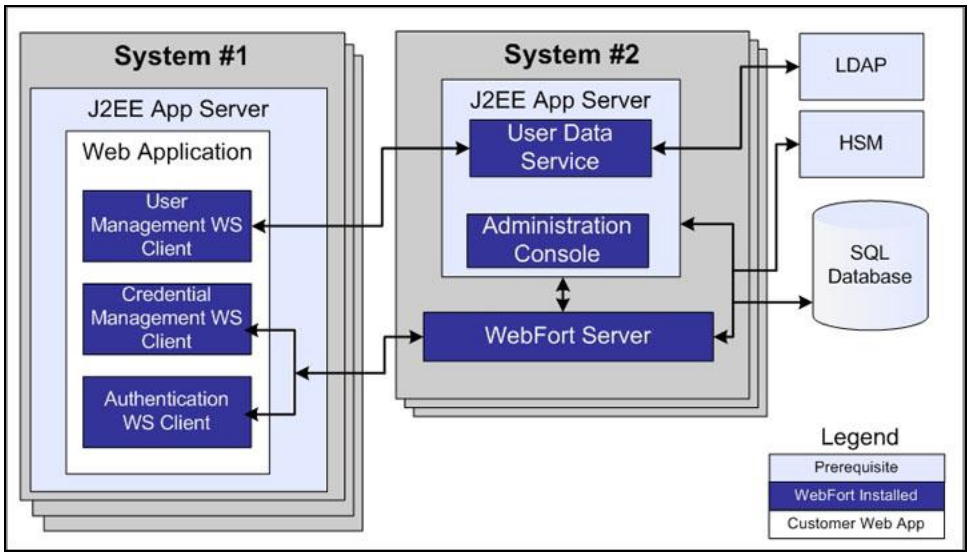
High-Availability Deployment Using Java SDK

The following figure illustrates multiple-instance deployment of AuthMinder using Java SDK.



High-Availability Deployment Using Web Services

The following figure illustrates multiple-instance deployment of AuthMinder using Web Services.



Chapter 3: Preparing for Installation

Before installing AuthMinder Server and its components, ensure that your computer meets the requirements that are described in this chapter. The chapter also provides configuration and planning-related information.

This chapter contains the following sections:

Hardware Requirements

The minimum hardware requirements for installing include:

- Requirements for Strong Authentication and Risk Authentication with database on a single system:
  - RAM: 2 GB
  - Hard Drive Space: 10 GB
  - Processor: 2.4 GHz
- Requirements for Strong Authentication and Risk Authentication with database on a separate system:
  - RAM: 1 GB
  - Hard Drive Space: 300 MB
  - Processor: 2.4 GHz

Note: Hardware resource requirements vary substantially for different applications and usage patterns. it is recommended that you load-test your site to determine the optimal memory that is required for the installation. While load-testing, keep in mind that some operating system utilities for monitoring memory can overstate memory usage (partially because of the representation of shared memory.) The preferred method for determining memory requirements is by monitoring the improvement in performance after adding more RAM/physical memory in the load test. See your platform vendor documentation for information about how to configure memory and processor resources for testing purposes.

Hardware Security Module (HSM) Requirements

You can now store sensitive keys either in the database or in an HSM. Currently, you can store the various encryption keys and the Strong Authentication Server listener SSL key in the HSM. The following table lists the requirements for the supported HSM modules.

HSM Module	Java Cryptography Extension (JCE)	PKCS #11
Thales nCipher netHSM (or nCipher netHSM)	JCE framework provided with the 32-bit versions of JDK 5.0, JDK 6.0, and JDK 7.0	pkcs11v2.01
SafeNet High Availability HSM (or Luna HSM)		

Note: The decision to use and configure an HSM, if required, must be made while you are still in the planning and preparation stages. Otherwise, you will need to re-initialize the database later, because all your current encryption would use keys in software.

Software Requirements

See the Product Support Matrix for a list of software requirements.

Minimum Software Requirements

Ensure that the software requirements listed in one of the following sections are addressed:

- For Solaris SPARC (see page 39)
- For Red Hat Enterprise Linux (see page 43)

For UNIX

The following table lists the minimum software requirements for installing Strong Authentication and Risk Authentication on Solaris.

Note: For all third-party software mentioned in the following table, it is assumed that the higher versions are compatible with the specified supported version.

Software Type	Version
Operating System	Solaris 10
Patches	Latest patches Access latest patches at *http://sunsolve.sun.com*. Click the **Patches and Updates** link, click the **Patch Cluster & Patch Bundle Downloads** link, and under **Solaris Patch Clusters** expand the **Recommended Patch Clusters** to display the Solaris 10 SPARC 05/08 Patch Bundle entries.
Database Server	■ Microsoft SQL Server 2005, Standard Edition (SP2) or higher ■ Microsoft SQL Server 2008 Enterprise Edition
	■ Oracle Database 10*g* ■ Oracle Database 11*g* or higher

Software Type	Version
	■ IBM DB2 UDB 9.5 ■ IBM DB2 UDB 9.7
	■ MySQL Enterprise Edition 5.1
JDBC Drivers (JARs) **Important!** It is recommended that the JDBC JAR version is same as or higher than your database server version.	The JDBC driver version that is compatible with your database.
Directory Server	The following Directory Servers are supported: ■ SunOne Directory Server 5.3 ■ SunOne Directory Server 6.1 ■ SunOne Directory Server 6.3 ■ Oracle Directory Server 11g ■ CA Directory Server 12.0 Service Pack 10 ■ Microsoft Windows Active Directory 2003 ■ Microsoft Windows Active Directory 2008

Software Type	Version
Application Server	The following Application Servers are supported: ■ Apache Tomcat 5.5.x (x can be 31 or higher) ■ Apache Tomcat 6.x ■ Apache Tomcat 7.x ■ Oracle WebLogic 10.1.x ■ Oracle WebLogic 11g (WebLogic Server 10.3) ■ IBM WebSphere 6.1.x **Important!** If you are planning to use WebSphere 6.1, then ensure that you apply the **6.1.0.41: WebSphere Application Server V6.1 Fix Pack 41** and **6.1.0.41: Java SDK 1.5 SR12 FP5 Cumulative Fix for WebSphere Application Server**. ■ IBM WebSphere Application Server 7.0 ■ JBoss Application Server 5.1.x

Software Type	Version
	The **JVM Memory** Settings (Heap Size) for the application server must be a minimum of 512 MB or higher to support User Data Service (UDS) deployment. **Note:** If you plan to create organizations in the LDAP repository with a large user base (for example, 100,000 users), then it is recommended that you increase the heap size to at least 1 GB. To set the heap size to 512 MB, use the -Xmx512M JVM memory setting. To set the heap size to 1 GB, use the -Xmx1024M JVM memory setting. *Do not* use the -Xms parameter when you set the JVM memory setting.
JDK **Note:** If you perform a fresh installation of JDK, then include the new path in the JAVA_HOME environment variable, and ensure that the application server uses the same JAVA_HOME. If you fail to do so, then the Administration Console and other JDK-dependent components may fail to start.	The JDK version that is best compatible with the Application Server that you are using: ■ IBM JDK 1.5 or higher ■ IBM JDK 1.6 or higher ■ Oracle JDK 5.0 ■ Oracle JDK 6.0 ■ Oracle JDK 7.0 ■ Oracle JRockit 5.0 or higher ■ Oracle JRockit 6.0 or higher **Important!** *<JROCKIT_HOME>*/jre/bin/ must be included in PATH environment variable. In addition, This change in the PATH variable must be effective before you start the WebLogic application server.
Web Service Clients	The following clients are supported: ■ Axis2 1.5 or higher ■ .NET Framework 4 or higher

Software Type	Version
Browsers	The following Web browsers are supported: ■ Mozilla Firefox 3.0 or higher ■ Apple Safari 5.0 or higher ■ Google Chrome 12 or higher ■ Internet Explorer 7.0 or higher

For Linux

The following table lists the minimum software requirements for installing Strong Authentication or Risk Authentication on Red Hat Enterprise Linux.

Note: For all third-party software mentioned in the following table, it is assumed that the higher versions are compatible with the specified supported version.

Software Type	Version
Operating System	■ Red Hat Enterprise Linux 4.x (x86) (32-bit) ■ Red Hat Enterprise Linux 5.x (x86) (32-bit) ■ Red Hat Enterprise Linux 5.x (x86) (64-bit) ■ Red Hat Enterprise Linux 6.1 (32-bit and 64-bit) ■ Red Hat Enterprise Linux 6.2 (32-bit and 64-bit)
Update	Update 1 and higher
Patches	Latest patches Access the latest patches at *http://www.redhat.com*. Log in to your account, download the latest updates and patches, and apply them as needed.
Database Server	■ Microsoft SQL Server 2005, Standard Edition (SP2) or higher ■ Microsoft SQL Server 2008 Enterprise Edition
	■ Oracle Database 10*g* or higher ■ Oracle Database 11*g* Release 2
	■ IBM DB2 UDB 9.5 ■ IBM DB2 UDB 9.7
	■ MySQL Enterprise Edition 5.1

Software Type	Version
JDBC Drivers (JARs) **Important!** It is recommended that the JDBC JAR version is same as or higher than your database server version.	The JDBC driver version that is compatible with your database.
Directory Server	The following directory servers are supported: ■ SunOne Directory Server 5.3 ■ SunOne Directory Server 6.1 ■ SunOne Directory Server 6.3 ■ Oracle Directory Server 11g ■ CA Directory Server 12.0 Service Pack 10 ■ Microsoft Windows Active Directory 2003 ■ Microsoft Windows Active Directory 2008

Software Type	Version
Application Server	The following Application Servers are supported: ■ Apache Tomcat 5.5.x (x can be 31 or higher) ■ Apache Tomcat 6.x ■ Apache Tomcat 7.x ■ Oracle WebLogic 10.1.x ■ IBM WebSphere 6.1.x **Important!** If you are planning to use WebSphere 6.1, then ensure that you apply the **6.1.0.41: WebSphere Application Server V6.1 Fix Pack 41** and **6.1.0.41: Java SDK 1.5 SR12 FP5 Cumulative Fix for WebSphere Application Server**. ■ IBM WebSphere Application Server 7.0 ■ JBoss Application Server 5.1.x ■ Oracle WebLogic Server 11g (WebLogic Server 10.3)

Software Type	Version
	The **JVM Memory** Settings (Heap Size) for the application server must be a minimum of 512 MB or higher to support User Data Service (UDS) deployment. **Note:** If you plan to create organizations in the LDAP repository with a large user base (for example, 100,000 users), then it is recommended that you increase the heap size to at least 1 GB. To set the heap size to 512 MB, use the -Xmx512M JVM memory setting. To set the heap size to 1 GB, use the -Xmx1024M JVM memory setting. *Do not* use the -Xms parameter when you set the JVM memory setting.
JDK **Note:** If you perform a fresh installation of JDK, then include the new path in the JAVA_HOME environment variable, and ensure that the application server uses the same JAVA_HOME. If you fail to do so, then the Administration Console and other JDK-dependent components may fail to start.	The JDK version that is best compatible with the Application Server that you are using: ■ IBM JDK 1.5 or higher ■ IBM JDK 1.6 or higher ■ Oracle JDK 5.0 or higher ■ Oracle JDK 6.0 or higher ■ Oracle JDK 7.0 ■ Oracle JRockit 5.0 or higher ■ Oracle JRockit 6.0 or higher **Important!** If you are using JRockit, then ensure that *<JROCKIT_HOME>*/jre/bin/ must be included in PATH environment variable. In addition, This change in the PATH variable must be effective before you start the WebLogic application server.
Web Service Clients	The following clients are supported: ■ Axis2 1.5 or higher ■ .NET Framework 4 or higher

Software Type	Version
Browsers	The following Web browsers are supported: ■ Mozilla Firefox 3.0 or higher ■ Apple Safari 5.0 or higher ■ Google Chrome 12 or higher ■ Internet Explorer 7.0 or higher

Strong Authentication Component-Specific Prerequisites

The prerequisite software is determined by the Strong Authentication components that will be installed on a system.

The following table indicates the prerequisite software that is required by each Strong Authentication component:

Component	Prerequisite		
	Database Server	**JDK**	**Application Server**
Strong Authentication Server			
Administration Console		+*	
User Data Service		+*	
Authentication Java SDK		+*	
Credential Management Java SDK		+*	
User Management Web Service		+*	
Authentication Web Service		+*	
Credential Management Web Service		+*	
Sample Application		+*	

The following 32-bit binaries are required for Strong Authentication to function on an RHEL 6.x 64-bit computer. If you are planning to install Strong Authentication, you can use the yum utility to install these binaries.

- ibgfortran-4.4.5-6.el6.i686.rpm
- libstdc++-4.4.5-6.el6.i686.rpm
- libgomp-4.4.5-6.el6.i686.rpm
- libstdc++-devel-4.4.5-6.el6.i686.rpm
- compat-libtermcap-2.0.8-49.el6.i686.rpm
- ncurses-libs-5.7-3.20090208.el6.i686.rpm
- ncurses-5.7-3.20090208.el6.i686.rpm

* The JDK depends on the application server that you use.

Configuring Database Server

Before installing AuthMinder, set up a database that is used for storing user information, server configuration data, audit log data, and other information.

AuthMinder supports a primary database and a backup database that can be used during failover and failback in high-availability deployments. Database connectivity can be configured in either of the following ways:

- Automatically during AuthMinder installation, when the Installer edits the arcotcommon.ini (see page 185) file with the database information you supply.
- By manually:
  - Creating the Data Source Name (DSN)
  - Editing the arcotcommon.ini (see page 185) file.
  - Updating securestore.enc using the dbutil tool

There are specific configuration requirements for each supported database. Use the following information to set up the database server yourself, or provide this information to your database administrator (DBA) when you request a database account.

Important! To protect the database, it is recommended that you ensure that the database server is protected with firewall or other access control mechanisms and is set to the same time-zone as all Arcot products.

- Configuring Microsoft SQL Server (see page 49)
- Configuring Oracle Database (see page 50)
- Configuring IBM DB2 Universal Database (see page 54)
- Configuring MySQL (see page 57)

SQL Server Configuration

This section provides the following configuration information for Microsoft SQL Server.

Important! Verify that Microsoft SQL Server is configured to use the **SQL Server Authentication** authentication method.

Note: See Microsoft SQL Server documentation for detailed information about performing the tasks that are listed in this section.

Use the following criteria to configure the database:

1. The recommended name is arcotdb.
2. The database size must be configured to grow automatically.
3. Create a Database User

Verifying Authentication Mode

Verify that Microsoft SQL Server is configured to use the **SQL Server Authentication** authentication method. AuthMinder will not be able to connect to the database if Microsoft SQL Server is configured for **Windows Only** authentication mode.

Creating a New Database

Use the following criteria to create a database:

1. The recommended name is arcotdb.
2. The database size must be configured to grow automatically.

Create a Database User

Perform the following steps to create a database user:

Follow these steps:

1. In the SQL Server Management Studio, go to *<SQL_Server_Name>*, expand the **Security** folder, and then click **Logins**.

   Note: The *<SQL_Server_Name>* refers to the host name or IP address of the SQL Server where you created your database.

2. Right-click the **Logins** folder, and click **New Logi**n.
3. Enter the Login name. The recommended name is arcotuser.
4. Set the following parameters:
   a. Authentication to **SQL Server Authentication**.
   b. Specify **Password** and **Confirm password** for the login.

      Ensure that you specify other password settings on this page according to your organization's password policies.

   c. **Default database** to the database (arcotdb) you created.
   d. **User Mapping** for the login (in the **Users mapped to this login section**).
   e. **User Mapping** (SQL 2005) for the default database to db_owner (in the **Database role membership for: <db_name>** section).

Oracle Database Configuration

This section provides the configuration information for Oracle Database and Strong Authentication Server.

Note: See the Oracle Database documentation for details about performing the tasks that are listed in the following sections.

Running Strong Authentication on Oracle requires two tablespaces:

- The first tablespace is used for configuration data, audit logs, and user information. This tablespace can be the default user tablespace.

  See "Creating a New Database" **(see page 52)** for creating a database.

- The second tablespace is used to run reports. For high performance, it is recommended that you use a separate tablespace.

If the database user running the script has sufficient permissions to create a tablespace, the database configuration script (arcot-db-config-for-common-8.0.sql) automatically creates the reports tablespace. If the user does not have the required permissions, a DBA can manually create the reports tablespace and delete the section in this script that creates this tablespace.

If a tablespace with that name already exists, then it is removed and re-created.

Important! The parameters for creating the reports tablespace in the arcot-db-config-for-common-8.0.sql database script can be changed according to the DBA's preferences. However, ensure that the tablespace name is ARReports.

Create a Database

Create a database to store information in the UTF-8 character set. This allows Strong Authentication to use international characters including double-byte languages.

Follow these steps:

1. Log in to the Oracle database server as SYS or SYSTEM.
2. Run the following command:
   ```
   Update sys.props$ set value$='UTF8' where
   name='NLS_NCHAR_CHARACTERSET' Or name = 'NLS_CHARACTERSET';
   ```
3. Restart the database, and check whether the character set is configured to UTF-8.
4. Create a user (the recommended name is arcotuser) in the new database arcotdb.
5. Set the quota of user to *at least* 5 to 10 GB for a development or test deployment, which is primarily used for audit logs.

   Note: If the deployment is for production, staging, or other intensive testing, see appendix, "Database Reference" to determine the quota required for a user.
6. Grant the user the following privileges:

   ```
   CREATE TABLE

   CREATE INDEX

   CREATE SEQUENCE

   CREATE PROCEDURE

   CREATE SESSION

   DML PRIVILEGES

    RESOURCE PRIVILEGES

   CONNECT PRIVILEGES

   ALTER TABLE
   ```

   - Additional privileges for upgrade only:

   ```
    ALTER EXTENT PARAMETERS

    CREATE TABLESPACE
   ```

   - Additional privileges for working with reports:

   ```
   UNLIMITED TABLESPACE

   (Optional) DROP TABLESPACE
   ```

Create a Database User

Create a user with the following criteria:

1. Create a user (the recommended name is arcotuser) in the new database arcotdb.
2. Set the quota of user to *at least* 5 to 10 GB for a development or test deployment, which is primarily used for audit logs.

   Note: If the deployment is for production, staging, or other intensive testing, see appendix, "Database Reference" to determine the quota required for a user.
3. Grant the user the following privileges:

   ```
   CREATE TABLE

   CREATE INDEX

   CREATE SEQUENCE

   CREATE PROCEDURE

   CREATE SESSION

   DML PRIVILEGES

    RESOURCE PRIVILEGES

   CONNECT PRIVILEGES

   ALTER TABLE
   ```

   - Additional privileges for upgrade only:

   ```
    ALTER EXTENT PARAMETERS

    CREATE TABLESPACE
   ```

   - Additional privileges for working with reports:

   ```
   UNLIMITED TABLESPACE

   (Optional) DROP TABLESPACE
   ```

IBM DB2 Universal Database Configuration

This section provides the following configuration information for IBM DB2 Universal Database (UDB):

Create a database (the recommended name is arcotdb) to store information in the UTF-8 character set. This allows Strong Authentication to use international characters including double-byte languages.

1. Log in to the IBM DB2 UDB database server.
2. Run the following command to enable UTF-8 support:
   `create db <DB-NAME> using codeset utf-8 territory us;`
3. Set the tablespace page size to **16K**. By default, the tablespace page size is 4K.

   See the vendor documentation for more information about changing the tablespace page size.
4. If your data volume is high, then the default size of the transaction log file may be insufficient. Therefore, it is recommended that you increase the log file size.

   See the vendor documentation for more information about changing the log file size.
5. Ensure that the configuration changes have been applied.

   If you plan to use an alternate schema, then see appendix, "Configuring Alternate Schema for IBM DB2 Universal Database" (see page 231) for more information.
6. Create a user (recommended name is arcotuser), with a schema in the new database arcotdb.
7. Grant the user the following privileges:

   `CREATE TABLE`

   `CREATE INDEX`

   `CREATE SEQUENCE`

   `CREATE PROCEDURE`

   `CREATE SESSION`

   `DML PRIVILEGES`

   `CONNECT PRIVILEGES`

   `ALTER TABLE`

   - Additional privileges for upgrade only:

   `CREATE TABLESPACE (with AUTORESIZE = yes)`

   - Additional privileges for working with reports:

   `DROP TABLESPACE`

Creating a New Database

Create a database (the recommended name is arcotdb) to store information in the UTF-8 character set. This allows AuthMinder to use international characters including double-byte languages.

Perform the following steps to configure the IBM DB2 UDB:

1. Log in to the IBM DB2 UDB database server.
2. Run the following command to enable UTF-8 support:
   `create db <DB-NAME> using codeset utf-8 territory us;`
3. Set the tablespace page size to **16K**. By default, the tablespace page size is 4K.

   See the vendor documentation for more information about changing the tablespace page size.
4. If your data volume is high, then the default size of the transaction log file may be insufficient. Therefore, it is recommended that you increase the log file size.

   See the vendor documentation for more information about changing the log file size.
5. Ensure that the configuration changes have been applied.

If you plan to use an alternate schema, then see appendix, "Configuring Alternate Schema for IBM DB2 Universal Database" (see page 231) for more information.

Creating a Database User

Create a user with the following criteria:

1. Create a user (recommended name is arcotuser), with a schema in the new database arcotdb.
2. Grant the user the following privileges:
    - CREATE TABLE
    - CREATE INDEX
    - CREATE SEQUENCE
    - CREATE PROCEDURE
    - CREATE SESSION
    - DML PRIVILEGES
    - CONNECT PRIVILEGES
    - ALTER TABLE
    - Additional privileges for upgrade only:
        - CREATE TABLESPACE (with AUTORESIZE = yes)
    - Additional privileges for working with reports:
        - DROP TABLESPACE

MySQL Database Configuration

This section provides the following configuration information for MySQL database.

Strong Authentication uses the InnoDB storage engine of MySQL. To check whether this storage engine is supported by your MySQL installation, use the SHOW ENGINES command. If the output of this command shows that InnoDB is not supported, enable support for InnoDB.

Note: For information about the procedure to enable support for InnoDB, see MySQL documentation.

1. Open a MySQL command window.
2. Run the following command to create the database schema:
   `CREATE SCHEMA '<schema-name>' DEFAULT CHARACTER SET utf8;`
3. Run the following command to create the database user:
   `CREATE USER '<user-name>' identified by '<user-password>';`
4. Create a user (recommended name is arcotuser) in the new database arcotdb.
5. Grant the following privileges to the user:

   Objects rights:

   `SELECT`

   `INSERT`

   `UPDATE`

   `DELETE`

   `EXECUTE`

   DDL rights:

   `CREATE`

   `ALTER`

   `CREATE ROUTINE`

   `ALTER ROUTINE`

   `DROP`

   Other rights:

   `GRANT OPTION`

Enabling Support for the InnoDB Transaction Engine

AuthMinder uses the InnoDB storage engine of MySQL. To check whether this storage engine is supported by your MySQL installation, use the SHOW ENGINES command. If the output of this command shows that InnoDB is not supported, enable support for InnoDB.

Note: For information about the procedure to enable support for InnoDB, see MySQL documentation.

Creating a New Database

To create a database:

1. Open a MySQL command window.
2. Run the following command to create the database schema:
   `CREATE SCHEMA '<schema-name>' DEFAULT CHARACTER SET utf8;`
3. Run the following command to create the database user:
   `CREATE USER '<user-name>' identified by '<user-password>';`

Creating a Database User

Create a user with the following criteria:

1. Create a user (recommended name is arcotuser) in the new database arcotdb.
2. Grant the following privileges to the user:
   - Object rights:
     - SELECT
     - INSERT
     - UPDATE
     - DELETE
     - EXECUTE
   - DDL rights:
     - CREATE
     - ALTER
     - CREATE ROUTINE
     - ALTER ROUTINE
     - DROP
   - Other rights:
     - GRANT OPTION

Getting Ready for Installation

Before you proceed with AuthMinder installation, set up the AuthMinder datastore, the Database Client, and gather the database information that is required for use during the installation. In addition, ensure that the JDK and application server that are required by AuthMinder components are installed.

This section discusses the following topics:

Configure UTF-8 Support on Client Systems

Enable UTF-8 support on the systems where you plan to install the Strong Authentication components that communicate with the database. For example, Strong Authentication Server, Administration Console, and User Data Service. This section walk you through the steps to do so.

To enable UTF-8 support on your UNIX platforms, set the following environment variables:

- NLS_LANG=en_US.UTF-8
- LC_CTYPE=en_US.UTF-8

Database Information that You Need for Installing AuthMinder

Perform the tasks that are described in this section on the system where you will install AuthMinder or the system that uses AuthMinder components.

Microsoft SQL Server

Get the following database information from the DBA. You use this information when you install AuthMinder:

1. Name of the DSN to be created by the installer
2. Server
3. Database
4. User Name
5. Password
6. Port Number

For more information about these parameters, see the table for Microsoft SQL Server in Performing Complete Installation (see page 68).

IBM DB2 UDB

Get the following database information from the DBA. You will need this information when you install AuthMinder:

1. Name of the DSN to be created by the installer
2. Server
3. Database
4. Host Name
5. Port Number
6. User Name
7. Password

For more information about these parameters, see the table for IBM DB2 UDB in Performing Complete Installation (see page 68).

Oracle Database

Get the following database information from the DBA. You will need this information when you install AuthMinder:

1. Name of the DSN to be created by the installer
2. Service ID (Instance identifier of the Oracle database)
3. Host Name
4. Port Number
5. User Name
6. Password

For more information about these parameters, see the table for Oracle Database in Performing Complete Installation (see page 68).

MySQL

Get the following database information from the DBA. You will need this information when you install AuthMinder:

1. Name of the DSN to be created by the installer
2. Server
3. Database
4. User Name
5. Password
6. Port Number

For more information about these parameters, see the table for MySQL in Performing Complete Installation (see page 68).

Requirements for Java-Dependent Components

Install the components that are required by Administration Console, Strong Authentication Java SDKs, and Web Services:

- JDK

  Note: If you perform a fresh installation of JDK, then set the JAVA_HOME environment variable. The path variable must point to $JAVA_HOME/**bin**/. And ensure that the application server uses the same JAVA_HOME. If you fail to do so, then the Administration Console and other JDK-dependant components may fail to start.

- Application Server
- UDS

Requirements for HSM

If you are planning to use HSM to store encryption keys, then set up the following before you proceed:

1. HSM Server
2. HSM Client
3. At least one 3DES key in HSM

   Important! Ensure that you have securely written down the labels of these 3DES keys. You use them while encrypting information in the database.

See your platform vendor documentation for detailed information about how to install and configure your HSM Server and Client components and generate the required keys.

Pre-Installation Checklist

It is recommended that you complete this checklist before you proceed with the installation and setup of Strong Authentication.

Note: The items and values in the following checklist are samples. Before you begin the installation procedure, modify this checklist so that it meets the requirements of your operating environment.

Your Information	Example Entry	Your Entry
DATABASE		
Type	Oracle	
Database Name (*MS SQL and DB2 Only*)	arcotdb	
DSN Name	arcotdsn	
Host Name (or Server IP Address)	51.100.25.24	
Port	1521	
Service ID (*Oracle Databases Only*)	oradb1	
Database User	arcotuser	
Database Login Password	password1234!	
Configured Privileges: **Note:** For all CREATE privileges, the corresponding DROP privilege is implied.		
Oracle Database		
CREATE TABLE		
CREATE INDEX		
CREATE SEQUENCE		
CREATE PROCEDURE		
CREATE SESSION		
DML PRIVILEGES		
RESOURCE PRIVILEGES		
CONNECT PRIVILEGES		
ALTER TABLE *(For Upgrade Only)*		
ALTER EXTENT PARAMETERS		

Your Information	Example Entry	Your Entry
CREATE TABLESPACE (*For Reports*)		
UNLIMITED TABLESPACE (*For Reports,* Optional)		
DROP TABLESPACE		
Microsoft SQL Server **Note:** The user performing these actions must belong to the ddladmin role.		
CREATE TABLE		
CREATE INDEX		
CREATE PROCEDURE		
REFERENCES		
DML PRIVILEGES		
CONNECT PRIVILEGES		
ALTER *(For Upgrade Only)*		
IBM DB2 UDB		
CREATE TABLE		
CREATE INDEX		
CREATE SEQUENCE		
CREATE PROCEDURE		
CREATE SESSION		
DML PRIVILEGES		
CONNECT PRIVILEGES		
ALTER TABLE *(For Upgrade Only)*		
CREATE TABLESPACE (with AUTORESIZE = yes) (*For Reports*)		
DROP TABLESPACE		
MySQL		

Your Information	Example Entry	Your Entry
SELECT		
INSERT		
UPDATE		
DELETE		
EXECUTE		
CREATE		
ALTER		
CREATE ROUTINE		
ALTER ROUTINE		
DROP		
GRANT OPTION		
APPLICATION SERVER		
Type	Apache Tomcat 5.5	
Host Name	localhost	
Port	8080	
JDK	1.5.0_10	
DIRECTORY SERVICE (OPTIONAL)		
Host Name	ds.myldap.com	
Port	389	
Schema Name	inetorgperson or user	
Base Distinguished Name	dc=myldap,dc=com	
User Name	cn=admin,cn=Administrators,cn=dscc	
Password	mypassword1234!	
WEB SERVER (OPTIONAL)		
Type	IIS 6	
Host Name	mywebserver.com	
Port	443	

Chapter 4: How to Deploy Strong Authentication on a Single System

The installation wizard guides you through the installation process. This Wizard supports *Complete* and *Custom* installation types. To install and configure Strong Authentication on a single computer, use the **Complete** option when you run the installer.

The following list provides a quick overview of the process:

1. Run the installer to add Strong Authentication components to your file-system and configure them to access your SQL database.

   See "Performing Complete Installation" (see page 68) for install instructions.

2. Run the database scripts to create schema and database tables. Also ensure that the database setup was successful.

   See "Running Database Scripts" (see page 75) and "Verifying the Database Setup" (see page 76) for more information.

3. Prepare the application server to copy the files that web components use.

   See "Preparing Your Application Server" (see page 76) for more information.

4. Deploy Administration Console on the application server and verify the deployment.

   See "Deploying Administration Console" (see page 84) and "Verifying the Administration Console" (see page 86) for more information.

5. Log in to Administration Console as the Master Administrator to initialize Strong Authentication.

   See "Logging In to Administration Console" (see page 86) and "Bootstrapping the System" (see page 87) for more information.

6. Start the Strong Authentication Server and verify if the service start successfully.

   See "Starting Strong Authentication Server" (see page 90) and "Verifying the Installation" (see page 90) for more information.

7. Deploy User Data Service on the application server and verify the deployment.

   See "Deploying User Data Service" (see page 92) for more information.

8. Deploy and use Sample Application to test Strong Authentication configuration.

   Note: Sample Application is automatically installed as a part of Complete installation.

   See "Deploying Sample Application" (see page 94) and "Using Sample Application" (see page 94) for more information.

9. (Optional) To ensure secure communication between the products components, you can configure them to support SSL.

   See "Configuring SSL" in *CA Strong Authentication Administration Guide* for more information.

10. Complete the installation checklist.

    See "Post-Installation Checklist" (see page 100) for more information.

Note the following:

- Verify that the *<install_location> must not contain* any special characters (such as ~ ! @ # $ % ^ & * ( ) _ + = { }[ ] ’ ").
- Currently, you cannot modify or repair Strong Authentication components by using the installer. Uninstall the component, and then re-install it.
- Do not close the installer window, if the installation is in progress. At any time during installation (*especially during the last stages*), if you click the **Cancel** button to abort the installation, then the installer may not remove *all* the directories it has created so far. Manually clean up the installation directory, <*install_location*>/arcot/ and its subdirectories.
- If you run the installer on a system that already contains an instance of an existing $ARCOT_HOME, then:
  - You are *not* prompted for an installation directory.
  - You are *not* prompted for the database setup. The Installer uses the existing database.
  - You are also *not* prompted to set up encryption.

Perform a Complete Installation

To install all component on a single system, use the *Complete* option. *Custom* installation allows you to install only the selected components from the package. This option is recommended for advanced users.

Follow these steps:

1. Log in and navigate to the directory where you extracted the installer.
2. Verify that you have the permission to run the installer. If not, run the following command:
   `chmod a=rx CA-StrongAuthentication-8.0-Linux-Installer.bin`
3. Run the installer as follows:
   `sh CA-StrongAuthentication-8.0-Linux-Installer.bin`
4. Press Enter to continue with the installation.

   The License Agreement appears.
5. Accept the License Agreement:
   a. Enter y to accept the acceptance of License Agreement and to continue with the installation.

   The Choose Installation Location options appear.
6. Perform *one* of the following steps:
   - If the installer detects an existing home directory, it displays the path to that directory. Press Enter to accept this directory path.
   - If the installer does not detect an existing home directory, it displays the default directory path and prompts you to either accept the default path or specify a new path. Press Enter if you want to accept the default path. Alternatively, enter the absolute path of the directory where you want to install the product and press Enter to continue.

     Note: The installation directory name that you specify must not contain spaces.

   The installation types (Complete and Custom) supported by Strong Authentication appear.

7. Enter 1 to select the default (Complete) option and install all components of Strong Authentication, and press Enter to continue.

   The Database Type options appear.

8. Enter the number corresponding to the database, and press Enter to continue:

   - 1 - Microsoft SQL Server
   - 2 - IBM DB2 (UDB)
   - 3 - Oracle Database
   - 4 - MySQL

   The Primary Database Access Configuration options appear.

   Note: Strong Authentication supports Oracle Real Application Clusters (Oracle RAC). To use Oracle RAC , select Oracle Database in this step, perform the next step (Step 9), and then perform the steps in Configuring CA Strong Authentication for Oracle RAC (see page 223).

9. Depending on the database that you use:

   - Define the information that is listed in the following table if you specified Microsoft SQL Server:

Parameter	Description
Primary ODBC DSN	The installer creates an ODBC connection that Strong Authentication uses to connect to the database. The recommended value to enter is arcotdsn.
User Name	The database user name for Strong Authentication to access the database. This name is specified by the database administrator. (MS SQL Server, typically, refers to this as login.) **Note:** It is recommended that you use different user names for the primary and backup DSNs.
Password	The password associated with the User Name you specified in the previous field and which is used by Strong Authentication to access the database. This password is specified by the database administrator.
Server Name	The host name or IP address of the Strong Authentication datastore. ■ Default Instance **Syntax:** *<server_name>* **Example:** demodatabase ■ Named Instance **Syntax**: *<server_name>**<instance_name>* **Example**: demodatabase\instance1

Parameter	Description
Port Number	The port on which the database server listens to the incoming requests. **Note:** Press Enter, if you want to accept the default port.
Database	The name of the MS SQL database instance.

- Define the information that is listed in the following table if you specified IBM DB2 (UDB):

Parameter	Description
Primary ODBC DSN	The installer creates an ODBC connection that Strong Authentication uses to connect to the database. The recommended value to enter is arcotdsn.
User Name	The database user name for Strong Authentication to access the database. This name is specified by the database administrator. **Note:** The User Name for the Primary and Backup DSNs *must* be different.
Password	The password associated with the User Name you specified in the previous field and which is used by Strong Authentication to access the database. This password is specified by the database administrator.
Host Name	The host name or IP address of the Strong Authentication datastore. ■ Default Instance **Syntax**: *<server_name>* **Example**: demodatabase ■ Named Instance **Syntax**: *<server_name>**<instance_name>* **Example**: demodatabase\instance1
Port Number	The port at which the Database listens to the incoming requests. **Note:** Press Enter, if you want to accept the default port.
Database	The name of the IBM DB2 database instance.

- Define the information that is listed in the following table if you specified Oracle Database:

Parameter	Description
Primary ODBC DSN	The installer creates an ODBC connection that Strong Authentication uses to connect to the database. The recommended value to enter is arcotdsn.
User Name	The database user name for Strong Authentication to access the database. This name is specified by the database administrator. **Note:** The User Name for the Primary and Backup DSNs *must* be different.
Password	The password associated with the User Name you specified in the previous field and which is used by Strong Authentication to access the database. This password is specified by the database administrator.
Service ID	The Oracle System Identifier (SID) that refers to the instance of the Oracle database running on the server.
Port Number	The port at which the Database listens to the incoming requests. **Note:** Press Enter, if you want to accept the default port.
Host Name	The host name or IP address of the datastore. **Syntax**: *<server_name>* **Example**: demodatabase

- Define the information that is listed in the following table if you specified MySQL.

Parameter	Description
Primary ODBC DSN	The installer creates an ODBC connection that Strong Authentication uses to connect to the database. The recommended value to enter is arcotdsn.
User Name	The database user name for Strong Authentication to access the database. This name is specified by the database administrator. (MS SQL Server, typically, refers to this as login.) **Note:** The User Name for the Primary and Backup DSNs *must* be different.

Parameter	Description
Password	The password associated with the User Name you specified in the previous field and which is used by Strong Authentication to access the database. This password is specified by the database administrator.
Server Name	The host name or IP address of the Strong Authentication datastore. ■ Default Instance **Syntax**: *<server_name>* **Example**: demodatabase ■ Named Instance **Syntax**: *<server_name>**<instance_name>* **Example**: demodatabase\instance1
Port Number	The port on which the database server listens to the incoming requests. **Note:** Press Enter, if you want to accept the default port.
Database	The name of the MS SQL database instance.

The Backup Database Access Configuration options appear.

1. Perform *one* of the following steps:

   - Type **N** to skip the configuration of the secondary DSN, when prompted, and press Enter to continue.
   - Type **Y** to configure the secondary DSN, when prompted, and press Enter to continue.

     For information about the parameters, see the table in the preceding step for the database that you are using.

   The Encryption Configuration options appear. Use these options to select the encryption mode and set the information used for encryption.

2. Specify the following information:

   - **Master Key**: Enter the master key that will be used to encrypt the data stored in the database. By default, the value of master key is set to MasterKey. This key is stored in the securestore.enc file, which is located at *<install_location>*/arcot/conf.

     If you want to change the value of the master key after the installation, regenerate the securestore.enc file with a new master key. For more information, see the *CA Strong Authentication Administration Guide*.

a. Enter y if you want to use a Hardware Security Module (HSM) to encrypt the sensitive data. Enter **n** if you want to use the software encryption, in this case you do not have to provide the following HSM information.

   Note: The following options appear *only* if you choose to use HSM.

b. Enter 1 if you want to use Luna HSM or 2 if you want to use nCipher netHSM.

   - **HSM PIN**: Enter the password that is used to connect to the HSM.
   - **Shared Library**: The absolute path to the PKCS#11 shared library corresponding to the HSM.

   Note: Enter the absolute path of the libcknfast.so file.

   - **Storage Slot Number**: The HSM slot where the 3DES keys used for encrypting the data are present. The default value for Luna is 0, and for nCipher netHSM the default value is 1.

   Note: The HSM parameter values are recorded in the arcotcommon.ini file located at <*install_location*>/arcot/conf. To change these values after installation, edit the arcotcommon.ini file, as discussed in appendix, "Configuration Files and Options" (see page 185).

3. Press Enter to continue.

   The Pre-Installation Summary appears.

4. Review the product details displayed carefully and press Enter to proceed with the installation.

   The Installing message appears. This may take several minutes

   After the preceding tasks are completed successfully, the Installation Complete message appears.

5. Press Enter to exit the installer.

   You may have to wait for a few minutes (for the installer to clean up temporary files) until the prompt reappears.

6. Verify that UTF-8 support is enabled:

   a. Navigate to the <*install_location*>/arcot/odbc32v70wf/odbc.ini file.

   b. Locate the [ODBC] section.

   c. Ensure that the IANAAppCodePage=106 entry is present in the section.

   d. If you do not find this entry, then add it.

   e. Save and close the file.

Post-Installation Tasks

This section guides you through the post-installation tasks that you perform after you install Strong Authentication.

1. Running Database Scripts (see page 75)
2. Verifying the Database Setup (see page 76)
3. Preparing Your Application Server (see page 76)
4. Deploying Administration Console (see page 84)
5. Verifying the Administration Console (see page 86)
6. Logging In to Administration Console (see page 86)
7. Bootstrapping the System (see page 87)
8. Starting Strong Authentication Server (see page 90)
9. Verifying the Installation (see page 90)
10. Deploying User Data Service (see page 92)
11. Deploying Sample Application (see page 94)
12. Using Sample Application (see page 94)

Note: After completing these post-installation tasks, perform the Java SDKs and Web Services configuration as discussed in chapter, "Configuring Strong Authentication Java SDKs and Web Services" (see page 139).

Run the Database Script

Strong Authentication includes with database scripts that create the schema and set initial configuration values in the Strong Authentication database.

Follow these steps:

1. Locate the folder with the scripts for your database type. The default location is:

   (for Microsoft SQL Server) <*install_location*>/arcot/dbscripts/mssql

   (for Oracle Database) <*install_location*>/arcot/dbscripts/oracle

   (for IBM DB2 UDB) <*install_location*>/arcot/dbscripts/db2

   (for MySQL) <*install_location*>/arcot/dbscripts/mysql

2. Run the *scripts in the following order* by using the database vendor tools:

   a. arcot-db-config-for-common-8.0.sql

      Important! If you have installed Risk Authentication 8.0, then do not run arcot-db-config-for-common-8.0.sql because you have already run it while installing Risk Authentication 8.0.

   b. arcot-db-config-for-webfort-8.0.sql

Note: If you encounter any errors while executing the scripts, then check with your database administrator whether you have the required privileges.

Verify the Database Setup

After you run the database scripts, use the arwfutil tool verify that the schema was seeded correctly:

Follow these steps:

1. Open a command prompt window.
2. Navigate to the following location:
   `<install_location>/arcot/sbin`
3. At the command prompt enter the following command:
   `./arwfutil vdb`

   The preceding command creates the arcotwebfort-vdb-*<dd>*-*<mmm>*-*<yy>*.txt file in the *<install_location>*/arcot/logs directory.
4. Open the arcotwebfort-vdb-*<dd>*-*<mmm>*-*<yy>*.txt file in a text editor, and check for entries of the following type:
   `ARWF*  FOUND`

   These lines indicate that your database setup has been successful.
5. Close the arcotwebfort-vdb-*<dd>*-*<mmm>*-*<yy>*.txt file.

How to Prepare the Application Server

Two components of Strong Authentication, User Data Service (UDS) and Administration Console, are web-based and support the following supported application servers:

- Apache Tomcat
- IBM WebSphere Application Server
- Oracle WebLogic Server
- JBoss Application Server

Before you deploy the UDS and Administration Console WAR files on the application server, copy the Strong Authentication files and JDBC JAR file to the appropriate location on your application server.

- Step 1: Setting Java Home (see page 77)
- Step 2: Copying Arcot Files to Your Application Server (see page 78)
- Step 3: Copying JDBC JARs to Your Application Server (see page 81)
- Step 4: (Mandatory for Oracle WebLogic 10.1) Creating Enterprise Archive File (see page 83)

Step 1: Set Java Home

Before you deploy the UDS and Administration Console on the application server verify that you set the JAVA_HOME environment variable. For Apache Tomcat, set JAVA_HOME to the Java home directory corresponding to the JDK that you are using.

In addition, include $JAVA_HOME/bin in the PATH environment variable. If you fail to do so, then Administration Console and other JDK-dependent components may fail to start.

Step 2: Copying Database Access Files to Your Application Server

UDS and Administration Console use the following files to access the AuthMinder database securely:

- libArcotAccessKeyProvider.so available at: `<install_location>/arcot/native/<platform name>/<32bit-or-64bit>/`
- arcot-crypto-util.jar available at: `<install_location>/arcot/java/lib/`

As a result, these files must be copied to the appropriate location on the application server where you have deployed these AuthMinder components. The following subsections provide information about copying these files for:

- Apache Tomcat
- IBM WebSphere Application Server
- Oracle WebLogic Server
- JBoss Application Server

Apache Tomcat

To copy the database access files on Apache Tomcat:

1. Copy the libArcotAccessKeyProvider.so file to the following directory:
   - **For Solaris**: <*JAVA_HOME used by Apache Tomcat*>/jre/bin
   - **For RHEL**: <*JAVA_HOME used by Apache Tomcat*>/jre/bin
2. Copy the arcot-crypto-util.jar file to the following directory: `<JAVA_HOME used by Apache Tomcat>/jre/lib/ext`
3. Set and export the LD_LIBRARY_PATH to the directory where the libArcotAccessKeyProvider.so file is copied.
4. Restart the application server.

   Note: An application server restart is required as part of some of the remaining installation tasks. To minimize the number of times you restart the application server, restart it once after performing the last task that requires a restart.

IBM WebSphere

Note: Depending on the WebSphere version that you are using, the steps that you follow may be different.

To copy the database access files on IBM WebSphere:

1. Log in to IBM WebSphere Administration Console.
2. Click **Environment,** and then click **Shared Libraries**.

a. From the **Scope** drop-down, select a valid visibility scope. The scope must include the target server/node on which the application is deployed.

b. Click **New**.

c. Enter the **Name**, for example, **ArcotJNI**.

d. Specify the **Classpath**. This path must point to the location where the arcot-crypto-util.jar file is present and must also include the file name. For example, <*install_location*>/arcot/java/lib/arcot-crypto-util.jar.

e. Enter the JNI library path. This path must point to the location where the libArcotAccessKeyProvider.so file is present. For example, <*install_location*>/arcot/java/native/linux/<*32bit-or-64bit*>.

f. Click **Apply**.

3. Configure server-level class loaders.

a. Click **Servers**, and then click **Application Servers.**

b. Under **Application Servers**, access the settings page of the server for which the configuration has been performed.

c. Click **Java and Process Management**, and then click **Class Loader**.

d. Click **New**. Select default **Classes loaded with parent class loader first** and click **OK.**

e. Click the auto-generated **Class Loader ID**.

f. In the class loader **Configuration** page, click **Shared Library References**.

g. Click **Add**, select the shared library that you created earlier in this procedure (for example, ArcotJNI), and then click **Apply**.

h. Save the changes.

4. Copy libArcotAccessKeyProvider.so file to the following directory:

- **For Solaris**: <*JAVA_HOME used by IBM WebSphere*>/jre/bin
- **For RHEL**: <*JAVA_HOME used by IBM WebSphere*>/jre/bin

5. Restart the application server.

Note: An application server restart is required as part of some of the remaining installation tasks. To minimize the number of times you restart the application server, restart it once after performing the last task that requires a restart.

Oracle WebLogic

To copy the database access files on Oracle WebLogic:

1. Copy libArcotAccessKeyProvider.so to the following directory:

- **For Solaris**:
  - For Sun JDK: <*JAVA_HOME used by Oracle WebLogic instance*>/jre/bin

- For all other versions of JDK: *<JAVA_HOME used by Oracle WebLogic instance>*/jre/bin
- **For RHEL**: *<JAVA_HOME used by Oracle WebLogic instance>*/jre/bin

2. Set and export the LD_LIBRARY_PATH to the directory where the libArcotAccessKeyProvider.so file is copied.
3. Copy arcot-crypto-util.jar to the *<JAVA_HOME used by Oracle WebLogic instance>*/jre/lib/ext directory.
4. Log in to WebLogic Administration Console.
5. Navigate to **Deployments**.
6. Enable the **Lock and Edit** option.
7. Click **Install** and navigate to the directory that contains the arcot-crypto-util.jar file.
8. Click **Next**.

   The Application Installation Assistant screen appears.
9. Click **Next**.

   The Summary page appears.
10. Click **Finish**.
11. Activate the changes.
12. Restart the application server.

    Note: An application server restart is required as part of some of the remaining installation tasks. To minimize the number of times you restart the application server, restart it once after performing the last task that requires a restart.

JBoss Application Server

To copy the database access files on JBoss Application Server:

1. Copy libArcotAccessKeyProvider.so to the following directory:
   - **For Solaris**: *<JAVA_HOME used by JBoss Application Server instance>*/jre/bin
   - **For RHEL**: *<JAVA_HOME used by JBoss Application Server instance>*/jre/bin
2. Copy arcot-crypto-util.jar to the *<JAVA_HOME used by JBoss Application Server instance>*/jre/lib/ext directory.
3. Restart the application server.

   Note: An application server restart is required as part of some of the remaining installation tasks. To minimize the number of times you restart the application server, restart it once after performing the last task that requires a restart.

Step 3: Copying JDBC JARs to Your Application Server

Administration Console, UDS, and Sample Application, which are the Java-dependent components of AuthMinder need JDBC JAR files to connect to the database. You copy these files to the application server.

Note: Before proceeding with the steps mentioned in the following sections, ensure that you have downloaded the JDBC JAR file. See chapter, "Preparing for Installation" (see page 37) for more information about the supported JDBC JAR files.

The following sections walk you through the steps for copying the JDBC JAR file that is required for your database to one of the following application servers:

- Apache Tomcat
- IBM WebSphere Application Server
- Oracle WebLogic Server
- JBoss Application Server

Apache Tomcat

To copy the JDBC JAR file to Apache Tomcat:

1. Navigate to the location where you have downloaded the JDBC JAR file.
2. Copy the JDBC JAR file and paste it in the following directory:
   - **For Apace Tomcat 5.5.x**: *<TOMCAT-HOME>*\common\lib
   - **For Apace Tomcat 6.x and 7.x**: *<TOMCAT-HOME>*\lib

   Alternatively, add the path that contains the JDBC JAR file to the **Classpath** environment variable.
3. Restart Apache Tomcat.

IBM WebSphere Application Server

To copy the JDBC JAR file to IBM WebSphere Application Server:

1. Log in to IBM WebSphere Administration Console.
2. Click **Environment,** and then click **Shared Libraries**.
   a. From the **Scope** drop-down, select a valid visibility scope. The scope must include the target server/node on which the application is deployed.
   b. Click **New**.
   c. Enter the **Name**, for example, **JDBCJAR**.
   d. Specify the **Classpath**. This path must point to the location where the JDBC JAR file is present and must also include the file name.
   e. Click **Apply** to save the changes.

3. Configure server-level class loaders.

   Note: You can either create a class loader or use the one that you created while performing the procedure described in Step 2: Copying Database Access Files to Your Application Server (see page 78).

   a. Click **Servers**, and then click **Application Servers.**

   b. Under **Application Servers**, access the settings page of the server for which the configuration is performed.

   c. Click **Java and Process Management**, and then click **Class Loader**.

   d. Click **New**. Select default **Classes loaded with parent class loader first** and click **OK.**

   e. Click the auto-generated **Class Loader ID**.

   f. In the class loader **Configuration** page, click **Shared Library References**.

   g. Click **Add**, select **JDBCJAR** and then click **Apply**.

   h. Save the changes.

4. Restart IBM WebSphere.

Oracle WebLogic Server

To copy the JDBC JAR file to Oracle WebLogic Server:

Note: If you are using Oracle Database, then you need *not* perform the procedure that is described in this section because Oracle WebLogic Server supports Oracle Database by default.

1. Copy the JDBC JAR file to the following directory:

   <*JAVA_HOME used by Oracle WebLogic instance*>/jre/lib/ext

2. Log in to WebLogic Administration Console.
3. Navigate to **Deployments**.
4. Enable the **Lock and Edit** option.
5. Click **Install** and navigate to the directory that contains the JDBC JAR file.
6. Click **Next**.

   The Application Installation Assistant screen appears.

7. Click **Next**.

   The Summary page appears.

8. Click **Finish**.
9. Activate the changes.
10. Restart the Oracle WebLogic server.

JBoss Application Server

To copy the JDBC JAR file on JBoss Application Server:

1. Copy the JDBC JAR file to the following location on the JBoss Application Server installation directory:
   `<JAVA_HOME Used by JBoss>/server/default/lib`
2. Restart the application server.

Step 4: Create Enterprise Archive File

Valid on Weblogic 10.1

Strong Authentication provides WAR files to deploy Administration Console and User Data Service. You can change the format of these files to EAR and then deploy the EAR file.

Follow these steps:

1. Open the command prompt window.
2. Navigate to the <*install_location*>/arcot/tools/common/bundlemanager directory.
3. Run the bundlemanager tool to create the EAR file by using the following command:
   `java -jar bundle-manager.jar -ear <filename.ear> -warList <war_file_name>`

   Note: In the preceding command, replace the <*war_file_name*> with arcotadmin.war for generating the EAR file for the Administration Console, and arcotuds.war for generating the EAR file for UDS.

   This command generates the EAR file at <*install_location*>/arcot/java/webapps.

Deploying Administration Console

Note: If you plan to deploy Administration Console on IBM WebSphere 7.0, then see the instructions that are provided in appendix, "Deploying Administration Console on IBM WebSphere 7.0" (see page 253).

You need the file arcotadmin.war to deploy the AuthMinder Administration Console. This file is available at:
<*install_location*>/arcot/java/**webapps**/

Note: To manage AuthMinder Server by using Administration Console, ensure that Administration Console can access the system where AuthMinder Server is installed by its host name.

To deploy Administration Console:

1. Change the working directory to:
   <*install_location*>/arcot/sbin
2. Type source arwfenv and press **Enter** to set the $ARCOT_HOME environment variable.
3. Restart the application server for the changes to take effect.
4. Deploy arcotadmin.war in the appropriate directory on the application server.

   Note: The deployment procedure depends on the application server that you are using. See your application server vendor documentation for detailed instructions. For example, in case of Apache Tomcat, deploy the WAR file at <*APP_SERVER_HOME*>/**webapps**/.
5. **(For JBoss Application Server Only)** Perform the following steps if you have deployed Administration Console on JBoss Application Server:
   a. Copy the Bouncy Castle JAR file (bcprov-jdk15-146.jar) from <*install_location*>/arcot/java/**lib**/ to the following location:
      <*JBOSS_HOME*>/common/lib
   b. Navigate to the following location:
      <*JBOSS_HOME*>/server/default/conf/
   c. Open jboss-log4j.xml file in a text editor.
   d. Add the following log configuration in the <log4j:configuration> section:

```
<appender name="arcotadminlog"
class="org.apache.log4j.RollingFileAppender">
<errorHandler
class="org.jboss.logging.util.OnlyOnceErrorHandler"></errorHandler>
<param name="Threshold" value="INFO"/>
<param name="MaxFileSize" value="10MB"/>
<param name="MaxBackupIndex" value="100"/>
<param name="Encoding" value="UTF-8"/>
<param name="Append" value="true"/>
<param name="File" value="${arcot.home}/logs/arcotadmin.log"/>
```

```
<layout class="org.apache.log4j.PatternLayout">
<param name="ConversionPattern" value="%d{yyyy-MM-dd hh:mm:ss,SSS z} : [%t]
: %-5p : %-5c{3} : %m%n"/>
</layout>
<filter class="org.jboss.logging.filter.TCLMCFilter">
<param name="AcceptOnMatch" value="true"/>
<param name="DeployURL" value="arcotadmin.war"/>
</filter>
<!-- end the filter chain here -->
<filter class="org.apache.log4j.varia.DenyAllFilter"></filter>
</appender>
```

e. Add the following log category:

```
<category name="com.arcot">
<priority value="INFO" />
<appender-ref ref="arcotadminlog"></appender-ref>
</category>
```

f. Add the following category for cryptographic operations:

```
<category name="com.arcot.crypto.impl.NCipherCrypter">
<priority value="FATAL" />
<appender-ref ref="arcotadminlog"></appender-ref>
</category>
```

g. Save and close the file.

h. Take the backup of existing JBoss Application Server logging libraries. These library files are available at:
<*JBOSS_HOME*>/lib

i. Upgrade the JBoss Application Server logging libraries available at <*JBOSS_HOME*>/lib to version 2.1.1. The following table lists the JAR file names and the location from where you can download the files.

File Name	Location
jboss-logging-jdk-2.1.1.GA.jar	*http://repo1.maven.org/maven2/org/jboss/logging/jboss-logging-jdk/2.1.1.GA/*
jboss-logging-spi-2.1.1.GA.jar	*http://repo1.maven.org/maven2/org/jboss/logging/jboss-logging-spi/2.1.1.GA/*
jboss-logging-log4j-2.1.1.GA.jar	*http://repo1.maven.org/maven2/org/jboss/logging/jboss-logging-log4j/2.1.1.GA/*

6. Ensure that the application is restarted.

Verifying the Administration Console

The arcotadmin.log file is used for logging the Administration Console information.

To verify if the Administration Console was deployed successfully:

1. Navigate to the following location:
   `<install_location>/arcot/logs`
2. Open the arcotadmin.log file in any editor and locate the following lines:
   `Arcot Administration Console Configured Successfully.`

   This line indicates that your Administration Console was deployed successfully.

   Note: Check and resolve if there are any FATAL and ERROR messages, and review all the WARNING messages for unexpected conditions.

Log In to Administration Console

When logging in to Administration Console for the first time, use the Master Administrator credentials that are created automatically in the database during installation.

Follow these steps:

1. Start Administration Console in a Web browser window, by using the following URL:

   http://<host>:<app_server_port>/arcotadmin/masteradminlogin.htm

   Note: The host and port information that you specify in the preceding URL must be of the application server where Administration Console is deployed.
2. Log in to Administration Console as a Master Administrator with the default Master Administrator account credentials. The credentials are:
   - **User Name*:*** **masteradmin**
   - **Password*:*** **master1234!**

Bootstrapping the System

Before you can start using the Administration Console to manage Strong Authentication, first perform the following steps to initialize the system:

- Change the default Master Administrator password
- Specify the Global Key label
- Specify the authentication mechanism for the Default organization

Bootstrapping is a wizard-driven process that walks you through these setup tasks. Other administrative links are enabled after you perform these tasks.

Before you proceed with Performing the Bootstrapping Tasks (see page 88), you must learn about the related concept of Default Organization.

Default Organization

When you deploy the Administration Console, an organization is created by default. This organization is referred to as *Default Organization* (**DEFAULTORG**). As a single-organization system, the Default Organization itself can be used without creating any organizations.

Perform Bootstrapping Task

When you first log in to the Administration Console as the Master Administrator (MA), the Summary screen for the Bootstrap wizard screen appears.

Follow these steps:

1. Click **Begin** to start the process.

   The Change Password screen appears.

2. Specify the **Old Password**, **New Password**, **Confirm Password**, and click **Next.**

   The Configure Global Key Label screen appears.

3. Specify the **Global Key Label**, and click **Next**.

   Strong Authentication enables you to use hardware- or software-based encryption for your sensitive data. Irrespective of hardware or software encryption.

   If you use hardware encryption, then this label serves only as a reference (or pointer) to the actual 3DES key stored in the HSM device, and therefore the key label *must* match the HSM key label. However, in the case of software-based encryption, this label acts as the reference to actual software key in database.

   Important! After you complete the bootstrapping process, you *cannot* update this key label.

   The **Storage Type** field indicates whether the encryption key is stored in the database (**Software**) or the HSM (**Hardware**).

   The Configure Default Organization screen appears.

4. Under the **Default Organization Configuration** section, specify the following parameters for the Default Organization:

   - **Display Name:** The descriptive name of the organization. This name appears on all other Administration Console pages and reports.
   - **Administrator Authentication Mechanism:** The mechanism that is used to authenticate administrators belonging to the Default Organization. Administration Console supports three types of authentication methods for the administrators to log in:

     - **Basic**

       If you select this option, then the built-in authentication method that is provided by the Administration Console is used for authenticating the administrators.

     - **LDAP User Password**

       If you select this option, then the administrators are authenticated using their credentials that are stored in the directory service.

     Note: If this mechanism is used for authenticating administrators, then deploy UDS by performing the procedure described in "Deploying User Data Service" (see page 92).

     - **Strong Authentication User Password**

       If you select the **Strong Authentication User Password** option here, then the credentials are issued and authenticated by the Server.

     Note: See *CA Strong Authentication Administration Guide* for more information about how to do this.

5. Under the **Key Label Configuration** section of the Configure Default Organization screen, specify the following:

   - **Use Global Key:** This option is selected by default. Deselect this option if you want to override the Global Key Label you specified in the preceding step and specify a new encryption label.
   - **Key Label:** If you deselected the **Use Global Key** option, then specify the new key label that you want to use for the Default Organization.
   - **Storage Type:** This field indicates whether the encryption key is stored in the database (**Software**) or the HSM (**Hardware**).

6. Click **Finish** to complete the bootstrapping process.

   The Administration Console initialization is completed, as indicated in the Finish screen.

7. Click **Continue** to proceed with other configurations using the Administration Console.

Start Strong Authentication

Perform the following steps to start Strong Authentication Server:

1. Navigate to the following directory:
   `<install_location>/arcot/bin/`
2. Run the following command:
   `./webfortserver start`

   Note: If you want to stop the server, then run the ./webfortserver stop command.

Verifying the Installation

You can verify whether the Strong Authentication Server and the web applications have started successfully by:

- Using Log files (see page 90)
- Using Strong Authenticationserver Utility (see page 91)
- Checking the Ports (see page 91)

Using Log files

Perform the following steps to verify if Strong Authentication Server started correctly:

1. Navigate to the following location:
   `<install_location>/arcot/logs`
2. Open the arcotwebfortstartup.log file in any editor and locate the following lines:
   ```
   INSTANCE_VER.................................: [8.0]
   Strong Authentication Service READY
   ```

   These lines indicate that Strong Authentication Server is installed successfully.

Note: Ensure that the log files do not contain any FATAL and WARNING messages.

Using webfortserver Utility

The webfortserver tool enables you to check the release of Strong Authentication that you have installed. See the *Strong Authentication Administration Guide* for more information about this tool.

Follow these steps:

1. Navigate to the following location:
   `<install_location>/arcot/bin`
2. Run the following command to start the tool in interactive mode:
   `./webfortserver -i`
3. Enter the version number at the prompt.

   The webfort-ver-<dd>-<mmm>-<yy>.txt file is created in the <*install_location*>/arcot/logs folder.
4. Open this file and check for the following lines to ensure that you are using the current version:
   - The version of Strong Authentication library files in the bin section is 8.0.
   - The version of the UDS library file (arwfuds.dll) in the bin section is 2.0.3.
5. Close the file.

Checking the Ports

Perform the following steps to verify whether the Strong Authentication Server is listening to different protocols on the default ports:

1. Navigate to the following location:
   `<install_location>/arcot/logs`
2. Open the arcotwebfortstartup.log file in any editor and search for the protocol names to verify whether they are listening on the correct port, as shown in the following snippet.

```
PROTOCOLNAME : [Administration-WS]
PORTNO : 9745
PROTOCOLID : [Transaction-Native]
PORTNO : 9742
PROTOCOLID : [ServerManagement-WS]
PORTNO : 9743
PROTOCOLID : [Transaction-WS]
PORTNO : 9744
```

Note: See appendix, "Default Port Numbers and URLs" (see page 235) for information about default ports and protocols.

Deploying User Data Service

AuthMinder can access user data either from a relational database (RDBMS) or directly from an LDAP server:

You need the file arcotuds.war to deploy the User Data Service (UDS). This file is available at:
<*install_location*>/arcot/java/**webapps**/

To deploy User Data Service:

1. Install arcotuds.war in the appropriate directory on the application server.

   Note: The deployment procedure depends on the application server that you are using. See your application server vendor documentation for detailed instructions. For example, in the case of Apache Tomcat, deploy the WAR file at <*APP_SERVER_HOME*>/**webapps**/.

2. (**For WebSphere Only**) Configure to reload the UDS class when the application files are updated.

   a. Navigate to **Application** > **Enterprise Applications** and access the UDS settings page.

   b. Under **Class loader order**, select the **Classes loaded with local class loader first (parent last)** option.

   c. Under **WAR class loader policy**, select the **Single class loader for application**.

   d. Click **Apply**.

3. **(For JBoss Application Server Only)** Perform the following steps if you have deployed UDS on JBoss Application Server:

   a. Copy the Bouncy Castle JAR file (bcprov-jdk15-146.jar) from <*install_location*>/arcot/java/**lib**/ to the following location:
   <*JBOSS_HOME*>/common/lib

   b. Navigate to the following location:
   <*JBOSS_HOME*>/server/default/conf/

   c. Open jboss-log4j.xml file in a text editor.

   d. Add the following log configuration in the <log4j:configuration> section:

```
<appender name="arcotudslog" class="org.apache.log4j.RollingFileAppender">
<errorHandler
class="org.jboss.logging.util.OnlyOnceErrorHandler"></errorHandler>
<param name="Threshold" value="INFO"/>
<param name="MaxFileSize" value="10MB"/>
<param name="MaxBackupIndex" value="100"/>
<param name="Encoding" value="UTF-8"/>
<param name="Append" value="true"/>
<param name="File" value="${arcot.home}/logs/arcotuds.log"/>
<layout class="org.apache.log4j.PatternLayout">
```

```
<param name="ConversionPattern" value="%d{yyyy-MM-dd hh:mm:ss,SSS z} : [%t]
: %-5p : %-5c{3}(%L) : %m%n"/>
</layout>
<filter class="org.jboss.logging.filter.TCLMCFilter">
<param name="AcceptOnMatch" value="true"/>
<param name="DeployURL" value="arcotuds.war"/>
</filter>
<!-- end the filter chain here -->
<filter class="org.apache.log4j.varia.DenyAllFilter"></filter>
</appender>
```

e. Add the following line in the com.arcot category that you created in Deploying Administration Console (see page 84):
`<appender-ref ref="arcotudslog"></appender-ref>`

f. Add the following line in the for cryptographic category that you created in Deploying Administration Console (see page 84):
`<appender-ref ref="arcotudslog"></appender-ref>`

g. Save and close the file.

4. Restart the application server.

5. Perform the following steps to verify if UDS started correctly:

   a. Navigate to the following location:
   `<install_location>/arcot/logs`

   b. Open the arcotuds.log file in any editor and locate the following lines:
   `User Data Service (Version: 2.0.3) initialized successfully.`

   This line indicates that UDS was deployed successfully

Note: Check and resolve if there are any FATAL and ERROR messages, and review all the WARNING messages for unexpected conditions.

Deploy the Sample Application

Sample Application can be used to verify if Strong Authentication was installed and configured properly. In addition, it demonstrates:

- The typical Strong Authentication workflows
- The tasks that you can perform using the Strong Authentication Java APIs
- Integration of your application with Strong Authentication

Important! Sample Application must *not* be used in production deployments. It is recommended that you build your own web application by using Sample Application as a code-reference.

Follow these steps:

1. Deploy the sample application war file from the following location:
   `<install_location>/arcot/samples/java`
2. Start Sample Application.
3. Access the Sample Application using the following URL:

   http://<host>:<app_server_port>/sample-application/

Using Sample Application

Sample Application enables you to issue and authenticate the credentials that AuthMinder Supports. You can use the Sample Application to perform these operations to test whether the AuthMinder installation has been successful.

This section covers the following tasks:

- Creating Users (see page 95)
- Setting Up ArcotID PKI Client (see page 96)
- Creating ArcotID PKI Credential (see page 97)
- Downloading ArcotID PKI (see page 98)
- Authenticating Using ArcotID PKI (see page 99)

Creating Users

Note: User creation must be performed either using Administration Console or Web Services.

To create users using Administration Console:

1. Log in to the Administration Console as a Global Administrator (GA) or an Organization Administrator (OA). The URL for the purpose is:

   http://<host>:<app_server_port>/arcotadmin/adminlogin.htm

2. If already not activated, activate the **Manage Users and Administrators** sub-tab under the **Users and Administrators** tab.
3. In the left pane, under **Manage Users and Administrators**, click **Create User** to open the Create User page.
4. On the Create User page:

   a. Enter a unique user name, their organization name, and (optionally) other user information in the **User Details** section.

   b. (Optional) Specify other user information in the corresponding fields on the page.

   c. Select the required **User Status**.

   d. Click **Create User.**

   The "Successfully created the user" message appears if the specified user was successfully added to the database.

5. Return to the AuthMinder Sample Application page.

Setting Up ArcotID PKI Client

Set up the ArcotID PKI Client to communicate with the AuthMinder Server for authenticating users with their ArcotID PKI.

Perform the following steps:

1. Access the Sample Application using the following URL:

   http://<host>:<app_server_port>/webfort-7.1.01-sample-application/

2. In the left pane, click **Setup** > **ArcotID Client** to open ArcotID Client Settings page.
3. In the **Choose ArcotID Client** section, select the type of client to be used for authenticating the ArcotID PKI.
4. In the **Choose ArcotID Download Type** section, select the location where you want to store the ArcotID PKI.
5. In the **Choose Where & When to Obtain the ArcotID Challenge** section, select the mode of obtaining the ArcotID PKI challenge.
6. Click **Select** to save the settings.

   The "The operation was successful" message appears if the ArcotID PKI Client configuration was performed successfully.

Creating ArcotID PKI Credential

To create ArcotID PKI credential for users:

1. Access the Sample Application using the following URL:

   http://<host>:<app_server_port>/webfort-7.1.01-sample-application/

2. In the left pane, click **ArcotID** > **Issuance** > **Create** to open Create ArcotID page.
3. Specify the name of the user you created in the **User Name** field.
4. (Optional) Specify the user's organization in the **Organization** field.
5. Specify the password to be used for authentication in the **ArcotID Password** field.
6. (Optional) Specify the profile to be used for issuing ArcotID in the **Profile Name** field.
7. (Optional) Specify the name-value pairs of the **Unsigned Attributes**. These attributes are set in the unsigned portion of the ArcotID PKI.
8. (Optional) Specify the **Custom Attributes** to be used for creating the ArcotID PKI.
9. (Optional) Specify the **Additional Input** that you want to pass to the AuthMinder Server.
10. (Optional) Pass the following **Transaction Logging Parameters**:
    - In the **Log Level** field, select the logging level.

      Note: For more information about the log levels, see the topic titled "AuthMinder Logging" in *CA AuthMinder Administration Guide*.
    - Select **Enable Trace Logging** if you want to capture flow details.
    - Select **Enable DB Logging** if you want to log the database activities.
    - Select **Enable Sensitive Data Logging** if you want to log the sensitive data.
11. Click **Create** to create the credential.

    The "The operation was successful" message appears if the ArcotID PKI was created successfully for the user.

Downloading ArcotID PKI

To download the ArcotID PKI:

1. Access the Sample Application using the following URL:

   http://<host>:<app_server_port>/webfort-7.1.01-sample-application/

2. In the left pane, click **ArcotID** > **Issuance** > **Download** to open the Download ArcotID page.
3. Specify the name of the user you created in the **User Name** field.
4. (Optional) Specify the user's organization in the **Organization** field.
5. (Optional) Specify the profile that has been used to issue ArcotID PKI in the **Profile Name** field.
6. (Optional) Specify the **Additional Input** that you want to pass to the AuthMinder Server.
7. (Optional) Pass the following **Transaction Logging Parameters**:
   - In the **Log Level** field, select the logging level.

     Note: See the topic titled "AuthMinder Logging" in *CA AuthMinder Administration Guide* for more information about the log levels.
   - Select **Enable Trace Logging** if you want to capture flow details.
   - Select **Enable DB Logging** if you want to log the database activities.
   - Select **Enable Sensitive Data Logging** if you want to log the sensitive data.
8. Click **Download** to download the user's ArcotID PKI.

Authenticating Using ArcotID PKI

To authenticate using the ArcotID PKI:

1. Access the Sample Application using the following URL:

   http://<host>:<app_server_port>/webfort-7.1.01-sample-application/

2. In the left pane, click **ArcotID** > **Authentication** > **Authenticate** to open the ArcotID Authentication page.
3. Specify the name of the user you created in the **User Name** field.
4. Specify the user's organization in the **Organization** field.
5. Specify the user's ArcotID PKI password in the **ArcotID Password** field.
6. If you are using aliases to identify the users, then specify the **Application Context** based on the alias of the user that you want to authenticate.
7. (Optional) Select the **Token Type** to be returned to the user after successful authentication.

   Note: See the chapter, "Authenticating Users" in the *CA AuthMinder Java Developer's Guide* for more information about the token types.
8. (Optional) Specify the **Authentication Policy Name** that is to be used for authenticating users.
9. If you have selected SAML as the token type, then specify the **SAML Policy Name** to be used.
10. (Optional) Specify the **Additional Input** that you want to pass to the AuthMinder Server.
11. (Optional) Pass the following **Transaction Logging Parameters**:
    - In the **Log Level** field, select the logging level.

      Note: See the topic titled, "AuthMinder Logging" in *CA AuthMinder Administration Guide* for more information about the log levels.
    - Select **Enable Trace Logging** if you want to capture flow details.
    - Select **Enable DB Logging** if you want to log the database activities.
    - Select **Enable Sensitive Data Logging** if you want to log the sensitive data.
12. Click **Authenticate** to verify the user's ArcotID PKI.

Performing Additional Configurations for CA Adapter 2.2.7

If you have a CA Adapter 2.2.7 instance and want to use it with AuthMinder 7.1.01, you must perform certain additional configurations. See Additional Configurations for CA Adapter 2.2.7 (see page 227) for the exact steps to be performed.

Post-Installation Checklist

Fill this checklist with installation and setup information for AuthMinder. You will need this information for various administrative tasks that you perform later.

Your Information	Example Entry	Your Entry
ARCOT_HOME	/var/opt/arcot/	
SYSTEM INFORMATION		
Host Name	my-bank	
User Name	administrator	
Password	password1234!	
Configured Components	WebFort Server Administration Console User Data Service	
ADMINISTRATION CONSOLE INFORMATION		
Host Name	localhost	
Port	8080	
Master Administrator Password	mypassword1234!	
USER DATA SERVICE INFORMATION		
Host Name	localhost	
Port	8080	
Application Context Root	arcotuds	

Chapter 5: How to Deploy Strong Authentication on Distributed Systems

Use the InstallAnywhere Wizard to install Strong Authentication components. This Wizard supports *Complete* and *Custom* installation types. However, to install and configure Strong Authentication in a distributed environment, use the *Custom* option when you run the installer.

The following steps provide a quick overview of the process:

1. Run the Strong Authentication installer to install Strong Authentication Server and Administration Console and to configure them to access your SQL database. You can also choose to install the Web Services on the same system.

   See "Installing on the First System" (see page 102) for installation instructions.

2. Execute the database scripts to create Strong Authentication schema and database tables. Also ensure that the database setup was successful.

   See "Running Database Scripts" (see page 112) and "Verifying the Database Setup" (see page 76) for more information.

3. Prepare the application server to copy the files that Web components use.

   See "Preparing Your Application Server" (see page 113) for more information.

4. Deploy Administration Console on the application server and verify the deployments.

   See "Deploying Administration Console" (see page 120) and "Verifying the Administration Console" (see page 86) for more information.

5. Log in to Administration Console with the Master Administrator credentials to initialize Strong Authentication.

   See "Logging In to Administration Console" (see page 86) and "Bootstrapping the System" (see page 87) for more information.

6. Start the Strong Authentication Server and verify if the service start successfully.

   See "Starting Strong Authentication Server" (see page 90) and "Verifying the Installation" (see page 90) for more information.

7. Deploy User Data Service on the application server and verify the deployment.

   See "Deploying User Data Service" (see page 128) for more information.

8. Install the Java SDKs and Web Services on one or more systems.

   See "Installing on the Second System" (see page 75) for more information.

Deploy, configure, and use Sample Application to test configuration.

Note: To install the Sample Application *only*, ensure that you select only the **SDKs and Sample Application** option and proceed with the installation.

See "Deploying Sample Application" (see page 131), "Configuring Sample Application for Communication with Strong Authentication Server" (see page 131), and "Using Sample Application" (see page 132) for more information.

1. (Optional) To verify secure communication between Strong Authentication components, you can configure them to support SSL

   See the topic titled "Configuring SSL" in *Strong Authentication Administration Guide* for more information.

2. Complete the installation checklist.

   See "Post-Installation Checklist " (see page 138) for more information.

Note the following:

- Ensure that the *<install_location> must not contain* any special characters (such as ~ ! @ # $ % ^ & * ( ) _ + = { }[ ] ’ ").
- Currently, you cannot modify or repair Strong Authentication components by using the installer. Uninstall the component, and then re-install it.
- Do not close the installer window, if the installation is in progress. If at any time during installation (*especially during the last stages*), if you click the **Cancel** button to abort the installation, then the installer may not remove *all* the directories it has created so far. Manually clean up the installation directory, <*install_location*>/**arcot**/ and its subdirectories.
- If you run the installer on a system that already contains an instance of an existing $ARCOT_HOME, then:
  - You are *not* prompted for an installation directory.
  - You are *not* prompted for the database setup. The Installer uses the existing database.
  - You are *not* prompted to set up encryption.

Install the First System

Perform the following steps to install Strong Authentication and related components.

Follow these steps:

1. Log in and navigate to the directory where you untarred the installer.
2. Ensure that you have the permission to run the installer. If you do not have the required permission, run the following command:
   `chmod a=rx CA-StrongAuthentication-8.0-Linux-Installer.bin`

3. Run the installer as follows:
   `sh Strong Authentication-<version_number>-<platform_name>-Installer.bin`

   If you are running the installer with root login, then a warning message appears. Enter **Y** to continue, or **N** to quit the installation. If you exit the installer, then run the installer again.

   The Welcome message appears.

4. Press **Enter** to continue with the installation.

   The License Agreement appears.

5. Accept the License Agreement.

   a. Enter **y** to accept the acceptance of License Agreement and to continue with the installation.

   The Choose Installation Location options appear.

6. Perform *one* of the following steps:

   - If the installer detects an existing home directory, then it displays the path to that directory. Press Enter to accept this directory path.
   - If the installer does not detect an existing home directory, it displays the default directory path and prompts you to either accept the default path or specify a new path. Press Enter if you want to accept the default path. Alternatively, enter the absolute path of the directory where you want to install Strong Authentication and press Enter to continue.

     Note: The installation directory name that you specify must not contain spaces.

   The installation types (Complete and Custom) screen appear.

7. Select the required option and press **Enter** to continue with the installation.

8. Type **2** and press **Enter** to accept the **Custom** installation option and to continue with the installation.

   The Choose Product Features options appear.

9. Specify a comma-separated list (*without any space between the comma and the number*) of numbers representing the Strong Authentication components you want to install.

   On the first system, you install the following components:

   a. **Strong Authentication Authentication Server**

   b. **Administration Console**

   c. **User Data Service**

The following table describes all components that are installed by the installer and the numbers that you must enter to install them.

Option	Component	Description
1	Strong Authentication Authentication Server	This option installs the core Processing engine (**Strong Authentication Server**) that serves the following requests from SDKs, Administration Console, and Web Services: ■ Credential Issuance Configurations ■ Credential Authentication Configurations ■ Server Configurations In addition, this component also enables you to access the following Web Services: ■ **Authentication and Authorization Web service -** Provides the programming interface for authenticating and authorizing users. ■ **Issuance SDK and Web Services -** Provides the programming interface for creating, reading, and updating user credential information in the Strong Authentication database. ■ **Authentication Web Service** - Provides the programming interface for authenticating users. ■ **Credential Management Web Service** - Provides the programming interface for creation and management of user credentials. ■ **Administration Web Service** - Provides the programming interface used by the Strong Authentication Administration Console. ■ **Bulk Operations Web Service**: Provides the programming interface for uploading and fetching OATH tokens.

Option	Component	Description
2	Java SDK and WS	This option provides programming interfaces (in form of APIs and Web Services) that can be invoked by your application to forward authentication and user credential issuance requests to the Strong Authentication Server. This package comprises the following sub-components: ■ **Authentication Java SDK and Web Services**- Provides the programming interface for authentication with Strong Authentication Server. ■ **Credential Management Java SDK and Web Services** - Provides the programming interface for creation and management of user credentials. ■ **Administration Web Service** - Provides the programming interface for creating configurations. ■ **Bulk Operations Web Service**: Provides the programming interface for uploading and fetching OATH tokens. See chapter, "Configuring Strong Authentication Java SDKs and Web Services" (see page 139) for more information about configuring these components.
3	Strong Authentication Sample Application	This option provides Web-based interface for demonstrating the use of Java APIs and verify if Strong Authentication was installed successfully, and if it is able to perform credential management and authentication requests.
4	Administration Console	This option provides the Web-based interface for managing Strong Authentication Server and authentication-related configurations.
5	User Data Service	This option installs UDS that acts as an abstraction layer for accessing different types of user repositories, such as relational databases (RDBMSs) and directory servers (LDAPs.)

1. Press **Enter** to continue.

   The database types screen appear.

2. Specify the number corresponding to the database, and press **Enter** to continue:

   - **1** - Microsoft SQL Server
   - **2** - IBM DB2 (UDB)
   - **3** - Oracle Database
   - **4** - MySQL

   The Primary Database Access Configuration options appear.

   Note: Strong Authentication Oracle Real Application Clusters (Oracle RAC). To use Oracle RAC with your Strong Authentication Installation, select Oracle Database in this step, perform the next step (Step 12), and then perform the steps in Configuring Strong Authentication for Oracle RAC (see page 223).

3. Depending on the database that you are using:

   - Specify the information that is listed in the following table if you specified Microsoft SQL Server:

Parameter	Description
Primary ODBC DSN	The installer creates an ODBC connection that Strong Authentication uses to connect to the database. The recommended value to enter is arcotdsn.
User Name	The database user name for Strong Authentication to access the database. This name is specified by the database administrator. (MS SQL Server, typically, refers to this as login.) **Note:** The User Name for the Primary and Backup DSNs *must* be different.
Password	The password associated with the **User Name** you specified in the previous field and which is used by Strong Authentication to access the database. This password is specified by the database administrator.

Parameter	Description
Server Name	The host name or IP address of the datastore. ■ Default Instance **Syntax**: *<server_name>* **Example**: demodatabase ■ Named Instance **Syntax**: *<server_name>**<instance_name>* **Example**: demodatabase\instance1
Port Number	The port on which the database server listens to the incoming requests. **Note:** Press **Enter**, if you want to accept the default port.
Database	The name of the MS SQL database instance.

- Specify the information that is listed in the following table if you specified IBM DB2 (UDB):

Parameter	Description
Primary ODBC DSN	The installer creates an ODBC connection that Strong Authentication uses to connect to the database. The recommended value to enter is arcotdsn.
User Name	The database user name for Strong Authentication to access the database. This name is specified by the database administrator. **Note:** The User Name for the Primary and Backup DSNs *must* be different.
Password	The password associated with the **User Name** you specified in the previous field and which is used by Strong Authentication to access the database. This password is specified by the database administrator.

Parameter	Description
Host Name	The host name or IP address of the datastore. ■ Default Instance **Syntax**: *<server_name>* **Example**: demodatabase ■ Named Instance **Syntax**: *<server_name>**<instance_name>* **Example**: demodatabase\instance1
Port Number	The port at which the Database listens to the incoming requests. **Note:** Press **Enter**, if you want to accept the default port.
Database	The name of the database that Strong Authentication will access.

- Specify the information that is listed in the following table if you specified Oracle Database Server:

Parameter	Description
Primary ODBC DSN	The installer creates an ODBC connection that Strong Authentication uses to connect to the database. The recommended value to enter is arcotdsn.
User Name	The database user name for Strong Authenticationto access the database. This name is specified by the database administrator. **Note:** The User Name for the Primary and Backup DSNs *must* be different.
Password	The password associated with the **User Name** you specified in the previous field and which is used by Strong Authentication to access the database. This password is specified by the database administrator.
Service ID	The Oracle System Identifier (SID) that refers to the instance of the Oracle database running on the server.
Port Number	The port at which the Database listens to the incoming requests. **Note:** Press **Enter**, if you want to accept the default port.

Parameter	Description
Host Name	The host name or IP address of the datastore. **Syntax**: *<server_name>* **Example**: demodatabase

- Specify the information that is listed in the following table if you specified MySQL.

Parameter	Description
Primary ODBC DSN	The installer creates an ODBC connection that Strong Authentication uses to connect to the database. The recommended value to enter is arcotdsn.
User Name	The database user name for Strong Authentication to access the database. This name is specified by the database administrator. (Microsoft SQL Server, typically, refers to this as login.) **Note:** The User Name for the Primary and Backup DSNs *must* be different.
Password	The password associated with the **User Name** you specified in the previous field and which is used by Strong Authentication to access the database. This password is specified by the database administrator.
Server Name	The host name or IP address of the datastore. ■ Default Instance **Syntax**: *<server_name>* **Example**: demodatabase ■ Named Instance **Syntax**: *<server_name>**<instance_name>* **Example**: demodatabase\instance1
Port Number	The port on which the database server listens to the incoming requests. **Note:** Press **Enter**, if you want to accept the default port.
Database	The name of the Microsoft SQL Server database instance.

The Backup Database Access Configuration options appear.

1. Perform *one* of the following steps:

   - Type **N** to skip the configuration of the secondary DSN, when prompted, and press **Enter** to continue.
   - Type **Y** to configure the secondary DSN, when prompted, and press **Enter** to continue.

     For information about the tasks to be performed, see the table in the preceding step.

   The Encryption Configuration options appear.

2. Specify the following information:

   - **Master Key**: Enter the master key, which is used to encrypt the data stored in the database. By default, the value of master key is set to MasterKey. This key is stored in the securestore.enc file, which is located at *<install_location>/*arcot/conf.

     If you want to change the value of the master key after the installation, then regenerate the securestore.enc file with a new master key. For more information, see the *Strong Authentication Administration Guide*.

   - Enter **y** if you want to use a Hardware Security Module (HSM) to encrypt the sensitive data. Enter **n** if you want to use the software encryption, in this case you do not have to provide the following HSM information.

   a. Enter **1** if you want to use Luna HSM or **2** if you want to use nCipher netHSM.

   b. **HSM PIN**: Enter the password that is used to connect to the HSM.

   - **Shared Library**: The absolute path to the PKCS#11 shared library corresponding to the HSM.

     For Luna (libCryptoki2.so) and for nCipher netHSM (libcknfast.so), enter the absolute path and full name of the file.

   - **Storage Slot Number**: The HSM slot where the 3DES keys used for encrypting the data are present. The default value for Luna is 0, and for nCipher netHSM the default value is 1.

     Note: The HSM parameter values are recorded in the arcotcommon.ini file located at *<install_location>*/arcot/conf. To change these values after installation, edit the arcotcommon.ini file, as discussed in appendix, "Configuration Files and Options" (see page 185).

3. Press **Enter** to continue.

   The Pre-Installation Summary appears.

4. Review the product details displayed carefully and press **Enter** to proceed.

   After the preceding tasks are completed successfully, the Installation Complete message appears.

5. Press **Enter** to exit the installer.

   You may have to wait for a few minutes (for the installer to clean up temporary files) until the prompt reappears.

6. Verify that UTF-8 support is enabled:

   a. Navigate to the *<install_location>*/arcot/odbc32v70wf/odbc.ini file.

   b. Locate the [ODBC] section.

   c. Ensure that the IANAAppCodePage=106 entry is present in the section.

   d. If you do not find this entry, then add it.

   e. Save and close the file.

Post-Installation Tasks on the First System

The following post-installation steps are discussed in this section:

1. Running Database Scripts (see page 112)
2. Verifying the Database Setup (see page 76)
3. Preparing Your Application Server (see page 113)
4. Deploying Administration Console (see page 120)
5. Verifying the Administration Console (see page 86)
6. Logging In to Administration Console (see page 86)
7. Bootstrapping the System (see page 87)
8. Starting Strong Authentication Server (see page 125)
9. Verifying the Installation (see page 90)
10. Deploying User Data Service (see page 128)

Note: After completing these post-installation tasks, perform the Java SDKs and Web Services configuration that is discussed in chapter, "Configuring Strong Authentication Java SDKs and Web Services" (see page 139).

Create the Database Schema

Strong Authentication comes with SQL scripts that create the schema and set initial configuration values in the Strong Authentication database.

Follow these steps:

1. Locate the folder with the scripts for your database type. The default location is:

   (For Microsoft SQL Server) <*install_location*>/arcot/dbscripts/mssql

   (For Oracle Database) <*install_location*>/arcot/dbscripts/oracle

   (For IBM DB2 UDB) <*install_location*>/arcot/dbscripts/db2

   (For MySQL) <*install_location*>/arcot/dbscripts/mysql

2. Run the *scripts in the following order* by using the database vendor tools:

   a. arcot-db-config-for-common-8.0.sql

      Important! If you have installed Risk Authentication 8.0, do not run arcot-db-config-for-common-8.0.sql because you have already run it while installing Risk Authentication 8.0.

   b. arcot-db-config-for-webfort-8.0.sql

Note: If you encounter any errors while running the scripts, then check whether you have the required privileges.

Verify the Database Setup

After running the required database scripts, verify that the schema was seeded correctly by using the arwfutil tool.

Follow these steps:

1. Open a command prompt window.
2. Navigate to the following location:
   `<install_location>/arcot/sbin`
3. At the command prompt enter the following command:
   `./arwfutil vdb`

   The preceding command creates the arcotwebfort-vdb-<*dd*>-<*mmm*>-<*yy*>.txt file in the <*install_location*>/arcot/logs directory.
4. Open the arcotwebfort-vdb-<*dd*>-<*mmm*>-<*yy*>.txt file in a text editor, and check for entries of the following type:
   `ARWF*  FOUND`

   These lines indicate that your database setup has been successful.
5. Close the arcotwebfort-vdb-<*dd*>-<*mmm*>-<*yy*>.txt file.

How to Prepare the Application Server

User Data Service (UDS) and Administration Console are web based application that support the following application servers:

- Apache Tomcat
- IBM WebSphere Application Server
- Oracle WebLogic Server
- JBoss Application Server

Before you deploy the UDS and Administration Console WAR files on the application server, copy the Strong Authentication files and JDBC JAR file to the appropriate location on your application server.

- Step 1: Setting Java Home (see page 113)
- Step 2: Copying Database Access Files to Your Application Server (see page 114)
- Step 3: Copying JDBC JARs to Your Application Server
- Step 4: (Mandatory for Oracle WebLogic 10.1) Creating Enterprise Archive File (see page 119)

Set Java Home

Before you deploy the WAR files for UDS and Administration Console on the application server of your choice, ensure that you set the JAVA_HOME environment variable. For Apache Tomcat, set JAVA_HOME to the Java home directory corresponding to the JDK that you are using.

In addition, include $JAVA_HOME/bin in the PATH environment variable. If you fail to do so, then Administration Console and other JDK-dependent components may fail to start.

Step 2: Copying Database Access Files to Your Application Server

UDS and Administration Console use the following files to access the AuthMinder database securely:

- libArcotAccessKeyProvider.so available at: `<install_location>/arcot/native/<platform name>/<32bit-or-64bit>/`
- arcot-crypto-util.jar available at: `<install_location>/arcot/java/lib/`

As a result, these files must be copied to the appropriate location on the application server where you have deployed these AuthMinder components. The following subsections provide information about copying these files for:

- Apache Tomcat
- IBM WebSphere Server
- Oracle WebLogic Server
- JBoss Application Server

Apache Tomcat

To copy the database access files on Apache Tomcat:

1. Copy the libArcotAccessKeyProvider.so file to the following directory:
   - **For Solaris**: <*JAVA_HOME used by Apache Tomcat*>/jre/bin
   - **For RHEL**: <*JAVA_HOME used by Apache Tomcat*>/jre/bin
2. Copy the arcot-crypto-util.jar file to the <*JAVA_HOME used by Apache Tomcat*>/jre/lib/ext directory.
3. Set and export the LD_LIBRARY_PATH to the directory where the libArcotAccessKeyProvider.so file is copied.
4. Restart the application server.

   Note: An application server restart is required as part of some of the remaining installation tasks. To minimize the number of times you restart the application server, restart it once after performing the last task that requires a restart.

IBM WebSphere

Note: Depending on the WebSphere version that you are using, the steps that you follow may be different.

To copy the database access files on WebSphere 6.1:

1. Log in to WebSphere Administration Console.
2. Click Environment, and then click Shared Libraries.

a. From the Scope drop-down, select a valid visibility scope. The scope must include the target server/node on which the application is deployed.

b. Click New.

c. Enter the Name, for example, ArcotJNI.

d. Specify the Classpath. This path must point to the location where the arcot-crypto-util.jar file is present and must also include the file name. For example, <install_location>/arcot/java/lib/arcot-crypto-util.jar.

e. Enter the JNI library path. This path must point to the location where the libArcotAccessKeyProvider.so file is present. For example, the *<install_location>*/arcot/java/native/linux/*<32bit-or-64bit>* directory.

f. Click Apply.

3. Configure server-level class loaders.

a. Click Servers, and then click Application Servers.

b. Under Application Servers, access the settings page of the server for which the configuration has been performed.

c. Click Java and Process Management, and then click Class Loader.

d. Click New. Select default Classes loaded with parent class loader first and click OK.

e. Click the auto-generated Class Loader ID.

f. In the class loader Configuration page, click Shared Library References.

g. Click Add, select the shared library that you created earlier in this procedure (for example, ArcotJNI), and then click Apply.

h. Save the changes.

4. Copy libArcotAccessKeyProvider.so file to the following directory:

- **For Solaris**: <*JAVA_HOME used by IBM WebSphere*>/jre/bin
- **For RHEL**: <*JAVA_HOME used by IBM WebSphere*>/jre/bin

5. Restart the application server.

Note: An application server restart is required as part of some of the remaining installation tasks. To minimize the number of times you restart the application server, restart it once after performing the last task that requires a restart.

Oracle WebLogic

To copy the database access files on the Oracle WebLogic server:

1. Copy libArcotAccessKeyProvider.so to the following directory:

- **For Solaris**: <*JAVA_HOME used by Oracle WebLogic instance*>/jre/bin
- **For RHEL**: <*JAVA_HOME used by Oracle WebLogic instance*>/jre/bin

2. Set and export the LD_LIBRARY_PATH to the directory where the libArcotAccessKeyProvider.so file is copied.
3. Copy arcot-crypto-util.jar to the <*JAVA_HOME used by Oracle WebLogic instance*>/jre/lib/ext directory.
4. Log in to WebLogic Administration Console.
5. Navigate to **Deployments**.
6. Enable the **Lock and Edit** option.
7. Click **Install** and navigate to the directory that contains the arcot-crypto-util.jar file.
8. Click **Next**.

   The Application Installation Assistant screen appears.
9. Click **Next**.

   The Summary page appears.
10. Click **Finish**.
11. Activate the changes.
12. Restart the application server.

    Note: An application server restart is required as part of some of the remaining installation tasks. To minimize the number of times you restart the application server, restart it once after performing the last task that requires a restart.

JBoss Application Server

To copy the database access files on JBoss Application Server:

1. Copy libArcotAccessKeyProvider.so to the following directory:
   - **For Solaris**: <*JAVA_HOME used by JBoss Application Server instance*>/jre/bin
   - **For RHEL**: <*JAVA_HOME used by JBoss Application Server instance*>/jre/bin
2. Copy arcot-crypto-util.jar to the <*JAVA_HOME used by JBoss Application Server instance*>/jre/lib/ext directory.
3. Restart the application server.

   Note: An application server restart is required as part of some of the remaining installation tasks. To minimize the number of times you restart the application server, restart it once after performing the last task that requires a restart.

Step 3: Copying JDBC JARs to Your Application Server

Administration Console, UDS, and Sample Application, which are the Java-dependent components of AuthMinder, need JDBC JAR files to connect to the database. Copy these files to the application server.

Note: Before proceeding with the steps mentioned in the following subsections, ensure that you have downloaded the JDBC JARs, see chapter, "Preparing for Installation" (see page 37) for more information about supported JDBC JARs.

The following sections walk you through the steps for copying the JDBC JAR required for your database to one of the following application servers:

- Apache Tomcat
- IBM WebSphere
- Oracle WebLogic
- JBoss Application Server

Apache Tomcat

To copy the JDBC JAR file to the Apache Tomcat installation directory:

1. Navigate to the location where you have downloaded the JDBC JAR file.
2. Copy the JDBC JAR file and paste it in the following directory:
   - **For Apace Tomcat 5.5.x**: *<TOMCAT-HOME>*\common\lib
   - **For Apace Tomcat 6.x and 7.x**: *<TOMCAT-HOME>*\lib

   Alternatively, add the path that contains the JDBC JAR file to the **Classpath** environment variable.
3. Restart Apache Tomcat.

IBM WebSphere

To copy the JDBC JAR file to the IBM WebSphere:

1. Log in to WebSphere Administration Console.
2. Click **Environment,** and then click **Shared Libraries**.
   a. From the **Scope** drop-down, select a valid visibility scope. The scope must include the target server/node on which the application is deployed.
   b. Click **New**.
   c. Enter the **Name**, for example, **JDBCJAR**.
   d. Specify the **Classpath**. This path must point to the location where the JDBC JAR file is present and must also include the file name.
   e. Click **Apply** to save the changes.

3. Configure server-level class loaders.

   Note: You can either create a class loader or use the one that you created while performing the procedure described in Step 2: Copying Database Access Files to Your Application Server (see page 114).

   a. Click **Servers**, and then click **Application Servers.**

   b. Under **Application Servers**, access the settings page of the server for which the configuration is performed.

   c. Click **Java and Process Management**, and then click **Class Loader**.

   d. Click **New**. Select default **Classes loaded with parent class loader first** and click **OK.**

   e. Click the auto-generated **Class Loader ID**.

   f. In the class loader **Configuration** page, click **Shared Library References**.

   g. Click **Add**, select **JDBCJAR** and then click **Apply**.

   h. Save the changes.

4. Restart IBM WebSphere.

Oracle WebLogic

To copy the JDBC JAR file to Oracle WebLogic Server:

Note: If you are using Oracle Database, then you need *not* perform the configuration described in this section because Oracle WebLogic Server supports Oracle Database by default.

1. Copy the JDBC JAR file to the following directory:
   `<JAVA_HOME used by Oracle WebLogic instance>/jre/lib/ext directory.`
2. Log in to WebLogic Administration Console.
3. Navigate to **Deployments**.
4. Enable the **Lock and Edit** option.
5. Click **Install** and navigate to the directory that contains the JDBC JAR file.
6. Click **Next**.

   The Application Installation Assistant screen appears.

7. Click **Next**.

   The Summary page appears.

8. Click **Finish**.
9. Activate the changes.
10. Restart Oracle WebLogic Server.

JBoss Application Server

To copy the JDBC JAR file on JBoss Application Server:

1. Copy the JDBC JAR file to the following location on the JBoss Application Server installation directory:
   `<JBOSS_HOME>/server/default/lib`
2. Restart JBoss Application Server.

Step 4: (Mandatory for Oracle WebLogic 10.1) Creating Enterprise Archive File

Most application servers (such as WebSphere and WebLogic) enable you to bundle related JAR files or WAR files from the same vendor into a single enterprise application (or archive). As a result, all such related JARs or WARs can be deployed together and loaded by a class loader. Such an archive also contains an application.xml file, which is generated automatically and describes how to deploy each bundled module.

By default, AuthMinder provides WAR files to deploy Administration Console and User Data Service. If required, you can change the format of these files to EAR and then deploy the EAR files.

Perform the following steps to create an EAR file:

1. Open the command prompt window.
2. Navigate to the <*install_location*>/arcot/tools/common/bundlemanager directory.
3. Run the bundlemanager tool to create the EAR file by using the following command:
   `java -jar bundle-manager.jar -ear <filename.ear> -warList <war_file_name>`

   Note: In this command, replace *<war_file_name>* with arcotadmin.war to generate the EAR file for Administration Console. Similarly, replace *<war_file_name>* with arcotuds.war to generate the EAR file for UDS.

   This command generates the EAR file at <*install_location*>/arcot/java/webapps.

Deploying Administration Console

Note: If you plan to deploy Administration Console on WebSphere 7.0, then see the instructions that are provided in appendix, "Deploying Administration Console on IBM WebSphere 7.0" (see page 253).

You need the file arcotadmin.war to deploy the AuthMinder Administration Console. This file is available at:
<*install_location*>/arcot/java/**webapps**/

Note: To manage AuthMinder Server by using Administration Console, ensure that Administration Console can access the system where AuthMinder Server is installed by its host name.

To deploy Administration Console:

1. Change the working directory to:
   <*install_location*>/arcot/sbin
2. Type source arwfenv and press **Enter** to set the $ARCOT_HOME environment variable.
3. Restart the application server for the changes to take effect.
4. Deploy arcotadmin.war in the appropriate directory on the application server.

   Note: The deployment procedure depends on the application server that you are using. See your application server vendor documentation for detailed instructions. For example, in the case of Apache Tomcat, deploy the WAR file at <*APP_SERVER_HOME*>/**webapps**/.
5. **(For JBoss Application Server Only)** Perform the following steps if you have deployed Administration Console on JBoss Application Server:
   a. Copy the Bouncy Castle JAR file (bcprov-jdk15-146.jar) from <*install_location*>/arcot/java/**lib**/ to the following location:
      <*JBOSS_HOME*>/common/lib
   b. Navigate to the following location:
      <*JBOSS_HOME*>/server/default/conf/
   c. Open jboss-log4j.xml file in a text editor.
   d. Add the following log configuration in the <log4j:configuration> section:

```
<appender name="arcotadminlog"
class="org.apache.log4j.RollingFileAppender">
<errorHandler
class="org.jboss.logging.util.OnlyOnceErrorHandler"></errorHandler>
<param name="Threshold" value="INFO"/>
<param name="MaxFileSize" value="10MB"/>
<param name="MaxBackupIndex" value="100"/>
<param name="Encoding" value="UTF-8"/>
<param name="Append" value="true"/>
<param name="File" value="${arcot.home}/logs/arcotadmin.log"/>
```

```
<layout class="org.apache.log4j.PatternLayout">
<param name="ConversionPattern" value="%d{yyyy-MM-dd hh:mm:ss,SSS z} : [%t]
: %-5p : %-5c{3} : %m%n"/>
</layout>
<filter class="org.jboss.logging.filter.TCLMCFilter">
<param name="AcceptOnMatch" value="true"/>
<param name="DeployURL" value="arcotadmin.war"/>
</filter>
<!-- end the filter chain here -->
<filter class="org.apache.log4j.varia.DenyAllFilter"></filter>
</appender>
```

e. Add the following log category:

```
<category name="com.arcot">
<priority value="INFO" />
<appender-ref ref="arcotadminlog"></appender-ref>
</category>
```

f. Add the following category for cryptographic operations:

```
<category name="com.arcot.crypto.impl.NCipherCrypter">
<priority value="FATAL" />
<appender-ref ref="arcotadminlog"></appender-ref>
</category>
```

g. Save and close the file.

h. Take the backup of existing JBoss Application Server logging libraries. These library files are available at:
<*JBOSS_HOME*>/lib

i. Upgrade the JBoss Application Server logging libraries available at <*JBOSS_HOME*>/lib to version 2.1.1. The following table lists the JAR file names and the location from where you can download the files:

File Name	Location
jboss-logging-jdk-2.1.1.GA.jar	*http://repo1.maven.org/maven2/org/jboss/logging/jboss-logging-jdk/2.1.1.GA/*
jboss-logging-spi-2.1.1.GA.jar	*http://repo1.maven.org/maven2/org/jboss/logging/jboss-logging-spi/2.1.1.GA/*
jboss-logging-log4j-2.1.1.GA.jar	*http://repo1.maven.org/maven2/org/jboss/logging/jboss-logging-log4j/2.1.1.GA/*

6. Ensure that the application is restarted.

Verifying the Administration Console

The arcotadmin.log file is used for logging the Administration Console information.

To verify if the Administration Console was deployed successfully:

1. Navigate to the following location:
   `<install_location>/arcot/logs`
2. Open the arcotadmin.log file in any editor and locate the following lines:
   `Arcot Administration Console Configured Successfully.`

   This line indicates that your Administration Console was deployed successfully.

   Note: Check and resolve if there any FATAL and ERROR messages, and review all the WARNING messages for unexpected conditions.

Logging In to Administration Console

When logging in to Administration Console for the first time, use the Master Administrator credentials that are created automatically in the database during installation.

To log in to Administration Console:

1. Start Administration Console in a Web browser window, by using the following URL:

   http://<host>:<app_server_port>/arcotadmin/masteradminlogin.htm

   Note: The host and port information that you specify in the preceding URL must be of the application server where Administration Console is deployed.
2. Log in to Administration Console as a Master Administrator with the default Master Administrator account credentials. The credentials are:
   - **User Name***:* **masteradmin**
   - **Password***:* **master1234!**

Bootstrapping the System

Before you can start using the Administration Console to manage AuthMinder, first perform the following mandatory steps to initialize the system:

- Change the default Master Administrator password
- Specify the Global Key label
- Specify the authentication mechanism for the Default organization

Bootstrapping is a wizard-driven process that walks you through these setup tasks. Other administrative links are enabled after you perform these tasks.

Before you proceed with Performing the Bootstrapping Tasks (see page 88), you must learn about the related concept of Default Organization.

Default Organization

When you deploy the Administration Console, an organization is created by default. This organization is referred to as *Default Organization* (**DEFAULTORG**). As a single-organization system, the Default Organization itself can be used without creating any organizations.

Performing the Bootstrapping Tasks

When you first log in to the Administration Console as the Master Administrator (MA), the Summary screen for the Bootstrap wizard screen appears.

To bootstrap the system using the wizard:

1. Click **Begin** to start the process.

   The Change Password screen appears.

2. Specify the **Old Password**, **New Password**, **Confirm Password**, and click **Next.**

   The Configure Global Key Label screen appears.

3. Specify the **Global Key Label**, and click **Next**.

   AuthMinder enables you to use hardware- or software-based encryption of your sensitive data. Irrespective of hardware or software encryption, all Arcot products use *Global Key Label* for encrypting user and organization data.

   If you are using hardware encryption, then this label serves only as a reference (or pointer) to the actual 3DES key stored in the HSM device, and therefore the key label *must* match the HSM key label. However, in the case of software-based encryption, this label acts as the reference to the actual software key in the database.

   Caution: After you complete the bootstrapping process, you cannot update this key label.

   The **Storage Type** field indicates whether the encryption key is stored in the database (**Software**) or the HSM (**Hardware**).

   The Configure Default Organization screen appears.

4. Under the **Default Organization Configuration** section, specify the following parameters for the Default Organization:

   - **Display Name:** The descriptive name of the organization. This name appears on all other Administration Console pages and reports.
   - **Administrator Authentication Mechanism:** The mechanism that is used to authenticate administrators belonging to the Default Organization. Administration Console supports three types of authentication methods for the administrators to log in:
     - **Basic**

       If you select this option, then the built-in authentication method that is provided by the Administration Console is used for authenticating the administrators.

     - **LDAP User Password**

       If you select this option, then the administrators are authenticated using their credentials that are stored in the directory service.

Note: If this mechanism is used for authenticating administrators, then deploy UDS by following the procedure described in "Deploying User Data Service" (see page 128).

- **WebFort User Password**

  If you select the **WebFort User Password** option here, then the credentials are issued and authenticated by the AuthMinder Server.

Note: See *CA AuthMinder Administration Guide* for more information about how to do this.

5. Under the **Key Label Configuration** section of the Configure Default Organization screen, specify the following:
   - **Use Global Key:** This option is selected by default. Deselect this option if you want to override the Global Key Label you specified in the preceding step and specify a new encryption label.
   - **Key Label:** If you deselected the **Use Global Key** option, then specify the new key label that you want to use for the Default Organization.
   - **Storage Type:** This field indicates whether the encryption key is stored in the database (**Software**) or the HSM (**Hardware**).
6. Click **Finish** to complete the bootstrapping process.

   The Administration Console initialization is completed, as indicated in the Finish screen.
7. Click **Continue** to proceed with other configurations using the Administration Console.

Starting AuthMinder Server

Perform the following steps to start AuthMinder Server:

1. Navigate to the following directory:
   `<install_location>/arcot/bin/`
2. Run the following command:
   `./webfortserver start`

   Note: If you want to stop the server, then execute ./webfortserver stop command.

Verifying the Installation

You can verify whether the AuthMinder Server and the web applications have started successfully by applying the instructions that are described in the following sections:

- Using Log files (see page 90)
- Using webfortserver Utility (see page 91)
- Checking the Ports (see page 91)

Using Log files

Perform the following steps to verify that AuthMinder Server started correctly:

1. Navigate to the following location:
   `<install_location>/arcot/logs`
2. Open the arcotwebfortstartup.log file in any editor. This log file is created in the application server home directory.
3. Look for the following lines in the log file:
   ```
   INSTANCE_VER.................................: [7.1.01]
   Arcot WebFort Authentication Service READY
   ```
   These lines indicate that AuthMinder Server is installed successfully.

Note: Ensure that the log files do not contain any FATAL and WARNING messages.

Using webfortserver Utility

The webfortserver tool enables you to check the release of AuthMinder that you have installed.

Follow these steps:

1. Navigate to the following location:
   `<install_location>/arcot/bin`
2. Run the following command to start the webfortserver tool in interactive mode:
   `./webfortserver -i`
3. Enter the release number at the prompt.

   The webfort-ver-<dd>-<mmm>-<yy>.txt file is created in the <*install_location*>/arcot/logs folder.
4. Open this file and check for the following lines to ensure that you are using the current release:
   - The version of AuthMinder library files in the bin section is 7.1.01.
   - The version of the UDS library file (arwfuds.dll) in the bin section is 2.0.3.
5. Close the file.

Checking the Ports

Perform the following steps to verify if the AuthMinder Server is listening to different protocols on the default ports:

1. Navigate to the following location:
   `<install_location>/arcot/logs`
2. Open the arcotwebfortstartup.log file in any editor and search for the protocol names to verify if they are listening at the correct port, as shown in the following snippet.

```
PROTOCOLNAME : [Administration-WS]
PORTNO : 9745
PROTOCOLID : [Transaction-Native]
PORTNO : 9742
PROTOCOLID : [ServerManagement-WS]
PORTNO : 9743
PROTOCOLID : [Transaction-WS]
PORTNO : 9744
```

Note: See appendix, "Default Port Numbers and URLs" (see page 235) for information about default ports and protocols.

Deploying User Data Service

AuthMinder can access user data either from a relational database (RDBMS) or directly from an LDAP server:

You need the file arcotuds.war to deploy the User Data Service (UDS). This file is available at:
<*install_location*>/arcot/java/**webapps**/

To deploy User Data Service:

1. Deploy arcotuds.war in the appropriate directory on the application server.

   Note: The deployment procedure depends on the application server that you are using. See your application server vendor documentation for detailed instructions. For example, in case of Apache Tomcat, deploy the WAR file at <*APP_SERVER_HOME*>/**webapps**/.

2. (**For WebSphere Only**) Configure to reload the UDS class when the application files are updated.

   a. Navigate to **Application** > **Enterprise Applications** and access the UDS settings page.

   b. Under **Class loader order**, select the **Classes loaded with local class loader first (parent last)** option.

   c. Under **WAR class loader policy**, select the **Single class loader for application**.

   d. Click **Apply**.

3. **(For JBoss Application Server Only)** Perform the following steps if you have deployed UDS on JBoss Application Server:

   a. Copy the Bouncy Castle JAR file (bcprov-jdk15-146.jar) from <*install_location*>/arcot/java/**lib**/ to the following location:
   <*JBOSS_HOME*>/common/lib

   b. Navigate to the following location:
   <*JBOSS_HOME*>/server/default/conf/

   c. Open jboss-log4j.xml file in a text editor.

   d. Add the following log configuration in the <log4j:configuration> section:

```
<appender name="arcotudslog" class="org.apache.log4j.RollingFileAppender">
<errorHandler
class="org.jboss.logging.util.OnlyOnceErrorHandler"></errorHandler>
<param name="Threshold" value="INFO"/>
<param name="MaxFileSize" value="10MB"/>
<param name="MaxBackupIndex" value="100"/>
<param name="Encoding" value="UTF-8"/>
<param name="Append" value="true"/>
<param name="File" value="${arcot.home}/logs/arcotuds.log"/>
<layout class="org.apache.log4j.PatternLayout">
```

```
<param name="ConversionPattern" value="%d{yyyy-MM-dd hh:mm:ss,SSS z} : [%t]
: %-5p : %-5c{3}(%L) : %m%n"/>
</layout>
<filter class="org.jboss.logging.filter.TCLMCFilter">
<param name="AcceptOnMatch" value="true"/>
<param name="DeployURL" value="arcotuds.war"/>
</filter>
<!-- end the filter chain here -->
<filter class="org.apache.log4j.varia.DenyAllFilter"></filter>
</appender>
```

e. Add the following line in the com.arcot category that you created in Deploying Administration Console (see page 120):
`<appender-ref ref="arcotudslog"></appender-ref>`

f. Add the following line in the cryptographic category that you created in Deploying Administration Console (see page 120):
`<appender-ref ref="arcotudslog"></appender-ref>`

g. Save and close the file.

4. Restart the application server.

5. Perform the following steps to verify whether the UDS started correctly:

   a. Navigate to the following location:
   `<install_location>/arcot/logs`

   b. Open the arcotuds.log file in any editor and locate the following lines:
   `User Data Service (Version: 2.0.3) initialized successfully.`

   This line indicates that UDS was deployed successfully.

Note: Check and resolve if there any FATAL and ERROR messages, and review all the WARNING messages for unexpected conditions.

How to Deploy Strong Authentication on Additional Servers

After you install Strong Authentication Server and Administration Console you install the other components on an additional system in this distributed-system deployment.

1. Locate the Strong Authentication installer Installer.bin file.
2. Run the installer using the following command:
   `sh Strong Authentication-`*`version_number`*`-<`*`platform name`*`>-Installer.bin`

   The installer starts preparing for the installation.
3. Complete the steps in the Installing on the First System (see page 102), until you reach the Choose Product Features screen.
4. Select the components that you want to install.
5. Follow steps 12 through 19 to complete the installation.

Post-Installation Tasks

Perform the following post-installation tasks on the second system where you have installed Java SDKs, Web Services, and Sample Application:

- Deploying Sample Application (see page 131)
- Configuring Sample Application for Communication with AuthMinder Server (see page 131)
- Using Sample Application (see page 132)

Deploy Sample Application

You can use the Sample Application to verify a successful installation of Strong Authentication. In addition, Sample Application demonstrates the following features:

- The typical workflows
- The tasks that you can perform using the Java APIs
- Integration of your application with Strong Authentication

Follow these steps:

1. Deploy the Strong Authentication *version_number*-sample-application.war file from the following location:
   `<install_location>/arcot/samples/java`
2. Start Sample Application.
3. Access Sample Application by using the following URL:

   http://<host>:<app_server_port>/Strong Authentication-version_number-sample-application/

Configure Sample Application Communication Server

If you installed the Sample Application and Strong Authentication Server on different systems, you must configure the communication settings.

Follow these steps:

Access the Sample Application in a Web browser window.

1. In the navigation pane, click **Setup** -> **Server Connectivity** to open the Strong Authentication Server Connectivity page.
2. Specify the values for the connection parameters that are listed in the following table:

   Note: The configurations that are made using these parameters are valid for the current session. Set these values again if you restart Sample Application or the application server.

Field	Default Value	Description
IP Address	localhost	The host name or the IP address where the Strong Authentication Server is available.
Port	9742	The port on which the Authentication and the Issuance service is available.

Field	Default Value	Description
Maximum Active Connections	64	The maximum number of database connections maintained by the Sample Application.

3. Click **Set Up** to save the connection.

Using Sample Application

Sample Application enables you to issue and authenticate the credentials that AuthMinder Supports. You can use the Sample Application to perform these operations to test whether the AuthMinder installation has been successful.

This section covers the following tasks:

- Creating Users (see page 95)
- Setting Up ArcotID PKI Client (see page 96)
- Creating ArcotID PKI Credential (see page 97)
- Downloading ArcotID PKI (see page 98)
- Authenticating Using ArcotID PKI (see page 99)

Creating Users

Note: User creation must be performed either using Administration Console or Web Services.

To create users using Administration Console:

1. Log in to the Administration Console as a Global Administrator (GA) or an Organization Administrator (OA). The URL for the purpose is:

   http://<host>:<app_server_port>/arcotadmin/adminlogin.htm

2. If already not activated, activate the **Manage Users and Administrators** tab under the **Users and Administrators** tab.
3. In the left pane, under **Manage Users and Administrators**, click **Create User** to open the Create User page.
4. On the Create User page:

   a. Enter a unique user name, their organization name, and optionally, other user information in the **User Details** section.

   b. (Optional) Specify other user information in the corresponding fields on the page.

   c. Select the required **User Status**.

   d. Click **Create User.**

   The "Successfully created the user" message appears if the specified user was successfully added to the database.

5. Return to the WebFort Sample Application page.

Setting Up ArcotID PKI Client

To set up the ArcotID PKI Client to communicate with the AuthMinder Server for authenticating users with their ArcotID PKI, perform the following steps:

1. Access the Sample Application using the following URL:

   http://<host>:<app_server_port>/webfort-7.1.01-sample-application/

2. In the left pane, click **Setup** -> **ArcotID Client** to open ArcotID Client Settings page.
3. In the **Choose ArcotID Client** section, select the type of client that is to be used to authenticate the ArcotID PKI.
4. In the **Choose ArcotID Download Type** section, select the location where you want to store the ArcotID PKI.
5. In the **Choose Where & When to Obtain the ArcotID Challenge** section, select the mode of obtaining the ArcotID PKI challenge.
6. Click **Select** to save the settings.

   The "The operation was successful" message appears if the ArcotID PKI Client configuration was performed successfully.

Creating ArcotID PKI Credential

To create ArcotID PKI credential for users:

1. Access the Sample Application using the following URL:

   http://<host>:<app_server_port>/webfort-7.1.01-sample-application/

2. In the left pane, click **ArcotID** > **Issuance** > **Create** to open the Create ArcotID page.
3. Specify the name of the user you created in the **User Name** field.
4. (Optional) Specify the user's organization in the **Organization** field.
5. Specify the password to be used for authentication in the **ArcotID Password** field.
6. (Optional) Specify the profile that has to be used to issue ArcotID PKI in the **Profile Name** field.
7. (Optional) Specify the name-vale pairs of the **Unsigned Attributes**. The attributes are set in the unsigned portion of the ArcotID PKI.
8. (Optional) Specify the **Custom Attributes** to be used for creating the ArcotID PKI.
9. (Optional) Specify the **Additional Input** that you want to pass to the AuthMinder Server.
10. (Optional) Pass the following **Transaction Logging Parameters**:
    - In the **Log Level** field, select the logging level.

      Note: See the topic titled "WebFort Logging" in *CA AuthMinder Administration Guide* for more information about the log levels.
    - Select **Enable Trace Logging** if you want to capture flow details.
    - Select **Enable DB Logging** if you want to log the database activities.
    - Select **Enable Sensitive Data Logging** if you want to log the sensitive data.
11. Click **Create** to create the credential.

    The "The operation was successful" message appears if the ArcotID PKI was created successfully for the user.

Downloading ArcotID PKI

To download the ArcotID PKI:

1. Access the Sample Application using the following URL:

   http://<host>:<app_server_port>/webfort-7.1.01-sample-application/

2. In the left pane, click **ArcotID** > **Issuance** > **Download** to open the Download ArcotID page.
3. Specify the name of the user you created in the **User Name** field.
4. (Optional) Specify the user's organization in the **Organization** field.
5. (Optional) Specify the profile that has been used to issue ArcotID PKI in the **Profile Name** field.
6. (Optional) Specify the **Additional Input** that you want to pass to the AuthMinder Server.
7. (Optional) Pass the following **Transaction Logging Parameters**:
   - In the **Log Level** field, select the logging level.

     Note: See the topic titled "AuthMinder Logging" in *CA AuthMinder Administration Guide* for more information about the log levels.
   - Select **Enable Trace Logging** if you want to capture flow details.
   - Select **Enable DB Logging** if you want to log the database activities.
   - Select **Enable Sensitive Data Logging** if you want to log the sensitive data.
8. Click **Download** to download the user's ArcotID PKI.

Authenticating Using ArcotID PKI

To authenticate using the ArcotID PKI:

1. Access the Sample Application using the following URL:

   http://<host>:<app_server_port>/webfort-7.1.01-sample-application/

2. In left pane, click **ArcotID** > **Authentication** > **Authenticate** to open the ArcotID Authentication page.
3. Specify the name of the user you created in the **User Name** field.
4. Specify the user's organization in the **Organization** field.
5. Specify the user's ArcotID PKI password in the **ArcotID Password** field.
6. If you are using aliases to identify the users, then specify the **Application Context** based on the alias of the user that you want to authenticate.
7. (Optional) Select the **Token Type** that has to be returned to the user after successful authentication.

   Note: See the topic titled "Authenticating Users" in the *CA AuthMinder Developer's Guide* for more information about the token types.
8. (Optional) Specify the **Authentication Policy Name** that has to be used for authenticating users.
9. If you have selected SAML as the token type, then specify the **SAML Policy Name** to be used.
10. (Optional) Specify the **Additional Input** that you want to pass to the AuthMinder Server.
11. (Optional) Pass the following **Transaction Logging Parameters**:
    - In the **Log Level** field, select the logging level.

      Note: See the topic titled "AuthMinder Logging" in *CA AuthMinder Administration Guide* for more information about the log levels.
    - Select **Enable Trace Logging** if you want to capture flow details.
    - Select **Enable DB Logging** if you want to log the database activities.
    - Select **Enable Sensitive Data Logging** if you want to log the sensitive data.
12. Click **Authenticate** to verify the user's ArcotID PKI.

Performing Additional Configurations for CA Adapter 2.2.7

If you have a CA Adapter 2.2.7 instance and want to use it with AuthMinder 7.1.01, you must perform certain additional configurations. See Additional Configurations for CA Adapter 2.2.7 (see page 227) for the exact steps to be performed.

Post-Installation Checklist

Fill this checklist with installation and setup information for AuthMinder. You will need this information for various administrative tasks that you perform later.

Your Information	Example Entry	Your Entry
ARCOT_HOME	/var/opt/arcot/	
SYSTEM INFORMATION		
Host Name	my-bank	
User Name	administrator	
Password	password1234!	
Configured Components	WebFort Server Administration Console User Data Service	
ADMINISTRATION CONSOLE INFORMATION		
Host Name	localhost	
Port	8080	
Master Administrator Password	mypassword1234!	
USER DATA SERVICE INFORMATION		
Host Name	localhost	
Port	8080	
Application Context Root	arcotuds	

Chapter 6: Configuring AuthMinder Java SDKs and Web Services

This chapter describes the steps to configure the Java Software Development Kit (SDK) and Web Services provided by AuthMinder.

The chapter covers the following topics:

AuthMinder APIs

AuthMinder is shipped with a set of Java APIs that are available as JAR files at the following location:
<*install_location*>/arcot/sdk/client/java/lib/arcot/

The following JAR files are located in this directory:

- **arcot-webfort-authentication.jar**

  The JAR file that you need to implement the AuthMinder Authentication SDK. This file consists the Java package that contains the logic for user authentication. Operations that this package enables include:

  - Authenticate users using credentials that AuthMinder supports
  - Authenticate custom credentials

- **arcot-webfort-common.jar**

  This file consists the Java package that contains the classes that are common for Authentication and Credential Management (Issuance) operations. This package is used for the following operations:

  - Pass additional information to the AuthMinder Server
  - Specify the One-Time Password type
  - Specify the User and Credential Status

- **arcot-webfort-issuance.jar**

  The JAR file that you need to implement the WebFort Credential Management SDK. This file consists the Java package that contains the logic for managing credentials. Operations that this package enables include:

  - Create and manage credentials
  - Manage ArcotID PKI key bag

Configuring Java SDKs

This section provides the procedure to configure the Authentication and Credential Management Java SDKs so that they can be integrated with your existing application.

- Configuring Authentication Java SDK (see page 141)
- Configuring Credential Management Java SDK (see page 142)

Note: Before proceeding with the configuration steps in this section, ensure that the JAR file that is required to implement the Java APIs are installed in the <*install_location*>/arcot/**sdk/client/java/lib/** location.

Configuring Authentication Java SDK

To configure Authentication SDK for using in a J2EE Application:

1. Copy the listed JAR files from the following location:
   `<install_location>/arcot`

   to the appropriate location in your *<APP_SERVER_HOME>* directory. For example, on Apache Tomcat this location is *<Application_Home>*/WEB-INF/lib.

   - sdk/client/java/lib/arcot/arcot-webfort-authentication.jar
   - sdk/client/java/lib/arcot/arcot-webfort-common.jar
   - sdk/client/java/lib/external/bcprov-jdk15-146.jar
   - sdk/client/java/lib/external/commons-pool-1.5.5.jar

2. Copy the webfort.authentication.properties configuration file containing the server connection parameters from:
   `<install_location>/arcot/sdk/client/java/properties`

   to the appropriate location in your <APP_SERVER_HOME> directory. For example, on Apache Tomcat this location is <Application_Home>/WEB-INF/classes/properties.

Note: To know more about APIs and their initialization, see *CA AuthMinder Java Developer's Guide* and the AuthMinder Javadocs at <*install_location*>/arcot/docs/webfort/Arcot-WebFort-7.1.01-authentication-sdk-javadocs.zip.

Configuring Credential Management Java SDK

To configure Credential Management SDK for using in a J2EE Application:

1. Copy the listed JAR files from the following location:
   `<install_location>/arcot`

   to the appropriate location in your *<APP_SERVER_HOME>* directory. For example, on Apache Tomcat this location is *<Application_Home>*/WEB-INF/lib

   - sdk/client/java/lib/arcot/arcot-webfort-common.jar
   - sdk/client/java/lib/arcot/arcot-webfort-issuance.jar
   - sdk/client/java/lib/external/bcprov-jdk15-146.jar
   - sdk/client/java/lib/external/commons-pool-1.5.5.jar

2. Copy the webfort.issuance.properties configuration file containing the server connection parameters from:
   `<install_location>/arcot/sdk/client/java/properties`

   to the appropriate location in your <APP_SERVER_HOME> directory. For example, on Apache Tomcat this location is <Application_Home>/WEB-INF/classes/properties.

Note: To know more about Java APIs and their initialization, see *CA AuthMinder Java Developer's Guide* and the AuthMinder Javadocs at <*install_location*>/arcot/docs/webfort/Arcot-WebFort-7.1.01-issuance-sdk-javadocs.zip.

Working With AuthMinder Web Services

This section covers the following topics:

Introduction to AuthMinder Web Services

AuthMinder provides Web Services for managing users, organizations, and user credentials, and authenticating users and performing bulk operations.

The following table lists the WSDL documents that you can use to generate the Web Services client code to communicate with the AuthMinder Server. These WSDLs are available at the following location:
<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.
uds/**ArcotUserManagementSvc.wsdl**	Used to create and manage users and user accounts.
webfort/**ArcotWebFortAdminSvc.wsdl**	Used to define AuthMinder configurations.
webfort/**ArcotWebFortIssuanceSvc.wsdl**	Used to manage user credentials.
webfort/**ArcotWebFortAuthSvc.wsdl**	Used to authenticate users.
webfort/**ArcotWebFortBulkOperationsSvc.wsdl**	Used to perform bulk operations such as, creating credentials, assigning and fetching OATH tokens that are available to the organizations.

Generating Client Code

Perform the following steps to generate the client code:

1. Stop the application server.
2. Navigate to the following location:
   <install_location>/arcot/wsdls/*<required_folder>*
3. Use the WSDL file to generate the client code.
4. Restart the application server.
5. In a browser window, access the end-point URLs to verify whether the client can access the Web Service.

Note: See the *CA AuthMinder Web Services Developer's Guide* for more information about how to access the end-point URLs.

Enabling SSL Communication

AuthMinder supports Secure Socket Layer (SSL) between the AuthMinder Server and Java SDKs. The topic titled "Configuring SSL" in *CA AuthMinder Administration Guide* describes how to set the transport mode as SSL between the AuthMinder Server and its clients.

Chapter 7: Upgrading to Release 7.1.01

This section describes the steps to upgrade your existing release of AuthMinder to release 7.1.01. It includes the following topics:

Notes on Backward Compatibility

- The older Java SDK client will continue to work with the new installation of AuthMinder Server. It will *not* require any client code modification.
- Older WSDLs will continue to work with the new installation of AuthMinder Server. They will *not* require any client code modification.

Upgrade Overview

You can upgrade to release 7.1.01 from any of the following releases:

- 5.3.x or 5.4.x
- 6.x or 7.x

Important! If you have a release of AuthMinder that is not listed here, then apply the required patches to upgrade to one of these releases, and then proceed with the upgrade. See the corresponding Release Notes for the patch upgrade instructions.

If you are upgrading from release 5.3.x or 5.4.x, first upgrade to release 6.0 and then upgrade to release 7.1.01. However, if you are upgrading from release 6.x or 7.x, then directly upgrade to release 7.1.01.

Database Privileges Required for Upgrade

The following table lists the database privileges that you must have for performing the database procedures that are related to upgrading to release 7.1.01:

Database Type	Upgrade Privileges	Run time Privileges
Oracle	CREATE TABLE	CREATE TABLE
	CREATE ANY INDEX	DML Privileges
	CREATE ANY SEQUENCE	
	CREATE TABLESPACE (*for Reports*)	
	CREATE PROCEDURE	
	UNLIMITED TABLESPACE (*for Reports, optional*)	
	DROP TABLESPACE	
	ALTER ANY TABLE	
	ALTER TABLESPACE	
	DML Privileges (including CREATE SESSION privilege)	
MS SQL Server **Note:** The UserID must also have the Database role of ddladmin.	CREATE TABLE	CREATE TABLE
	CREATE INDEX	DML Privileges
	CREATE PROCEDURE	
	EXECUTE PROCEDURE	
	REFERENCES	
	ALTER TABLE	
	DML Privileges	
IBM DB2	CREATE TABLE	CREATE TABLE
	CREATE INDEX	DML Privileges
	CREATE SEQUENCE	
	CREATE TABLESPACE	
	CREATE TABLESPACE with AUTORESIZE = yes (*for Reports, optional*)	
	CREATE PROCEDURE (SQL - native)	

Database Type	Upgrade Privileges	Run time Privileges
	ALTER TABLE	
	DML Privileges	

Upgrading to 7.1.01

Upgrading from 5.3.x or 5.4.x to 7.1.01 is a two-stage procedure. You first upgrade to 6.0 and then upgrade from 6.0 to 7.1.01. In contrast, if you are upgrading from 6.x or 7.x, you directly upgrade to 7.1.01.

Perform the following steps to upgrade to 7.1.01:

1. Performing Pre-Upgrade Tasks
2. If you are migrating from 5.3.x or 5.4.x, perform the steps described in the following sections:
   - Migrating the Database to Release 6.0 for Arcot Common Components
   - Migrating the Database to Release 6.0 for AuthMinder Components
   - Performing Post-Migration Tasks for Release 6.0
3. Updating the securestore.enc File and Setting the TrustStore Password (shared)
4. Migrating the Database to Release 7.1.01 for Arcot Common Components
5. Migrating the Database to Release 7.1.01 for AuthMinder Components
6. Uninstalling the Existing Release of AuthMinder
7. Depending on whether the existing AuthMinder deployment is on a single system or distributed system, perform the steps described in one of the following sections:
   - Reinstalling AuthMinder on a Single System
   - Reinstalling AuthMinder on a Distributed System
8. During upgrade, if there are any warnings during AuthMinder Server startup and if your transactions fail, then perform the steps listed in (In Error Scenario Only) Reverting to Your Initial Setup (IG).
9. Performing Post-Upgrade Tasks (shared)

Perform Pre-Upgrade Tasks

This section depicts pre-upgrade steps.

Important! For an Strong Authentication deployment on a distributed system, perform the upgrade on the system where the Strong Authentication Server is installed.

Follow these steps:

1. Shut down the following servers:
   - Strong Authentication Server
   - Application server where CA Advanced Authentication and User Data Service are deployed.
2. Verify that JDK installed on the system. Refer to *Platform Support Matrix* https://support.ca.com/phpdocs/7/8190/adv_authentication_platform_support_matrix.pdf?intcmp=searchresultclick&resultnum=5 for JDK versions.
3. Verify that the database is available throughout the upgrade process.
4. Verify that the database on which you are performing the upgrade is not set up for replication. Disable database replication before the upgrade.
5. Copy the contents of the existing ARCOT_HOME directory to a new directory.

   ARCOT_HOME refers to the base directory that contains the entire directory structure. **Example:** ARCOT_HOME refers to <*install_location*>/arcot/.

   Copy the entire contents of $ARCOT_HOME to a new directory. This directory is referred to as ARCOT_HOME_BACKUP.
6. Open the $ARCOT_HOME/conf/arcotcommon.ini file in a text editor, and then perform the following steps:
   a. Verify that the primary database details are correct. The upgrade tool uses the database that is configured in this file.
   b. If you have configured a backup database, disable the backup database by commenting the lines with the following properties. These properties are in the arcot/db/backupdb section of the arcotcommon.ini file:
      - URL.1
      - AppServerConnectionPoolName.1
      - Username.1
   c. Include the following section in the arcotcommon.ini file if the current version is 6.2.9:
      ```
      [arcot/crypto/device]
      HSMDevice=S/W
      ```

7. If you are upgrading from release 6.0 on IBM DB2, set the SYSTEM TEMPORARY tablespace page size to a minimum of 16K. See database vendor documentation for more information..
8. Back up the database containing the Strong Authentication schema.
9. Consult your DBA to configure the database depending on the database volume requirements.
10. Verify that you have sufficient database privileges to upgrade Strong Authentication.
11. If you have stored your user details in an LDAP repository in the previous release, verify that the LDAP server is available throughout the upgrade process.
12. Verify that the $ARCOT_HOME environment variable is set to the directory where Strong Authentication is installed.
13. If you have registered plug-ins with this installation, save the startup log. This file is available in the $ARCOT_HOME/logs/ directory, and it contains details of the plug-ins. After the upgrade, recompile the plug-ins.

Migrate the Common Components Database

To migrate the database for the common components that Strong Authentication uses, perform the following steps.

Follow these steps:

1. Copy the upgrade directory to a temporary location on the system where you plan to upgrade.

   This folder contains the following zip files that are applicable for this migration path:

   - ca-common-upgrade-1.0.x-2.0.zip
   - ca-strongaauth-upgrade-6.2.9-or-7.x-8.0 .zip

   Important! If Risk Authentication is already upgraded to 8.0, or if current Strong Authentication version is 7.x, or if current Risk Authentication version is 3.x, then ignore Steps# 2 to 12.

2. Copy the ca-common-upgrade-1.0.x-2.0.zip file to the $ARCOT_HOME directory:
3. Extract the contents of the ca-common-upgrade-1.0.x-2.0.zip file in this directory.
4. Navigate to the following directory:
   `%ARCOT_HOME%/tools/common/upgrade/`

   Extract the contents of the arcot-common-db-upgrade zip in this directory.
5. Copy the database jar file corresponding to the following d**atabas**e with the same name, to the %ARCOT_HOME%/tools/common/upgrade/lib/ directory:

   - **Oracle Database:** ojdbc.jar
   - **Microsoft SQL Server:** sqljdbc.jar
   - **IBM DB2 UDB:** db2jcc.jar

   Note: For Oracle Database and IBM DB2 UDB, use the JDBC JAR version that is applicable to the database. For Microsoft SQL Server, use sqljdbc4.0 (Microsoft JDBC Driver 4.0 for SQL Server).
6. Copy the libArcotAccessKeyProvider.so file to *<JAVA_HOME used by App Server>*/jre/sbin. Example: In Strong Authentication release 6.x, the libArcotAccessKeyProvider.so file is available in <ARCOT_HO*ME>/java/ext/<platform name>/<32bit or 64 bit>.*
7. Copy the $ARCOT_HOME/java/lib/arcot-crypto-util.jar file

   to
   `JAVA_HOME-used-by-App-Server/jre/lib/ext`
8. Set and export the LD_LIBRARY_PATH to the directory where libArcotAccessKeyProvider.so is copied.
9. Change your working directory to:
   `$ARCOT_HOME/tools/common/upgrade/`

10. Run the arcot-common-upgrade-framework.jar file by using the following command:
    `java [JVM_Options] -jar arcot-common-upgrade-framework.jar [--log-file <log-file-name>] [--log-level <log-level>][--commit-batch-size <batch_size>] [--product-name common][--prompt][--mst]`

    The following table describes the options that are supported by this JAR:

Option	Description
JVM-Options	The following JVM options are required only if LDAP organizations are configured: ■ **-Xmx1024m**: Sets the maximum heap size to 1GB. If more than 1,00,000 users exist in the configured LDAP, then it is highly recommended that you increase the heap size to 2048m (2 GB). ■ **-Dcom.arcot.ldap.migration.timeout=<duration>**: The time duration, in minutes, after which the migration of the LDAP organization is marked as failed. The LDAP migration timeout for 1,00,000 users is approximately 240 minutes or 4 hours. However, this would depend on the type of hardware configuration being used. The default value of this parameter is 240 minutes.
log-file	Specifies the path to the log file: ■ If you do not provide any value, the arcot_common_upgrade.log file is created in the %ARCOT_HOME%\logs\ directory. ■ If you provide an absolute path, the log file is created at the given location. ■ If you provide a file name, the log file is created in %ARCOT_HOME% with the given file name.
-log-level	Specifies the log level. If you do not provide any value, the upgrade log level is set to INFO.
commit-batch-size	Specifies the number of transactions to be issued to the database before a COMMIT statement is issued.
product-name	Specifies the name of the product that has to be migrated. The default value is common.

Option	Description
prompt	Prompts whether to proceed further after each phase is completed successfully. You can choose to run the tool later to continue from where it stopped. If this option is not specified, the upgrade tool runs without any prompting until the upgrade process is completed.
mst	Refers to the Monitoring Sleep Time. If you specify this option, the upgrade tool prints diagnostic messages describing the progress made during upgrade after sleeping for the specified duration (in minutes.) The default value is two minutes.

12. Check the log file $ARCOT_HOME/logs/arcot_common_upgrade.log to ensure that the common database migration was successful.

Important! While migrating the database, if you encounter errors due to a step that you have performed incorrectly, restore the database backup that you created earlier. After you verify that the database is correctly restored, retry the database migration procedure.

Migrate the Database

After you migrate the database for common components, migrate the database to the release 6.0 state for components.

Important! Perform this procedure only if you upgrade from release 5.3.x or 5.4.x.

Follow these steps:

1. Copy the arcot-webfort-upgrade-5.x-6.0.zip file to the $ARCOT_HOME directory.
2. Unzip the arcot-webfort-upgrade-5.x-6.0.zip file in this directory.
3. Navigate to the following directory:
   `$ARCOT_HOME/arcot-webfort-upgrade-5.x-6.0/dbscripts/<db_type>`
4. Run the scripts in the *following* order:
   a. arcot-db-config-for-webfort-6.0.sql
   b. upgrade-for-webfort.sql
5. Open a command window.
6. Change your working directory to:
   `$ARCOT_HOME/arcot-webfort-upgrade-5.x-6.0/tools/<platform_name>`
7. At the prompt, run the following command:
   `wfupgrade -migrate`

   The preceding command loads the OpenSSL CA and the domain key that is used for the ArcotID, and migrates the configuration data from older database tables to the release 6.0 tables.

   The wfupgrade tool generates the webfort-6.0-upgrade.log file in the $ARCOT_HOME/logs/ directory.

   Important! Ensure that you run the wfupgrade upgrade tool from the system where the AuthMinder Server is installed.
8. Open the webfort-6.0-upgrade.log file in a text editor and ensure that it does not contain any FATAL and WARNING messages.

Perform Post-Migration Tasks

After you migrate the database, test the new installation and then remove the earlier schema from the database, and then perform some other post-migration steps.

Follow these steps:

1. Navigate to the following location:
   `$ARCOT_HOME/dbscripts/<db_type>/upgrade-scripts/`
2. Run the arcot-post-upgrade-for-common-1.0.sql script file.

   Important! If you have installed both Strong Authentication and Risk Authentication, then run this script only after you have upgraded both products.

   This script verifies that the following changes are applied:

   - The user ID for Master Administrator is changed from MASTER_ADMIN to MASTERADMIN.
   - The password for the MASTERADMIN account is master1234!.
   - The MASTERADMIN account belongs to the MASTERADMIN organization. This is useful when you filter reports.
   - The Administrators group is configured with Strong Authentication User Password authentication. Administrators belonging to this group must continue to use the same user name and password.
   - Group2 is the initial Default Organization.

3. Navigate to the following location:
   `$ARCOT_HOME/dbscripts/<db_type>`
4. Run the following script available in this directory to drop the schema:
   - For 5.3.x: ArcotWebFort.drop.sql
   - For 5.4.x: drop-webfort-<*version_number*>.sql
5. For the 5.3.x or 5.4.x Java SDKs to work with Strong Authentication 6.0:
   a. Remove the existing Strong Authentication Java SDK files.
   b. Update your applications with the following new files:
      - $ARCOT_HOME/arcot-webfort-upgrade-5.x-6.0/sdk/java/lib/arcot/arcot_core.jar
      - $ARCOT_HOME/arcot-webfort-upgrade-5.x-6.0/sdk/java/lib/arcot/arwf_issuance_sdk.jar
      - $ARCOT_HOME/arcot-webfort-upgrade-5.x-6.0/sdk/java/lib/arcot/arwf_auth_sdk.jar
      - All files available in $ARCOT_HOME/arcot-webfort-upgrade-5.x-6.0/sdk/java/lib/external/
   c. Migrate the existing configuration that is saved in the authentication.properties file to the following file:
      `$ARCOT_HOME/arcot-webfort-upgrade-5.x-6.0/sdk/java/properties/webfort-5.4.1.authentication.properties/`
   d. Migrate the existing configuration that is saved in the issuance.properties file to the following file:
      `$ARCOT_HOME/arcot-webfort-upgrade-5.x-6.0/sdk/java/properties/webfort-5.4.1.issuance.properties/`
6. Perform the following steps to verify that the migration of the database to the Release 6.0 state was successful:
   a. Navigate to the following location:
      `<install_location>/arcot/logs/`
   b. Open the arcotwebfortstartup.log file in any editor, and locate the following lines in the file:
      ```
      INSTANCE_VER.................................: [6.0]
      Arcot WebFort Authentication Service READY
      ```
   c. Ensure that the log file does not contain any FATAL or WARNING messages.

Update the Server Connection Pooling Settings

If application server connection pooling is used, or if SSL is configured for the connection with the database, perform the following steps.

Follow these steps:

1. If the application server connection pooling is used in your existing deployment, then update the database details in the securestore.enc file by the following steps:
   a. Change the working directory to the following location where DBUtil is available:
   `$ARCOT_HOME/tools/linux`
   b. Run the following command for the primary database:
   `DBUtil -pi <DB_username> <DB_password>`
2. If SSL has been configured for the connection with the database, then update the database details in the securestore.enc file by the following steps:
   a. Change the working directory to the following location where DBUtil is available:
   `$ARCOT_HOME/tools/linux`
   b. If the database is enabled for SSL communication, then set the truststore password as:
   `DBUtil -pi TrustStorePath.1 <truststore-password>`

Migrate the Database

Important! If you have installed CA RiskMinder with AuthMinder and you have completed the RiskMinder upgrade, then do *not* migrate the database for Arcot common components because this step has already been performed during the RiskMinder upgrade.

Migrate the database for the Arcot common components that are used in release 7.01.

Follow these steps:

1. Copy the upgrade directory to a temporary location on the system where you plan to upgrade.

   This folder contains the following zip files that are applicable for this migration path:

   - arcot-common-upgrade-1.0.x-2.0.zip
   - arcot-webfort-upgrade-6.x-or-7.x-7.1.01.zip

2. Copy the arcot-common-upgrade-1.0.x-2.0.zip file to the following directory: `$ARCOT_HOME`
3. Extract the contents of the arcot-common-upgrade-1.0.x-2.0.zip file in this directory.
4. Navigate to the $ARCOT_HOME/tools/common/upgrade/ directory.
5. Extract the contents of the arcot-common-db-upgrade.zip file in this directory.
6. Copy the database jar file corresponding to your database to the $ARCOT_HOME/tools/common/upgrade/lib directory with the same name, as follows:

   - Oracle: ojdbc.jar
   - Microsoft SQL Server: sqljdbc.jar
   - DB2: db2jcc.jar

   Note: For Oracle and DB2, use the JDBC jar version that is applicable to your database. For Microsoft SQL Server, use sqljdbc4.0 (Microsoft JDBC Driver 4.0 for SQL Server).

7. Locate the JAVA_HOME used by the existing installation. When you run the upgrade tool, ensure that you use the same JAVA_HOME or the JAVA_HOME from a later, supported version.
8. Ensure that the libArcotAccessKeyProvider.so file is configured for your application server. This file is in the $ARCOT_HOME/native/*<platform-name>*/32bit-or-64bit/ directory.
9. Set and export the LD_LIBRARY_PATH to the directory where the libArcotAccessKeyProvider.so file is copied.
10. Change your working directory to:

```
$ARCOT_HOME/tools/common/upgrade/
```

11. Run the arcot-common-upgrade-framework.jar file by using the following command:

```
java [JVM_Options] -jar arcot-common-upgrade-framework.jar
[--log-file <log-file-name>] [--log-level
<log-level>][--commit-batch-size <batch_size>] [--product-name
common][--prompt][--mst]
```

The following table describes the options that are supported by this JAR file:

Option	Description
JVM-Options	The following JVM options are required only if LDAP organizations are configured: ■ -Xmx1024m: Sets the maximum heap size to 1 GB. If more than 1,00,000 users exist in the configured LDAP, then it is highly recommended that you increase the heap size to 2048m (2 GB). ■ -Dcom.arcot.ldap.migration.timeout=<*duration*>: The time duration, in minutes, after which the migration of the LDAP organization is marked as failed. The LDAP migration timeout for 1,00,000 users is approximately 240 minutes or 4 hours. However, this would depend on the type of hardware configuration being used. The default value of this parameter is 240 minutes.
log-file	Specifies the path to the log file: ■ If you do not provide any value, the arcot_common_upgrade.log file is created in the $ARCOT_HOME/logs/ directory. ■ If you provide an absolute path, the log file is created at the given location. ■ If you provide a file name, the log file is created in $ARCOT_HOME with the given file name.
-log-level	Specifies the log level. If you do not provide any value, the upgrade log level is set to INFO.
commit-batch-size	Specifies the number of transactions to be issued to the database before a COMMIT statement is issued.
product-name	Specifies the name of the product that has to be migrated. The default value is common.

Option	Description
prompt	Prompts whether to proceed further after each phase is completed successfully. You can choose to run the tool later to continue from where it stopped. If this option is not specified, the upgrade tool runs without any prompting until the upgrade process is completed.
mst	Refers to the Monitoring Sleep Time. If you specify this option, the upgrade tool prints diagnostic messages describing the progress made during upgrade after sleeping for the specified duration (in minutes.) The default value is two minutes.

1. Check the log file (default file is $ARCOT_HOME/logs/arcot_common_upgrade.log) to make sure that the common database upgrade is successful.

   The upgrade tool also prints verification information.

   Important! While migrating the database, if you encounter errors due to a step that you have performed incorrectly, restore the database backup that you created earlier. After you verify that the database is correctly restored, retry the database migration procedure.

Migrate the Strong Authentication Database

After you upgrade the database for the common components, upgrade the database for Strong Authentication components.

Follow these steps:

1. Unzip the ca-strongaauth-upgrade-6.2.9-or-7.x-8.0.zip in $ARCOT_HOME directory.
2. Change your working directory to the following directory:
   `$ARCOT_HOME/ca-strongaauth-upgrade-6.2.9-or-7.x-8.0.zip/tools/<platform_name>`
3. Navigate to $ARCOT_HOME/sbin, and run the following cmd:

   ```
   source arwfenv
   ```

4. Run the following command:
   `./wfupgrade -migrate`

   This command migrates the 6.x or 7.x, 8.0 configuration data from older database tables to the 8.0 tables.

   The wfupgrade tool generates the ca-strongaauth-8.0-upgrade.log file in the $ARCOT_HOME/logs/ directory.

   Important! Verify that you run the wfupgrade upgrade tool from the system where the Server is installed.

5. Open the log file in a text editor and ensure that it does not contain any FATAL and WARNING messages.

Uninstall the Existing Release

Uninstall the existing release of Strong Authentication, and un-deploy the CA Advanced Authentication and UDS that are installed on the application server.

Note: If the instructions given in this section do not match the uninstallation options, follow the uninstallation instructions that are given in the installation guide for your existing release of Strong Authentication.

Follow these steps:

1. Uninstall the existing release follows:
   a. Shut down the following components:
      - Strong Authentication Server
      - Any application servers where other components are deployed
   b. Verify that the CA Advanced Authentication is close.
   c. Verify that all INI and other files that are related to configuration are closed.
   d. Navigate to the following directory:
      `$ARCOT_HOME/arcot/Uninstall_Arcot WebFort/`
   e. Run the installer using the following command:
      `sh Uninstall Arcot WebFort`

      You are prompted to specify uninstallation options.
   f. Enter 1 to specify that you want all features and components to be removed.
   g. Press Enter to confirm.
   h. Press Enter to complete the uninstallation.
   i. Delete any files that are left over in the $ARCOT_HOME directory.
2. Un-deploy the CA Advanced Authentication and User Data Service Web applications from the application server and shut down the application server gracefully. Then, refresh the application server cache.
3. Verify that the .com.zerog.registry.xml file has been deleted. This hidden file is created during the installation. The location of this file depends on the user account that is used to install Strong Authentication:
   - If you had installed Strong Authentication as the root user, this file is located in the /var directory.
   - If you had installed Strong Authentication as another user, this file is located in the user's HOME directory.

Reinstall Strong Authentication

Depending on the deployment model you used perform the tasks described in one of the following sections:

- Reinstall Strong Authentication on a Single System (see page 162)
- Reinstall Strong Authentication on a Distributed System (see page 163)

How to Reinstall Strong Authentication on a Single System

Perform the following tasks to reinstall Strong Authentication on a single system.

Important! The information in these sections apply to a fresh installation of Strong Authentication.

Use the database that you migrated earlier during the upgrade operation. Install Strong Authentication at the same location where the older release is installed. If you install in a different location, the Strong Authentication Server will *not* start.Perform Complete Installation

Complete the following steps:

1. Verify the Database Setup
2. Prepare Your Application Server
3. Deploy CA Advanced Authentication
4. Verify CA Advanced Authentication
5. Start Strong Authentication Server
6. Verify the Installation

   Note: If there is any warning during Server startup or while verifying the installation and transactions fails, then the upgrade has not been performed successfully. Use the information given in Troubleshooting Issues During Upgrade. If that does not help, revert to your initial setup by following the steps that are listed in Reverting to Your Initial Setup.
7. Deploy User Data Service
8. Deploy Sample Application
9. Use Sample Application
10. Perform Additional Configurations for CA Adapter 2.2.7
11. Post-Installation Checklist

How to Reinstall Strong Authentication on a Distributed System

Perform the tasks described in the following sections to reinstall Strong Authentication on a distributed system:

Important!

The information given in these sections apply to both a fresh installation of Strong Authentication on, *Chapter: Deploy Strong Authentication on Distributed System*, on *Install Strong Authentication On UNIX*.

Use the database that you migrated earlier during the upgrade. Install Strong Authentication at the same location where the older release is installed. If you install in a different location, the Strong Authentication Server does *not* start.

Follow these steps:

1. Install on the First System
2. Verify the Database Setup
3. Prepare Your Application Server
4. Deploy CA Advanced Authentication
5. Verify CA Advanced Authentication
6. Log In to CA Advanced Authentication
7. Start Strong Authentication Server
8. Verify the Installation
9. Deploy User Data Service
10. Install on the Second System
11. Deploy Sample Application
12. Configure Sample Application for Communication with Strong Authentication Server
13. Use Sample Application
14. Perform Additional Configurations for CA Adapter 2.2.7

Complete Post-Upgrade Tasks

Complete the following post-upgrade task:

- If you have disabled the backup database, enable it by editing the arcot/db/backupdb section of the $ARCOT_HOME/conf/arcotcommon.ini file, and synchronize the backup database with the primary database. If you disable database replication before the upgrade, then enable replication after the upgrade.
- If you had registered plug-ins with this installation before the upgrade, then recompile the plug-ins. Verify that the names of the recompiled files are the same as before. Use the startup log that you had saved before the upgrade to determine details of the plug-ins.
- If you require multi-byte character or internationalization support and if your database does not currently support multi-byte data, then migrate the database to a character set that supports multi-byte data. For more information, see "Configuring Database Server" in the *CA Strong Authentication Installation and Deployment Guide for UNIX Platforms*.

If your upgrade path is from 6.x or 7.x, 8.0, then perform the following configurations using CA Advanced Authentication.

1. Create new authentication policies at the global-level for all the previous authentication configurations.
2. Re-create the RADIUS configuration.
3. Set up the following Server instance configurations:
   - Database connectivity settings
   - Log file settings
4. Set up thread configurations per protocol.
5. Create ASSP configurations.
6. (Optional) Configure CA AuthID grace period. To do so, set the **Allow Successful Authentication** field in the CA AuthID policy.
7. (Optional) Enable caller verification for QnA credential. To do so, set the **Enable Caller Verification** field in the QnA policy.

Troubleshooting Issues During Upgrade

This section describes troubleshooting steps that you can apply to resolve errors that you may face while upgrading AuthMinder.

Problem:

The upgrade tool fails with the following error:

```
Error Occured: IO exception while parsing,
$ARCOT_HOME/tools/common/upgrade/xml/arcot-common-upgrade-meta-dat
a.xml
```

Cause:

The upgrade tool was not able to find the arcot-common-upgrade-meta-data.xml file.

Solution:

Check if the arcot-common-upgrade-meta-data.xml file exists in $ARCOT_HOME/tools/common/upgrade/xml. This error can commonly occur when the arcot-common-db-upgrade.zip file is not extracted using the **Extract To Here** option.

Problem:

The upgrade tool fails with the following error:

```
Internal Error: Could not initialize upgrade tool. Error:: Cannot load
JDBC driver class 'oracle.jdbc.driver.OracleDriver' Error Occured:
Upgrade Initialization Error:oracle.jdbc.driver.OracleDriver
```

Cause:

The upgrade tool could not find the JDBC library to connect to the database.

Solution:

Check whether the JDBC library is copied to the $ARCOT_HOME/tools/common/upgrade/lib directory.

If the JDBC library is already copied, then check whether the name of the JDBC jar file is correctly specified, as described in the procedure to migrate the database to release 6.0 for Arcot common components. Also, check if the JDBC jar file corresponds to the database configured in the arcotcommon.ini file against the DbType parameter.

Problem:

The upgrade tool fails with the following error:
`FATAL: ARCOT_HOME Environment Variable Not Set`

Cause:

$ARCOT_HOME is not set.

Solution:

Set the $ARCOT_HOME environment variable and run the upgrade tool.

Problem:

The upgrade tool fails with the following error:
`Error Occured: Upgrade Initialization Error:Could not create DBService instance"`

Solution:

Check if the user name and password are configured correctly in the arcotcommon.ini and securestore.enc files, respectively.

Problem:

The upgrade tool fails with the following error:
`Error Occured: Upgrade Initialization Error:Io exception: The Network Adapter could not establish the connection"`

Solution:

Check if the JDBC URL is correct and points to the correct database. Ensure that the database is up and running.

Problem:

The upgrade tool fails with the following error:
`javax.crypto.BadPaddingException: Given final block not properly padded`

Cause:

The key label that is used to encrypt the data is not the same as the key used for decryption.

Solution:

Ensure that the master key label used to encrypt data in the database is the same as the key label used by the upgrade tool to decrypt data. The master key label is stored in the securestore.enc file in the $ARCOT_HOME/conf folder.

Problem:

The upgrade tool fails with the following or similar error:
`"java.sql.SQLException: ORA-20010: -1031-ORA-01031: insufficient privileges"`

Cause:

The database user that is configured in the arcotcommon.ini file does not have sufficient database privileges to carry out the database upgrade.

Solution:

Ensure that the administrator performing the upgrade has the required database privileges. Installation time privileges are applicable to upgrade also.

Problem:

The upgrade tool fails with the following or similar error:
`"ORA-01536: space quota exceeded for tablespace"`

Cause:

The database user that is configured in the arcotcommon.ini file has exhausted the space quota in the tablespace.

Solution:

The DBA must increase the quota for the user. Restart the upgrade tool after you reimport the pre-upgrade data.

Problem:

After the upgrade process, the Administration Console fails to start and returns the following error:

```
ERROR : taglib.tiles.InsertTag : ServletException in
'/WEB-INF/jsp/dynamic/navbar_GA.jsp': File
&amp;quot;/WEB-INF/jsp/dynamic/navbar_GA.jsp&amp;quot;
```

Cause:

The Work folder of the application server where the Administration Console is deployed still contains the cache of the earlier Administration Console version.

Solution:

Clear the Work folder of the application server where the Administration Console is deployed and restart the application server.

Problem:

After the upgrade process, an administrator belonging to the LDAP repository can no longer log in to the Administration Console.

Cause:

The administrator may be disabled in LDAP.

Solution:

Ensure that the administrator is not disabled in LDAP. Disabled administrators are not allowed to log in to the Administration Console.

Problem:

After the upgrade process, for the IBM DB2 database the AuthMinder Server-specific reports fail to generate with the following error:

```
DB2 SQL Error: SQLCODE=-1585, SQLSTATE=54048, SQLERRMC=null
```

Cause:

The SYSTEM TEMPORARY tablespace page size may be set to the default value of 4K, which is not sufficient for report queries that involve columns exceeding 4K size.

Solution:

Set the SYSTEM TEMPORARY tablespace page size to a minimum of 16K. See database vendor documentation for more information.

(In Error Scenario Only) Reverting to Your Initial Setup

During upgrade, if there are any warnings during AuthMinder Server startup and if your transactions fail, then you may want to revert your setup to the earlier stage.

To revert to the initial setup:

1. Uninstall AuthMinder 7.1.01.

   See chapter, "Uninstalling AuthMinder" (see page 169) for more information.

2. Install the version that you want to revert to. For example, 5.3.x, 5.4.x, or 6.x.

   See the *CA AuthMinder Installation and Deployment Guide* that is shipped with the corresponding version.

3. Navigate to the location where ARCOT_HOME_BACKUP directory is available.
4. Copy the contents of ARCOT_HOME_BACKUP to your current $ARCOT_HOME.
5. Deploy the Administration Console and UDS.
6. Start AuthMinder Server and the application server.
7. Test the installation.

Chapter 8: Uninstall Strong Authentication

This procedure guides you through the steps to uninstall Strong Authentication and related components.

Follow these steps:

1. Based on the database type you are using, navigate to the folder:

   (For Microsoft SQL Server) <*install_location*>/arcot/dbscripts/mssql

   (For Oracle Database) <*install_location*>/arcot/dbscripts/oracle

   (For IBM DB2 UDB) <*install_location*>/arcot/dbscripts/db2

   (For MySQL) <*install_location*>/arcot/dbscripts/mysql

2. Run the scripts *in the following order:*

   a. drop-webfort-8.0.sql

      Note: (For MYSQL), Safe Updates should be disabled when running drop-webfort-8.0.sql

   b. drop-arcot-common-8.0.sql

      This deletes all the database tables.

3. Stop Strong Authentication Server.

4. Navigate to the following directory:
   `sh <install_directory>/arcot/"Uninstall_Strong Authentication"/Uninstall Strong Authentication`

5. Run the installer using the following command:
   `sh Uninstall Strong Authentication`

6. Select the type of the uninstall:

   - **1-Completely remove all features and components:** Select this option if you want to uninstall *all* components from the current system.
   - **2-Choose specific features that were installed by InstallAnywhere:** Select this option if you want to uninstall only *selected* components from the current system.

7. Press **Enter** to continue.

   If you selected to uninstall all components, then proceed to Step 8.

   If you selected to uninstall selected components, then the Choose Product Features screen appears.

8. This screen displays the components that are installed on the current system. Enter the component numbers (separated by comma) and press **Enter**.

   Important! To uninstall specific features, follow the reverse sequence in which you performed the installation. For example, if you have installed Strong Authentication Authentication Server followed by Administration Console, first uninstall Administration Console and then Strong Authentication Authentication Server.

   The Uninstall Complete screen appears at the end of successful uninstallation.

9. Press **Enter** to exit the wizard.

Uninstalling AuthMinder Schema

To uninstall AuthMinder schema from the database:

1. Based on the database type you are using, navigate to the folder:

   (For Microsoft SQL Server) <*install_location*>/arcot/dbscripts/mssql

   (For Oracle Database) <*install_location*>/arcot/dbscripts/oracle

   (For IBM DB2 UDB) <*install_location*>/arcot/dbscripts/db2

   (For MySQL) <*install_location*>/arcot/dbscripts/mysql

2. Run the scripts *in the following order:*

   a. drop-webfort-7.1.01.sql

   b. drop-arcot-common-2.0.sql

      This deletes all the database tables.

Uninstalling AuthMinder

Perform the following tasks to uninstall AuthMinder:

1. Stop AuthMinder Server.
2. Ensure that INI files are not open in any editor.
3. Navigate to the following directory:
   `prompt> cd <install_directory>/arcot/Uninstall_Arcot WebFort`
4. Run the installer using the following command:
   `prompt> sh Uninstall Arcot WebFort`

   You are prompted to specify uninstallation options.
5. Select the type of uninstallation:
   - **1-Completely remove all features and components:** Select this option if you want to uninstall *all* components of the AuthMinder on the current system.
   - **2-Choose specific features that were installed by InstallAnywhere:** Select this option if you want to uninstall only *selected* components of the AuthMinder on the current system.
6. Press **Enter** to continue.

   If you selected to uninstall all components, then proceed to Step 8.

   If you selected to uninstall selected components, then the Choose Product Features screen appears.

   The Choose Product Features screen appears.
7. (**For Uninstalling Specific Components Only**) This screen displays the AuthMinder components that are installed on the current system. Enter the component numbers (separated by comma) and press **Enter**.

   Important! To uninstall specific features, follow the reverse sequence in which you performed the installation. For example, if you have installed AuthMinder Authentication Server followed by Administration Console, first uninstall Administration Console and then AuthMinder Authentication Server.

   The Uninstall Complete screen appears at the end of successful uninstallation.
8. Press **Enter** to exit the wizard.

Post-Uninstallation Steps

The following are the post-uninstallation steps:

1. Delete the <install_location>/arcot folder.
2. Uninstall the following web applications from the application server:
   - arcotadmin - Administration Console
   - arcotuds - User Data Service
   - ca-strongauth-8.0-sample-application - Sample Application
3. Ensure that the .com.zerog.registry.xml file has been deleted. This hidden file is copied during the installation. The location of this file depends on the user account that was used to install Strong Authentication:
   - If you had installed Strong Authentication as the root user, then this file is located in the /var directory.
   - If you had installed Strong Authentication as any other user, then this file is located in the user's HOME directory.

Note: If you have performed distributed-system deployment, then locate these files on the system where you have deployed the particular application.

Appendix A: AuthMinder File System Structure

This appendix provides information about the location of all the files that are installed by the AuthMinder installer.

Important! Do not delete any of the files that are installed by AuthMinder.

Issuance and Authentication AuthMinder Server Files

The following table lists the folder location of the files that are used by AuthMinder Server:

Folder	File Description
<*install_location*>/	Contains the arcotkey and wfkey files. These file are used by the installer to detect previously installed Arcot products. If you delete these file, then the installer will not be able to detect previously installed Arcot products, and will allow new installations to be performed in any location. As a result, the installer will not be able to ensure the same destination folder for multiple Arcot products and components, in which case, the products (or components) might not work, as expected. This file has no impact on patches and upgrade. The wfdbkey and wftpkey files are the reference key files for database and third-party JAR files. These keys files are used during upgrade.
<*install_location*>/arcot/bin	Contains webfortserver script that calls the server binary in the sbin folder.

Folder	File Description
<*install_location*>/arcot/conf	Contains the following configuration files: ■ arcotcommon.ini (see page 185) ■ securestore.enc: File that contains the keys that are used for encrypting sensitive data.
<*install_location*>/arcot/dbscripts	Contains the SQL scripts to create the AuthMinder schema. See "Running Database Scripts" (see page 75) for information about the database scripts.
<*install_location*>/arcot/logs	Contains the AuthMinder Server log file.
<*install_location*>/arcot/odbc32v70wf	Contains the branded DataDirect ODBC libraries for all the databases supported by AuthMinder.
<*install_location*>/arcot/sbin	Contains library files and following executables that are required by administrators: ■ arwfutil - It is used to shut down and refresh the AuthMinder Server. ■ arwfserver.real - Contains a symbolic link to the script that sets the configuration and executes arwfutil binary. ■ arwfwatchdog - This tool monitors the server health and also starts the server if it stops. ■ arwfenv- Script that is used to set the environment variables.
<*install_location*>/samples/xml/webfort	Contains the following files used by the ArcotWebFortBulkOperationsSvc.wsdl file: ■ oath-token-assign.xml ■ oath-token-upload.xml
<*install_location*>/arcot/sdk/server/plugin/c/docs	Contains the following plug-in interface file: ■ webfort-plugin-cpp-interface.html
<*install_location*>/arcot/sdk/server/plugin/c/include/webfort/vas	Contains the following SDK plug-in header files: ■ wf-common-interface.h ■ wf-common-interface.hpp ■ wf-plugin-interface.h
<*install_location*>/arcot/sdk/server/plugin/c/lib	Contains the arwfpluginsdk.so containing the plug-in libraries.

Folder	File Description
<*install_location*>/arcot/tools/<*platform_name*>	Contains the following file: ■ **DBUtil**: Tool for editing securestore.enc, which stores the database information that is required to connect to the AuthMinder database in the encrypted format.
<*install_location*>/arcot/Uninstall_Arcot WebFort	Contains the executable required to uninstall AuthMinder.

Administration Console Files

The following table lists the folder location of the files that are used by Administration Console:

Folder	File Description
<*install_location*>/	Contains the arcotkey and wfkey files. These file are used by the installer to detect previously installed Arcot products. If you delete these file, then the installer will not be able to detect previously installed Arcot products, and will allow new installations to be performed in any location. As a result, the installer will not be able to ensure the same destination folder for multiple Arcot products and components, in which case, the products (or components) might not work, as expected. This file has no impact on patches and upgrade. The wfdbkey and wftpkey files are the reference key files for database and third-party JAR files. These keys files are used by the installer during upgrade.
<*install_location*>/arcot/conf	Contains the following configuration files: ■ arcotcommon.ini (see page 185) ■ adminserver.ini (see page 195) ■ securestore.enc: File that contains the keys that are used for encrypting the sensitive data.
<*install_location*>/arcot/conf/resourcebundles	Contains the following message properties files: ■ arcot-common-message_en_US.properties ■ arcot-uds-message_en_US.properties
<*install_location*>/arcot/dbscripts	Contains the SQL scripts to create the Administration Console schema. See "Running Database Scripts" (see page 75) for information about the database scripts.

Folder	File Description
<*install_location*>/arcot/java/lib	Contains the WAR and JAR files required by the Administration Console Framework: ■ adminframework.jar ■ adminframework.war ■ arcot-common.jar ■ arcot-crypto-util.jar ■ bcprov-jdk15-146.jar
<*install_location*>/arcot/java/lib/sdk	This is an empty directory. You have to include the JAR files that have to be used by the bundlemanager tool to create the arcotadmin.war and arcotuds.war files.
<*install_location*>/arcot/java/webapps	Contains the arcotadmin.war file required to deploy Administration Console.
<*install_location*>/arcot/logs	Contains the Administration Console log file.
<*install_location*>/arcot/java/native/<*platform_name*>/<*32 or 64 bit*>	Contains the libArcotAccessKeyProvider.so file that is used to read the contents of securestore.enc file.
<*install_location*>/arcot/odbc32v70wf	Contains the branded DataDirect ODBC libraries for all the databases supported by AuthMinder.
<*install_location*>/arcot/resourcepacks	Contains the following AuthMinder and Administration Console packages: ■ bundle_webfort.zip ■ bundler_adminconsole.zip
<*install_location*>/arcot/resourcepacks/i18n//	Contains the following properties file to update the account status value: ■ framework-useraccount-status.properties
<*install_location*>/arcot/tools/common/	Contains the following subdirectories: ■ The **arreporttool** subdirectory contains the report tool that enables you to export reports. ■ The **bundlemanager** subdirectory contains the files that are required by the Administration Console Resource pack.
<*install_location*>/arcot/tools/<*platform_name*>	Contains the following file: ■ **DBUtil**: Tool for editing securestore.enc, which stores the database information that is required to connect to the AuthMinder database in the encrypted format.

Folder	File Description
<*install_location*>/arcot/Uninstall_Arcot WebFort	Contains the uninstallation-related files.

User Data Service Files

The following table lists the folder location of the files that are used by User Data Service:

Folder	File Description
<*install_location*>/	Contains the arcotkey file. This file is used by the installer to detect previously installed Arcot products. If you delete these file, then the installer will not be able to detect previously installed Arcot products, and will allow new installations to be performed in any location. As a result, the installer will not be able to ensure the same destination folder for multiple Arcot products and components, in which case, the products (or components) might not work, as expected. This file has no impact on patches and upgrade. The wfdbkey and wftpkey files are the reference key files for database and third-party JAR files. These keys files are used during upgrade.
<*install_location*>/arcot/conf	Contains the following configuration files: ■ arcotcommon.ini (see page 185) ■ udsserver.ini (see page 197) ■ securestore.enc: File that contains the keys that are used for encrypting the sensitive data.
<*install_location*>/arcot/conf/resourcebundles	Contains the following message properties files: ■ arcot-common-message_en_US.properties ■ arcot-uds-message_en_US.properties
<*install_location*>/arcot/dbscripts	Contains the SQL scripts to create the Administration Console schema. See "Running Database Scripts" (see page 75) for information about the database scripts.
<*install_location*>/arcot/docs/uds	Contains the arcot-uds-2_0-wsdl-docs.zip file, which contains the WSDL documents for UDS.

Folder	File Description
<*install_location*>/arcot/java/lib	Contains the WAR and JAR files required by the User Data Service: ■ arcot-common.jar ■ arcot-crypto-util.jar ■ arcot-euds.jar ■ bcprov-jdk15-146.jar ■ udsframework.war
<*install_location*>/arcot/java/lib/sdk	This is an empty directory. You have to include the JAR files that have to be used by the bundlemanager tool to create the arcotadmin.war and arcotuds.war files.
<*install_location*>/arcot/java/webapps	Contains the arcotuds.war file required to deploy and User Data Service.
<*install_location*>/arcot/logs	Contains the UDS log file.
<*install_location*>/arcot/java/native/<*platform_name*>/<*32 or 64 bit*>	Contains the libArcotAccessKeyProvider.so file that is used to read the contents of securestore.enc file.
<*install_location*>/arcot/odbc32v70wf	Contains the branded DataDirect ODBC libraries for all the databases supported by AuthMinder.
<*install_location*>/arcot/tools/common/	Contains the following subdirectory: ■ The **uds-monitor** subdirectory contains the tool that enables you to monitor UDS.
<*install_location*>/arcot/tools/<*platform_name*>	Contains the following file: ■ **DBUtil**: Tool for editing securestore.enc, which stores the database information that is required to connect to the AuthMinder database in the encrypted format.
<*install_location*>/arcot/Uninstall_Arcot WebFort	Contains the uninstallation-related files.
<*install_location*>/arcot/wsdls/uds	Contains the following documents: ■ WSDL files that the Web Services client uses to generate the code: ArcotConfigManagementSvc.wsdl ArcotOrganizationManagementSvc.wsdl ArcotUserManagementSvc.wsdl ■ XSD files that are used by Web Services to perform bulk operations: ArcotUserSchema.xsd

Authentication Java SDK Files

The following table lists the folder location of the files that are used by Authentication Java SDK:

Folder	File Description
<*install_location*>/arcot/docs/webfort	Contains the Arcot-WebFort-7.1.01-authentication-sdk-javadocs.zip file, which contains the Javadocs for Authentication SDK.
<*install_location*>/arcot/samples/java	Contains the webfort-7.1.01-sample-application.war file to deploy Sample Application.
<*install_location*>/arcot/sdk/client/java/lib/arcot	Contains the following JAR files for AuthMinder Authentication Java SDK. ■ arcot-webfort-common.jar ■ arcot-webfort-authentication.jar
<*install_location*>/arcot/sdk/client/java/lib/external	Contains the third-party JAR files required by AuthMinder Authentication Java SDK. ■ bcprov-jdk15-146.jar ■ commons-pool-1.5.5.jar
<*install_location*>/arcot/sdk/client/java/properties	Contains the sample properties (webfort.authentication.properties) file. You can either use the parameters of this file for initializing the Java SDKs or use the init() function.

Issuance Java SDK Files

The following table lists the folder location of the files that are used by Issuance Java SDK:

Folder	File Description
<*install_location*>/arcot/docs/webfort	Contains the Arcot-WebFort-7.1.01-issuance-sdk-javadocs.zip file, which contains the Javadocs for Issuance SDK.
<*install_location*>/arcot/samples/java	Contains the webfort-7.1.01-sample-application.war file to deploy Sample Application.
<*install_location*>/arcot/sdk/client/java/lib/arcot	Contains the following JAR files for Issuance Java SDK. ■ arcot-webfort-common.jar ■ arcot-webfort-issuance.jar

Folder	File Description
<*install_location*>/arcot/sdk/client/java/lib/external	Contains the third-party JAR files required by AuthMinder Issuance Java SDK. ■ bcprov-jdk15-146.jar ■ commons-pool-1.5.5.jar
<*install_location*>/arcot/sdk/client/java/properties	Contains the sample properties (webfort.issuance.properties) file. You can either use the parameters of this file for initializing the Java SDKs or use the init() function.

Web Services Files

The following table lists the location of the files that are related to the UDS Web Services:

Folder	File Description
<*install_location*>/arcot/docs/uds	Contains the arcot-uds-2_0-wsdl-docs.zip file, which contains the WSDL documents for UDS.
<*install_location*>/arcot/docs/webfort	Contains the following WSDL documents: ■ Arcot-WebFort-7.1.01-admin-wsdldocs.zip ■ Arcot-WebFort-7.1.01-authentication-wsdldocs.zip ■ Arcot-WebFort-7.1.01-issuance-wsdldocs.zip
<*install_location*>/arcot/wsdls/uds	Contains the following documents: ■ WSDL files that the Web Services client uses to generate the code: ArcotConfigManagementSvc.wsdl ArcotOrganizationManagementSvc.wsdl ArcotUserManagementSvc.wsdl ■ XSD files that are used by Web Services to perform bulk operations: ArcotUserSchema.xsd

Folder	File Description
<*install_location*>/arcot/wsdls/webfort	Contains the following documents: ■ WSDL files that the Web Services client uses to generate the code: ArcotWebFortAdminSvc.wsdl ArcotWebFortAuthSvc.wsdl ArcotWebFortBulkOperationsSvc.wsdl ArcotWebFortIssuanceSvc.wsdl ■ XSD files that are used by Web Services to perform bulk administration and credential operations: ArcotWebFortAdminMsgs.xsd ArcotWebFortAdminSchema.xsd ArcotWebFortAuthMsgs.xsd ArcotWebFortAuthSchema.xsd ArcotWebFortCommonSchema.xsd ArcotWebFortCredMgmt.xsd ArcotWebFortIssuanceMsgs.xsd ArcotWebFortIssuanceSchema.xsd ArcotWebFortTokenXchange.xsd

Plug-In SDK

The following table lists the folder location of the files that are used by plug-in SDK:

Folder	File Description
<*install_location*>/arcot/sdk/server/plugin/c/lib	Contains the arwfpluginsdk.so containing the plug-in libraries.
<*install_location*>/arcot/sdk/server/plugin/c/include/webfort/vas	Contains the following SDK plug-in header files: ■ wf-common-interface.h ■ wf-common-interface.hpp ■ wf-plugin-interface.h
<*install_location*>/arcot/sdk/server/plugin/c/lib	Contains the arwfpluginsdk.so containing the plug-in libraries.

Appendix B: Configuration Files and Options

This topic discusses the configuration files that AuthMinder uses and the parameters that you must configure in these files.

The following AuthMinder configuration files are available in the <*install_location*>/arcot/**conf** location:

- arcotcommon.ini (see page 185)
- adminserver.ini (see page 195)
- udsserver.ini (see page 197)

The following properties files are available in the <*install_location*>/arcot/sdk/client/java/**properties**/ location:

- webfort.authentication.properties (see page 199)
- webfort.issuance.properties (see page 201)

INI Files

arcotcommon.ini

The arcotcommon.ini file contains the parameters for database and instance settings for the AuthMinder Server and other components (Administration Console and User Data Service) of AuthMinder. This section discusses these parameters of arcotcommon.ini file:

- Parameters Used by AuthMinder Server (see page 185)
- Parameters Used by Administration Console and User Data Service (see page 189)

Parameters Used by AuthMinder Server

The following table lists the database and encryption settings that are used by AuthMinder Server. Additional database configurations for the AuthMinder Server must be performed using the Instance Management screen of the Administration Console.

Parameter	Default	Description
Common Database Parameters in[arcot/db/dbconfig] Section		

Parameter	Default	Description
DbType	No default	The type of the database applicable to all database connections. The supported values are: ■ mssqlserver ■ oracle ■ db2 ■ mysql
EnableBrandLicensing	0	Whether a branded ODBC driver is in use. This can be used when you are using the branded ODBC drivers from DataDirect.
BrandLicenseFile	No Default	The license file name when you use a branded ODBC driver.
StartWithAnyPool	1 (enabled)	Enables AuthMinder to start with the backup database when the primary database is unavailable.
Primary and Backup Database Connection Parameters in [arcot/db/primarydb] and [arcot/db/backupdb] Sections		
Datasource.*N*	No Default	The name of the ODBC System Data Source Name (DSN) pointing to the primary database hosting the server data.
Username.*N*	No Default	The User Name used by the server to access the database.
Encryption Mode Setting Parameter in [arcot/crypto/device] Section		
HSMDevice	s/w	The mode that sets whether the AuthMinder information must be encrypted with a key stored in the database or with one in a Hardware Security Module (HSM). Supported values are: ■ **s/w**: Indicates that the data will be encrypted with the key label stored in the database. ■ **chrysalis**: Indicates the Chrysalis (Luna) HSM will be used to encrypt the data. ■ **nfast**: Indicates nFast (nCipher netHSM) will be used to encrypt the data.
Chrysalis (Luna) HSM Configuration Parameters in [crypto/pkcs11modules/chrysalis] Section		

Parameter	Default	Description
sharedLibrary	No Default	The absolute path to the PKCS#11 shared library corresponding to the HSM. The default value for Chrysalis (Luna) is: /usr/lunasa/lib/libCryptoki2.so
storageSlot	0	The HSM slot where the 3DES keys used for encrypting the data are present.
accelSlot	0	The slot for internal use by Arcot.
sessionCount	20	The maximum number of sessions that can be established with the HSM device.
nFast (nCipher netHSM) HSM Configuration Parameters in [crypto/pkcs11modules/nfast] Section		
sharedLibrary	No Default	The absolute path to the PKCS#11 shared library corresponding to the HSM. The default value for nFast (nCipher netHSM) is: /opt/nfast/toolkits/pkcs11/libcknfast.so
storageSlot	1	The HSM slot where the 3DES keys used for encrypting the data are present.
accelSlot	0	The slot for internal use by Arcot.
sessionCount	200	The maximum number of sessions that can be established with the HSM device.
Watchdog Configurations in [arcot/watchdog] Section		
ServerStartsTimeout	25	The time period from the Server startup. If watchdog brings up the Server for 5 times within the specified duration of ServerStartsTimeout (25 minutes), then the Server is not restarted again. The time is in minutes.
ServerStartsCount	5	The maximum count for restarting the Server. After this, the Server is not be restarted again.
RestartSleepTime	5000	The sleep time after which watchdog restarts the Server. The sleep time is in milliseconds.

Changing Server Startup Logging Parameters

If you want to change the logging parameters that you see when AuthMinder Server starts up, then:

1. Navigate to the conf directory in ARCOT_HOME.
2. Open arcotcommon.ini in a text editor of your choice.
3. Add the following section at the end of the file:

```
[arcot/WebFort/startup]
LogFile=
LogFileSize=10485760
BackupLogFileDir=
LogLevel=
LogTimeGMT=0
```

The following table explains these parameters:

Parameter	Default	Description
LogFile		The file path to the default directory and the file name of the log file. **Note:** This path is relative to ARCOT_HOME (<*install_location*>/arcot/).
LogFileSize	10485760	The maximum number of **bytes** the log file can contain. When a log file reaches this size, a new file is started and the old file is moved to the location specified for BackupLogFileDir.
BackupLogFileDir		The location of the directory where backup log files are maintained, after the current file exceeds LogFileSize bytes. **Note:** This path is relative to ARCOT_HOME (<*install_location*>/arcot/).
LogLevel		The default logging level for the server, unless an override is specified. The possible values are: ■ 0 FATAL ■ 1 WARNING ■ 2 INFO ■ 3 DETAIL

Parameter	Default	Description
LogTimeGMT	0	The parameter which indicates the time zone of the time stamp in the log files. The possible values are: ■ 0 Local Time ■ 1 GMT

1. Set the required values for the parameters that you want to change.
2. Save and close the file.
3. Restart AuthMinder Server.

Parameters Used by Administration Console and User Data Service

The following table describes the database setting parameters in the arcotcommon.ini file:

Parameter	Default	Description
Common Database Parameters in[arcot/db/dbconfig] Section		
DbType	No default	The type of database applicable to all database connections. The supported values are: ■ oracle ■ mssqlserver ■ db2 ■ mysql
Driver	No default	The fully-qualified name of the JDBC driver class that is supplied by the JDBC driver vendor. Consult your JDBC vendor documentation for the right driver name. ■ **For Oracle Database** - oracle.jdbc.driver.OracleDriver ■ **For Microsoft SQL Server** - com.microsoft.sqlserver.jdbc.SQLServer Driver ■ **For IBM DB2 UDB** - jcom.ibm.db2.jcc.DB2Driver ■ **For MySQL** - com.mysql.jdbc.Driver

Parameter	Default	Description
MinConnections	4	The minimum number of connections to initially create between AuthMinder components and the database.
MaxConnections	32	The maximum number of connections that can be created between AuthMinder components and the database.
IncConnections	2	The number of connections that are created when a new connection is needed between AuthMinder components and the database.
MaxIdleConnections	4	The maximum number of idle database connections that server can maintain.
MaxWaitTimeForConnection	30000	Maximum amount of time (in milliseconds) the Server must wait (when there are no available connections) for a connection to become available, before timing out.
AutoRevert	1	Specifies whether or not the system attempts to connect to the primary database after a failover occurs. Set AutoRevert=1 if you have a backup database configured and if you want the server to try to connect to the primary database after a failover occurs.
MaxTries	3	The number of times the server will attempt to connect to the database before aborting the connection.
ConnRetrySleepTime	100	The number of **milliseconds** to delay between attempts to connect to the database.
MonitorSleepTime	50	The amount of time in **seconds** the Monitoring Thread sleeps between heartbeat checks on all databases.
Profiling	0	Specifies whether to log the database messages. Set the value to 1 if you want to enable logging of database messages.
EnableBrandLicensing	0	Specifies whether a branded ODBC driver is in use.

Parameter	Default	Description
BrandLicenseFile	IVWF.LIC	The license file name when you use a branded ODBC driver. This parameter is required if the value of EnableBrandLicensing is 1. Otherwise it is ignored. If present, this value must *not* be edited.
MaxTransactionRetries	3	The maximum number of times the transaction is retried with a database instance for pre-defined error conditions.
TransactionRetrySleepTime	10	The interval in **milliseconds** between two consecutive transaction retries.
Primary and Backup Database Connection Parameters in [arcot/db/primarydb] and [arcot/db/backupdb] Sections		
Datasource.*N*		The name of the ODBC System Data Source Name (DSN) pointing to the primary database hosting the server data.

Parameter	Default	Description
AppServerConnectionPoolName.*N*	No default	The JNDI name used to look up the connection pool object, if the database connection pooling feature of the application server is being used. A pool by this JNDI name should be created in the containing application server, and sufficient access right must be given to AuthMinder Web applications for it to use the connection pool: ■ If the JNDI name is configured in **Apache Tomcat**, then use a fully qualified JNDI name. For example: AppServerConnectionPoolName.1=java:comp/env/SampleDS ■ For **other application servers**, specify only the JNDI name. For example: AppServerConnectionPoolName.1=SampleDS See appendix, "Configuring Application Server" for more information. If the application server connection pool is *not* required, then leave this configuration empty.
URL.*N*	No default	The name of the JDBC data source. ■ For Oracle Database - jdbc:oracle:thin:@<*server*>:<*oracle_port*>:<*sid*> ■ For Microsoft SQL Server - jdbc:sqlserver://<*server*>:<*sql_port*>;databaseName=<*databasename*>;selectMethod=cursor ■ For IBM DB2 UDB - jdbc:db2://<*server*>:<*db2_port*>/<*database*> ■ For MySQL - jdbc:mysql://<*server*>:<*mysql_port*>/<*database*>

Parameter	Default	Description
Username.*N*	No default	The User Name used by the server to access the database.
TrustStorePath.*N*	No default	The SSL certificate truststore path corresponding to Datasource.*N*. The path (including the filename) refers to the certificate truststore file, which contains the list of certificates that the client trusts. **Note:** The password corresponding to TrustStore Path.*N* must to be securely stored in securestore.enc with value of TrustStorePath.*N* as the key by using DBUtil tool. See *CA AuthMinder Administration Guide* for more information about this tool.
KeyStorePath.*N*	No default	**Note:** This attribute is used only for MySQL. If you want to configure one-way SSL between RiskMinder and a MySQL Database, this is one of the parameters for which you must specify a value. This parameter holds the SSL Certificate Keystore Path corresponding to Datasource.*N*. The path (including the filename) refers to the certificate keystore file. The password corresponding to KeyStorePath.*N* must be securely stored in securestore.enc with the value of KeyStorePath.*N* as the key.
HostNameInCertificate.*N*	No default	The value of Common Name (CN) in the subject Distinguished Name (DN) of Datasource.*N* SSL certificate in truststore. **Note:** This parameter is required *only* if you are using Microsoft SQL Server.

Parameter	Default	Description
	Encryption Mode Setting Parameter in [arcot/crypto/device] Section	
HSMDevice	s/w	The mode that sets whether the information must be encrypted with a key stored in the database or with one in a Hardware Security Module (HSM). Supported values are: ■ **s/w**: Indicates that the data will be encrypted with the key label stored in database. ■ **chrysalis**: Indicates the Chrysalis (Luna) HSM will be used to encrypt the data. ■ **nfast**: Indicates nFast (nCipher netHSM) will be used to encrypt the data.
Chrysalis (Luna) HSM Configuration Parameters in [crypto/pkcs11modules/chrysalis] Section		
sharedLibrary	No Default	The absolute path to the PKCS#11 shared library corresponding to the HSM. The default value for Chrysalis (Luna) is: /usr/lunasa/lib/libCryptoki2.so
storageSlot	0	The HSM slot where the 3DES keys used for encrypting the data are present.
accelSlot	0	The slot for internal use by Arcot.
sessionCount	20	The maximum number of sessions that can be established with the HSM device.
nFast (nCipher netHSM) HSM Configuration Parameters in [crypto/pkcs11modules/nfast] Section		
sharedLibrary	No Default	The absolute path to the PKCS#11 shared library corresponding to the HSM. The default value for nFast (nCipher netHSM) is: /opt/nfast/toolkits/pkcs11/libcknfast.so
storageSlot	1	The HSM slot where the 3DES keys used for encrypting the data are present.
accelSlot	0	The slot for internal use by Arcot.

Parameter	Default	Description
sessionCount	200	The maximum number of sessions that can be established with the HSM device.
Instance ID Configuration Parameter in [arcot/system] Section		
InstanceId	1	A parameter that can be used to identify Administration Console or User Data Service instance. It is mandatory that you provide unique values for every instance of the server. The instance ID is also displayed in the transaction reports. You must specify an integer value for this parameter.

adminserver.ini

The adminserver.ini file contains the parameters to set the Administration Console log information. The following table lists the log file information of Administration Console.

Parameter	Default Value	Description
Log Configuration Parameters in [arcot/admin/logging] Section		
log4j.rootCategory	ERROR, roothandle **Important!** roothandle is the name of the Administration Console log handle and *must* be specified.	The root logger that resides at the top of the logger hierarchy. All children loggers inherit this value, if no value is specified.

Parameter	Default Value	Description
■ log4j.logger.com.arcot.euds ■ log4j.logger.com.arcot.admin ■ log4j.logger.com.arcot.admin.framework ■ log4j.logger.com.arcot.adminconsole ■ log4j.logger.com.arcot.common.cache ■ log4j.logger.com.arcot.common.crypto ■ log4j.logger.com.arcot.crypto.impl.SecureStoreUtil ■ log4j.logger.com.arcot.common.database ■ log4j.logger.com.arcot.common.ldap	INFO	Specify the log level that must be used to write Administration Console logs. The supported log levels are: ■ FATAL ■ WARNING ■ INFO ■ DEBUG **Note:** See *CA AuthMinder Administration Guide* for more information about the log levels.
log4j.appender.roothandle.Encoding	UTF-8	The encoding to use when writing the entries in the log file.
log4j.appender.roothandle.File	${arcot.home}/logs/arcotadmin.log	The log file name and the location where the Administration Console logs will be created. By default, the Administration Console log file name is arcotadmin.log and is created in the following location: <*install_location*>/arcot/**logs**/
log4j.appender.roothandle.MaxFileSize	10 MB	The maximum allowed file size of the log file.
log4j.appender.roothandle.MaxBackupIndex	100	The maximum number of backup files that can be created. When the number of backup files reaches this number, then the application starts to overwrite from the first log file.

Parameter	Default Value	Description
log4j.appender.roothandle.layout	org.apache.log4j.PatternLayout	The output format, as specified by ConversionPattern.
log4j.appender.roothandle.layout.ConversionPattern	%d{yyyy-MM-dd hh:mm:ss,SSS z} : [%t] : %-5p : %-5c{3} : %m%n	The format in which the Administration Console log file entries are written: ■ Time Stamp (%d{yyyy-MM-dd hh:mm:ss,SSS z} :) ■ Thread ID ([%t] :) ■ Log Level (or Severity) (%-5p :) ■ Logger Class (%-5c{3} :) ■ Message (%m%n) This pattern is similar to the C language printf function.

udsserver.ini

The udsserver.ini file contains the parameters to set the User Data Service (UDS) log information. The following table lists the log file information of UDS:

Parameter	Default Value	Description
Log Configuration Parameters in [arcot/uds/logger] Section		
log4j.rootCategory	ERROR, debuglog	The root logger that resides at the top of the logger hierarchy. All children loggers inherit this value, if no value is specified.

Parameter	Default Value	Description
■ log4j.logger.com.arcot.euds ■ log4j.logger.com.arcot.crypto.impl.SecureStoreUtil ■ log4j.logger.com.arcot.common.database ■ log4j.logger.com.arcot.common.cache	INFO	Specify the log level that must be used to write UDS logs. The supported log levels are: ■ FATAL ■ WARNING ■ INFO ■ DEBUG **Note:** See *CA AuthMinder Administration Guide* for more information about the log levels.
log4j.appender.debuglog.File	${arcot.home}/logs/arcotuds.log	Specify the log file name and the location where the UDS logs must be written to. By default, the UDS log file name is arcotuds.log and is created in the logs folder present in <*install_location*>/arcot/.
log4j.appender.debuglog.MaxFileSize	10MB	Specify the size of the log file. By default, it is 2 MB.
log4j.appender.debuglog.MaxBackupIndex	100	Specify the number of backup files that can be created. When the number of backup files is equivalent to this number, the application starts to overwrite from the first log file.
log4j.appender.debuglog.layout	org.apache.log4j.PatternLayout	The output format, as specified by ConversionPattern.
log4j.appender.debuglog.Encoding	UTF-8	The encoding to use when writing the entries in the log file.

Parameter	Default Value	Description
log4j.appender.debuglog.layout.ConversionPattern	%d{yyyy-MM-dd hh:mm:ss,SSS z} : [%t] : %-5p : %-5c{3} : %m%n	The format in which the UDS log file entries are written: ■ Time Stamp (%d{yyyy-MM-dd hh:mm:ss,SSS z} :) ■ Thread ID ([%t] :) ■ Log Level (or Severity) (%-5p :) ■ Logger Class (%-5c{3} :) ■ Message (%m%n) This pattern is similar to the C language printf function.

Properties Files

webfort.authentication.properties

The webfort.authentication.properties file provides the parameters for the Authentication Java SDK to read AuthMinder Server information. The following table lists the configuration parameters.

By default, the configuration parameters are appended with **.1**, which indicate that these configurations are for the primary AuthMinder Server. If you have multiple instances of AuthMinder Server and want to enable failover, then duplicate the sections that are based on the number of servers you plan to support and configure the parameters accordingly.

Parameter	Default	Description
pool.maxActive	64	Maximum number of connections allowed in the pool from the SDK to the AuthMinder Server.
pool.maxIdle	16	The maximum number of idle connections allowed in the pool from the SDK to the AuthMinder Server.
pool.maxWaitTimeMillis	-1	The maximum amount of time (in milliseconds) that a request will wait for the connection. Default -1 indicates that the thread will wait for infinite time.

Parameter	Default	Description
pool.minEvictableIdleTimeMillis	-1	The minimum amount of time a connection might be idle in the pool before it is evicted by the idle connection evictor (if any).
pool.timeBetweenEvictionRunsMillis	-1	The amount of time in milliseconds to wait before checking the pool to evict the idle connections.
authentication.host.*n*	localhost	Host name or the IP address of AuthMinder Server.
authentication.port.*n*	9742	Port number configured for the Authentication Native protocol.
authentication.transport.*n*	tcp	To enable the SSL communication between AuthMinder Authentication SDK and AuthMinder Server set this parameter to 1SSL or 2SSL. **Note:** If you change the transport mode to SSL, then restart AuthMinder Server.
authentication.connectionTimeout.*n*	10000	Maximum time in milliseconds before the AuthMinder Server is considered unreachable.
authentication.readTimeout.*n*	30000	The maximum time in milliseconds allowed for a response from AuthMinder Server.
authentication.serverCACertPEMPath.*n*	No Default	Provide the path for the CA certificate file of the server. The file *must* be in PEM format. Provide the complete path for the file. For example: server.CACertPEMPath=<*%SystemDrive%*>/certs/webfort_ca.pem
authentication.clientCertKeyP12Path.*n*	No Default	Provide the path for the client certificate, which is in the p12 format.
authentication.clientCertKeyPassword.*n*	No Default	Enter the client key pair password to open the p12 file.

webfort.issuance.properties

The webfort.issuance.properties file provides the parameters for the Issuance Java SDK to read AuthMinder Server information. The following table lists the configuration parameters.

By default, the configuration parameters are appended with **.1**, which indicate that the configurations are for the primary AuthMinder Server. If you have multiple instances of AuthMinder Servers and want to enable failover, then duplicate the sections that are based on the number of servers you plan to support and configure the parameters accordingly.

Parameter	Default	Description
pool.maxActive	64	Maximum number of connections allowed in the pool from the SDK to the AuthMinder Server.
pool.maxIdle	16	The maximum number of idle connections allowed in the pool from the SDK to the AuthMinder Server.
pool.maxWaitTimeMillis	-1	The maximum amount of time (in milliseconds) that a request will wait for the connection. Default -1 indicates that the thread will wait for infinite time.
pool.minEvictableIdleTimeMillis	-1	The minimum amount of time a connection might be idle in the pool before it is evicted by the idle connection evictor (if any).
pool.timeBetweenEvictionRunsMillis	-1	The amount of time in milliseconds to wait before checking the pool to evict the idle connections.
issuance.host.*n*	localhost	Host name or the IP address of AuthMinder Server.
issuance.port.*n*	9744	Port number configured for the Transaction Web Services protocol.
issuance.transport.*n*	tcp	To enable the SSL communication between AuthMinder Issuance SDK and AuthMinder Server set this parameter to 1SSL or 2SSL. **Note:** If you change the transport mode to SSL, then restart AuthMinder Server.
issuance.connectionTimeout.*n*	10000	Maximum time in milliseconds before the AuthMinder Server is considered unreachable.
issuance.readTimeout.*n*	30000	The maximum time in milliseconds allowed for a response from AuthMinder Server.

Parameter	Default	Description
issuance.serverCACertPEMPath.*n*	No Default	Provide the path for the CA certificate file of the server. The file *must* be in PEM format. Provide the complete path for the file. For example: server.CACertPEMPath=<*%SystemDrive%*>/certs/webfort_ca.pem
issuance.clientCertKeyP12Path.*n*	No Default	Provide the path for the client certificate, which is in p12 format.
issuance.clientCertKeyPassword.*n*	No Default	Enter the client key pair password to open the p12 file.

Appendix C: Changing HSM Configurations

This appendix lists the steps that you must perform, if you want to change the Hardware Security Module (HSM) configurations that you have specified during installation.

Note: Before proceeding with the configurations explained in this section, ensure that you have set up the HSM server and client, and generated the 3DES key in the HSM. See "(Optional, Only If You are Using HSMs) Requirements for HSM" (see page 62) for more information.

As mentioned in "Hardware Security Module (HSM) Requirements" (see page 38), AuthMinder now supports Hardware Security Module (HSM) to secure your data. If you choose to encrypt the data using HSM, then the data stored in the database is encrypted with the key that resides in the HSM.

AuthMinder supports Luna and nCipher netHSM for data encryption using hardware. The configurations for the HSMs are available in the arcotcommon.ini file. This file provides separate sections for configuring the required HSM, which in the current release are:

- Luna HSM ([crypto/pkcs11modules/chrysalis])
- nCipher netHSM ([crypto/pkcs11modules/nfast])

Based on the HSM you are configuring, specify the sharedLibrary parameter in the corresponding section. After specifying the HSM information, re-create the securestore.enc file with the HSM key label, initialize the HSM, and then initialize AuthMinder to use the HSM key.

To change the HSM information that AuthMinder needs:

1. Navigate to the following location:
   `<install_location>/arcot/conf`
2. Take a backup of the securestore.enc file.
3. Delete the existing securestore.enc file from *<install_location>*/arcot/conf.

4. To change the HSM information that AuthMinder needs:
   a. Navigate to the following location:
   <*install_location*>/arcot/conf
   b. Open arcotcommon.ini in a text editor.
   c. Ensure the HSMDevice parameter in the [arcot/crypto/device] section is set to the HSM that you plan to use:
      - chrysalis for Luna HSM.

      or

      - nfast for nCipher netHSM.
   d. Depending on the HSM that you are configuring, set the sharedLibrary parameter to the location where the HSM library file is located.

      For Luna (libCryptoki2.so) and for nCipher netHSM (libcknfast.so), enter the absolute path and full name of the file.

      Note: See "arcotcommon.ini" (see page 185) for more information about the other HSM configuration parameters available in this section.
   e. Save and close the arcotcommon.ini file.
5. Navigate to the following location, where the DBUtil tool is available:
   <*install_location*>/arcot/tools/<*platform_name*>
6. Run the DBUtil tool with the following commands:
   a. dbutil -init <*HSM_key_label*>

      Note: The <*HSM_key_label*> corresponds to the 3DES key that resides in the HSM.

      The preceding command creates a securestore.enc file with the specified key label. The generated file in stored in the <*install_location*>/arcot/conf location.
   b. dbutil -i <*HSM_module_name*> <*HSM_password*>

      Note: The <*HSM_module_name*> is chrysalis for Luna HSM, and nfast for nCipher netHSM.

      The preceding command initializes the HSM.
   c. dbutil -pi <*DSN_Name*> <*Database_password*> -h <*HSM_password*> -d <*HSM_module_name*>

      Note: <*DSN_NAME*> refers to the ODBC DSN that AuthMinder Server uses to connect to the AuthMinder database. <*Database_password*> refers to password used to connect to the database.

      The preceding command initializes the AuthMinder Server data to be encrypted using HSM.

d. dbutil -pi <*Database_Username*> <*Database_password*> -h <*HSM_password*> -d <*HSM_module_name*>

Note: <*Database_Username*> refers to the user name used to connect to the AuthMinder database. The database user name is case-sensitive, therefore ensure that you provide the correct value. <*Database_password*> refers to password used to connect to the database.

The preceding command initializes the Administration Console and the User Data Service data to be encrypted by using HSM.

Appendix D: Database Reference

AuthMinder database contains a number of tables, some of which grow with increased usage. Some tables grow in direct relation to the number of users, while others grow in direct relation to the usage of the product. Also, a user accessing the system multiple times will cause the tables to grow. Because of restricted disk space, as a database administrator managing AuthMinder deployments, you may not want these tables to grow indefinitely. In this case, you can use the information in this appendix to trim some tables to manage your disk space and improve the database performance.

Trim only the tables that capture transaction details, such as audit log information. Do *not* trim tables that capture user information, which is necessary to authenticate users.

Note: It is recommended that you make appropriate adjustments to the SQL databases based on the configuration and the need for reporting data. For example, deleting a large volume of data will adversely impact performance during the delete process. Depending on the size of the rollback segments, this may even cause the system to fail. It is also recommended that you archive older records and not delete them completely.

This appendix provides information about the following topics:

AuthMinder Database Tables

This section briefly explains all the database tables:

Used by AuthMinder

The following table lists the database tables that are used by the AuthMinder Server:

Table Name	Description
ARWFADMINAUDITLOG	Contains the audit log information for the AuthMinder administration activities.

Table Name	Description
ARWFARCOTEMV	Contains the CA AuthID OTP-EMV credentials for the users. It contains an individual entry for each user.
ARWFARCOTEMVHISTORY	Contains all the CA AuthID OTP-EMV credentials that are in the *reissue* state.
ARWFARCOTID	Contains the CA AuthID credentials for the users. It contains an individual entry for each user.
ARWFARCOTIDHISTORY	Contains all the CA AuthIDs that are in the *reissue* state.
ARWFARCOTOTP	Contains the CA AuthID OTP-OATH credentials for the users. It contains an individual entry for each user.
ARWFARCOTOTPHISTORY	Contains all the CA AuthID OTP-OATH credentials that are in the *reissue* state.
ARWFAUTHAUDITLOG	Contains the audit log information for the authentication activities.
ARWFAUTHTOKENS	Contains the authentication tokens that are generated after a successful authentication. One entry is made in this table for every successful authentication irrespective of the type of token requested.
ARWFCONFIG	Contains the AuthMinder configuration information. The information in this table contains version information and therefore has multiple entries per configuration.
ARWFDBERRORCODES	Contains the database error codes that indicate the communication failure.
ARWFDBQUERIES	Contains the list of database queries used by the AuthMinder Server.
ARWFDISPLAYNAMES	Contains names and values of different keys used in AuthMinder.
ARWFGENERICCRED	Contains the information about the miscellaneous credentials of the user. For example, the credentials supported by custom APIs.
ARWFGENERICCREDHISTORY	Contains all the miscellaneous (for example, custom API supported) credentials that are in *re-issue* state.
ARWFINSTANCES	Contains information about all the instances of AuthMinder Servers that communicate with a specific database.

Table Name	Description
ARWFISSUANCEAUDITLOG	Contains the audit log information for the credential issuance activities.
ARWFMESSAGES	Contains the messages that are posted by the AuthMinder Server.
ARWFMODULEREGISTRY	Contains information about the internal modules of the AuthMinder Server and about the plug-ins.
ARWFOATH	Contains the OATH One-Time Password (OTP) credentials of the users. It contains an individual entry for each user.
ARWFOATHHISTORY	Contains all the OATH OTP credentials that are in *re-issue* state.
ARWFOATHTOKENREGISTRY	Contains the OATH token details such as, seed value, token ID, and token type.
ARWFORGACTIVECONFIG	Contains configuration mapping of the currently active organization. The information in this table contains version information and therefore has multiple entries per configuration.
ARWFORGCONFIG	Contains configuration mapping per organization. The information in this table contains version information and therefore has multiple entries per configuration.
ARWFOTP	Contains the One-Time Password (OTP) credentials for the users. It contains an individual entry for each user.
ARWFPASSWORD	Contains the user name-password credentials for the users. It contains an individual entry for each user.
ARWFPASSWORDHISTORY	Contains all the user-name password credentials that are in *re-issue* state.
ARWFPROTOCOLCONFIGURATION	Contains configuration of each listener port of the AuthMinder Server.
ARWFQNA	Contains the Question and Answer (QnA) credentials for the users. It contains an individual entry for each user.
ARWFQNAHISTORY	Contains all the question and answer credentials that are in *re-issue* state.
ARWFSEQUENCE	Contains information about sequences used for version configurations.

Table Name	Description
ARWFSSLTRUSTSTORE	Contains the SSL root certificates that are trusted by the AuthMinder Server.
ARWFSVRMGMTAUDITLOG	Contains the audit log information for the server management activities.
ARWFUNIQUEFIELDS	Enforces uniqueness of certain fields in the configuration. For example, a RADIUS client IP address cannot be used by two organizations.
ARWFVERIFIEDCHALLENGES	Contains information about the challenges for which the CA AuthID signature is successfully verified. An entry is made for successful CA AuthID PKICA AuthIDCA AuthID authentication, provided No Replay for challenge is turned on. It is turned off by default.

Used by Administration Console

The following table lists all database tables that are used by the Administration Console:

Table Name	Description
ARADMINAUDITTRAIL	Stores administrator activity audit.
ARADMINAUTHTOKEN	Stores the tokens that Administration Console uses for pluggable authentication. Every time you log in to the Console by using password, a token is internally generated after password match and stored in this table.
ARADMINBASICAUTHPWDHISTORY	Stores the last *n* occurrences of password of all administrators in all organizations that use Basic Authentication (for administrators) to log in to the Administration Console. This information is stored to prevent password reuse.
ARADMINBASICAUTHUSER	Stores the basic authentication credentials of all administrators in all organizations that use Basic Authentication (for administrators) to log in to the Administration Console.
ARADMINCONFIG	Stores the Administration Console configurations.
ARADMINCUSTOMROLE	Stores the configurations for all custom-defined roles.
ARADMINMANAGEROLE	Stores the list of roles that a specified role can manage.

Table Name	Description
ARADMINMAP	Stores the information of the AuthMinder Server instance, which is entered as a key-value pair.
ARADMINPAFCONFIG	Stores the authentication configurations for all administrators in all organizations in the system.
ARADMINPREDEFINEDROLE	Stores the role information for all supported administrators.
ARADMINPWDPOLICY	Stores the details of password policies for all administrators in all organizations.
ARADMINROLEPRIVILEGE	Stores the mapping of all administrative actions (or tasks) supported by the Administration Console, the scope of each task, and which role can perform the task.
ARADMINSCOPE	Stores the list of organizations over which each administrator has control (scope).
ARADMINSCOPEALL	Stores the list of all administrators who have control (scope) over *all* the existing organizations in the system.
ARADMINSUPPORTEDAUTHMECH	Stores the information about all supported authentication mechanisms to log in to the Administration Console.
ARADMINSUPPORTEDTIMEZONE	Stores the list of all available time zones. **Note:** This is an internal table.
ARADMINTURNEDOFFPRIVILEGE	Stores the list of the privileges that are not available for the given custom role.
ARADMINTXID	Stores the information required to generate a unique ID for each transaction.
ARADMINUITAB	Stores the information about the tabs that are available and the order in which they are available in the Administration Console.
ARADMINUITASK	Stores the information about all the tasks that are available and the order in which they are available through the Administration Console.
ARADMINUITASKATTRIBUTES	Stores the details of the tasks that are displayed, when the first-level and the second-level tabs in the Administration Console are clicked. These tasks are referred to as landing pages.

Table Name	Description
ARADMINUITASKCONTAINER	Stores the information related to available *task containers*. A task container can either be a second-level tab ID or the task group in the Administration Console.
ARADMINUSER	Stores the detailed information (such as organization to which they belong, current status, time zone, locale, last login time) of all existing administrators.
ARADMINUSER_ARCHIVE	Stores the information related to all deleted users.
ARADMINWIZARDTASK	Stores the information about all the tasks that can be performed by using the Bootstrap Wizard.
ARCMNBULKOPERATION	Stores information related to all supported bulk operations that include uploading users and uploading user accounts.
ARCMNBULKOPERATIONATTRIBUTE	Stores attributes for all bulk operations in the ARCMNBULKOPERATION table.
ARCMNBULKREQUEST	Stores details (such as organization name, Request ID, status of the request, data uploaded, and operation) for each bulk-upload request.
ARCMNBULKTASKPARAM	Stores the name and the value of each attribute for each task supported in the system.
ARCMNBULKUPLOADTASK	Stores the status of each task for every bulk-upload request.
ARCMNCACHEREFRESH	Stores cache-related housekeeping information that indicates whether the Administration Console needs to be refreshed or not.
ARCMNCONFIG	Stores common Administration Console configuration information. Some of these include whether Bootstrap is complete, whether the cache refresh is automatic or manual, whether attribute encryption is enabled, and whether the bulk upload feature is enabled or not.
ARCMNDBERRORCODES	Stores vendor-specific database error codes and SQL state values that signify whether the database is down or non-responsive. This information is used by the system to decide if database should be failed over, in case a backup database is configured.
ARCMNMAPDATATYPE	Stores CA product-specific information that the Administration Console uses for rendering the Console pages.

Table Name	Description
ARCMNKEY	Stores the key configurations used while creating or updating organizations.
ARPFCMNCACHEREFRESHEVENT	Stores details of all cache refresh events for all instances in the system.
ARPFCMNCACHEREFRESHSCOPE	Stores information about all organizations that will be affected if a server cache refresh event occurs.
ARPFCMNCACHEREFRESHSTATUS	Stores the status of each cache refresh event for every instance for which it was triggered.
ARPFCMNINSTANCE	Stores detailed information for all AuthMinder Server instances configured in the system. This also includes the last time the instance was refreshed.
ARPFCMNORGCONFIGDATA	Stores configuration details for each organization. This includes global configurations that can be, typically, overridden at the organization-level.
ARPFCMNORGCONFIGSTATE	Stores the status of each assigned configuration from the ARPFCMNORGCONFIGDATA table.
ARPFCMNPRIVILEGEMAPPING	Stores the details of each privilege available through the Administration Console.
ARREPORTTABLES	Stores the information about the reports that are generated using Administration Console.
ARSEQUENCETABLE	Simulates sequences using stored procedures. **Note:** This table is used *only* by MS SQL Server.

Used by User Data Service

The following table explains the database tables that are used by UDS:

Table Name	Description
ARUDSACCOUNTTYPE	Stores the details of all account types that are configured in the system.
ARUDSATTRMAP	Stores the configuration details that describe the field names of custom attributes for accounts, specific to each organization.
ARUDSAUTHSESSION	Stores authentication session details for currently active sessions. If this table is not replicated, then active authentication sessions can be lost.

Table Name	Description
ARUDSCALLOUT	Stores user-specific Callout configurations. These Callouts are called, if configured, for specific events, such as user creation and update.
ARUDSCALLOUTINTERNAL	Stores the configuration information (SDK method to be invoked), for Callouts when a delete event with cascade effect is triggered or enabled.
ARUDSCALLOUTINTERNALPARAMS	Stores the details, such as parameters and types specific to internal Callouts.
ARUDSCALLOUTPARAM	Stores the details, such as parameters and types specific to external Callouts.
ARUDSCONFIG	Stores the UDS configuration parameters and their values.
ARUDSCONFIGAUDITLOG	Stores the audit log information for the User Data Source (UDS) operations and their return status.
ARUDSCONTACTTYPE	Stores additional contact information (such as secondary email and telephone number) that can be configured at the organization or global level.
ARUDSCUSTOMATTREXT	Stores the additional user account custom attributes. By default, up to 10 user account custom attributes are stored in the ARUDSUSERACCOUNT table. Any additional attributes (after the first 10) are stored in this table.
ARUDSCUSTOMATTREXT_ARCHIVE	Stores the archived information related to user account custom attributes when a user account is deleted.
ARUDSLDAPREPOSITORYCONFIG	Stores the LDAP Repository configurations, such as LDAP host and port details.
ARUDSORGANIZATION	Stores organization definitions, their attributes and repository connectivity details.
ARUDSORGANIZATIONAUDITLOG	Stores detailed organization-specific UDS audit logging information.
ARUDSORGREPOATTRIBUTES	Stores the organization-specific repository mapping information. For example, if you are using LDAP as the user repository, then a CA attribute (say FNAME) might be mapped to a corresponding LDAP attribute (say GIVENNAME).

Table Name	Description
ARUDSORGSECUREATTRIBUTES	Stores organization-specific attributes that need to be encrypted, such as PII fields. **Note:** These attributes are configured using the Administration Console.
ARUDSREPOCLONESTATUS	Stores the status of temporary cloning of user information from an external repository (such as LDAP) to the ARUDSREPOSITORYUSER table.
ARUDSREPOSITORYTYPES	Stores the definitions of all repositories supported by UDS.
ARUDSREPOSITORYUSER	Temporarily stores user information from an external repository (such as LDAP) to increase performance. This is typically done when user data for a large number of users must be retrieved from the external repository.
ARUDSRESOURCESCOPE	Stores the resource-to-organization mapping. In other words, this table specifies which resource is applicable for which organizations. For example, specific account types might be applicable only for specific organizations.
ARUDSRESOURCESCOPEALL	Stores the resource-to-organization mapping. However, it is different from the ARUDSRESOURCESCOPE table, because it specifies which resource is applicable for *all* organizations.
ARUDSSECUREATTRIBUTES	Stores information related to attributes that need to be encrypted, such as PII fields. **Note:** These attributes are configured using the Administration Console.
ARUDSUSER	Stores user details and attributes of all users who belong to the organization.
ARUDSUSER_ARCHIVE	Stores user details for all user accounts that have been deleted from the system.
ARUDSUSERACCOUNT	Stores user account information for specific users.
ARUDSUSERACCOUNT_ARCHIVE	Stores user account information for all user accounts that have been deleted from the system.
ARUDSUSERATTRIBUTE	Stores all user attribute definitions. This table is expected to change rarely, only when new user attributes are added by individual products.

Table Name	Description
ARUDSUSERAUDITLOG	Stores user operation-specific detailed audit logging information.
ARUDSUSERCONTACT	Stores the secondary contact information (such as email or telephone numbers) for users.
ARUDSUSERCONTACT_ARCHIVE	Stores the secondary contact information (such as email or telephone numbers) for the user accounts that have been deleted from the system.

Database Sizing Calculations

This section helps the database administrator calculate the approximate size of the database that is to be set up for AuthMinder.

Denotations Used in Sample Calculations

The following denotations are used in the calculations:

- Number of users = *N*
- Average number of transactions per day = *T*
- Computation timeframe (in days) = *D*

Value Assumptions

The following assumptions have been made for calculation purposes:

- Number of users (**N**) = 1,000,000 (one million)
- Average number of transactions per day (**T**) = 24,000
- Computation timeframe (**D**) = 90 days

Sample Calculation Based on Assumptions

Considering the figures that are assumed in section, "Value Assumptions" (see page 216) the final requirement should be:

- Based on **total number of users**, the database size = (21 * *N*) KB
- Based on **daily activity,** the database size = (*T* * *D* * 5) KB

Database Tables Replication Advice

This section provides information about how frequently the tables must be replicated between the primary and the backup databases. It covers the following topics:

Tables That Need Real-Time Synchronization

The following table lists the database tables that need real-time synchronization between the primary and the backup databases. This category mainly includes the tables that contain user-related information and this data is required for authentication. Therefore, perform real-time synchronization of these tables.

AuthMinder Component	Table
Administration Console	ARADMINAUDITTRAIL
	ARADMINBASICAUTHUSER
	ARADMINSCOPE
	ARADMINSCOPEALL
	ARADMINUSER
	ARADMINTXID
	ARPFCMNINSTANCE
	ARSEQUENCETABLE
User Data Service	ARUDSORGANIZATION
	ARUDSORGREPOATTRIBUTES
	ARUDSORGSECUREATTRIBUTES
	ARUDSLDAPREPOSITORYCONFIG
	ARUDSACCOUNTTYPE
	ARUDSRESOURCESCOPE
	ARUDSRESOURCESCOPEALL
	ARUDSATTRMAP
	ARUDSCONTACTTYPE

AuthMinder Component	Table
	ARUDSUSER
	ARUDSUSERACCOUNT
	ARUDSCUSTOMATTREXT
	ARUDSAUTHSESSION
	ARUDSUSERCONTACT
	ARUDSREPOSITORYUSER
WebFort Server	ARWFARCOTEMV
	ARWFARCOTID
	ARWFARCOTOTP
	ARWFAUTHTOKENS
	ARWFINSTANCES
	ARWFGENERICCRED
	ARWFOATH
	ARWFOTP
	ARWFQNA
	ARWFPASSWORD
	ARWFVERIFIEDCHALLENGES

Tables That Need Periodic Synchronization

The following table lists the database tables that need periodic synchronization between the primary and the backup databases. These database tables are synchronized when there is any change in the configurations.

Component	Table
Administration Console	ARADMINCONFIG
	ARADMINCUSTOMROLE
	ARADMINMAP
	ARADMINPAFCONFIG
	ARADMINPWDPOLICY
	ARADMINBASICAUTHPWDHISTORY

Component	Table
	ARADMINTURNEDOFFPRIVILEGE
	ARADMINCACHEREFRESH
	ARADMINAUDITTRAIL
	ARADMINUSER_ARCHIVE
	ARADMINMANAGEROLE
	ARADMINROLEPRIVILEGE
	ARPFCMNORGCONFIGDATA
	ARPFCMNORGCONFIGSTATE
	ARPFCMNCACHEREFRESHSTATUS
	ARPFCMNCACHEREFRESHEVENT
	ARPFCMNCACHEREFRESHSCOPE
	ARCMNBULKTASKPARAM
	ARCMNBULKUPLOADTASK
	ARCMNBULKREQUEST
	ARCMNBULKOPERATIONATTRIBUTE
	ARCMNBULKOPERATION
UDS	ARUDSUSERAUDITLOG
	ARUDSORGANIZATIONAUDITLOG
	ARUDSCONFIGAUDITLOG
	ARUDSCONFIG
	ARUDSREPOSITORYTYPES
	ARUDSUSERATTRIBUTE
	ARUDSUSERACCOUNT_ARCHIVE
	ARUDSCUSTOMATTREXT_ARCHIVE
	ARUDSUSER_ARCHIVE
	ARUDSUSERCONTACT_ARCHIVE
	ARCMNCONFIG
	ARUDSREPOCLONESTATUS
	ARUDSCALLOUTINTERNAL

Component	Table
	ARUDSCALLOUTINTERNALPARAMS
	ARUDSCALLOUT
	ARUDSCALLOUTPARAM
WebFort	ARWFADMINAUDITLOG
	ARWFARCOTEMVHISTORY
	ARWFARCOTIDHISTORY
	ARWFARCOTOTPHISTORY
	ARWFAUTHAUDITLOG
	ARWFCONFIG
	ARWFGENERICCREDHISTORY
	ARWFISSUANCEAUDITLOG
	ARWFMODULEREGISTRY
	ARWFOATHHISTORY
	ARWFOATHTOKENREGISTRY
	ARWFORGACTIVECONFIG
	ARWFORGCONFIG
	ARWFPASSWORDHISTORY
	ARWFPROTOCOLCONFIGURATION
	ARWFQNAHISTORY
	ARWFSEQUENCE
	ARWFSSLTRUSTSTORE
	ARWFSVRMGMTAUDITLOG
	ARWFUNIQUEFIELDS

Tables That Do Not Need Synchronization

The following table lists the database tables that do not need any synchronization between the primary and backup databases:

Component	Table
Administration Console	ARCMNDBERRORCODES
	ARADMINAUTHTOKEN
	ARADMINMANAGEROLE
	ARADMINPREDEFINEDROLE
	ARADMINSUPPORTEDAUTHMECH
	ARADMINSUPPORTEDTIMEZONE
	ARADMINUITAB
	ARADMINUITASK
	ARADMINUITASKATTRIBUTES
	ARADMINUITASKCONTAINER
	ARADMINWIZARDTASK
	ARCMNMAPDATATYPE
	ARCMNCACHEREFRESH
	ARPFCMNPRIVILEGEMAPPING
	ARREPORTTABLES
User Data Service	ARUDSSECUREATTRIBUTES
WebFort	ARWFDBERRORCODES
	ARWFDBQUERIES
	ARWFDISPLAYNAMES
	ARWFMESSAGES

Database Tuning Parameters

The following table lists the common parameters that you can use to tune the connection between AuthMinder Server and the database. These configurations are made using the Instance Management screen of the Administration Console.

Field	Description
Minimum Connections	Enter the minimum number of connections that will be created between AuthMinder Server and the database when the server starts up.
Maximum Connections	Enter the maximum number of connections that can be created between the AuthMinder Server and the database. **Note:** Set this value depending on the maximum connections that the database supports, because this overrides the MaxConnections parameter. See your database vendor documentation for more information.
Increment Connections By	Enter the number of connections that will be added to the existing connections at a time, when the need arises. **Important!** The total number of connections *cannot* exceed the maximum number of connections.
Monitor Thread Sleep Time (in Seconds)	Enter the time for which the monitoring thread will sleep between successive heartbeat checks for all databases.
Monitor Thread Sleep Time in Fault Condition (in Seconds)	Enter the interval at which the database monitor thread will check the health of the connection pool in case of faulty database connections.
Log Query Details	Enable this option if you want to log the details for all database queries.
Monitor Database Connectivity	Enable checking of the pools proactively in the database monitor thread.
Auto-Revert to Primary	Enable this option if you want server to switch from the backup database to the primary database when the primary database becomes available again after a failover condition.

Chapter 9: Strong Authentication Configuration for Oracle RAC

Perform the steps in this section only if you want to use Oracle RAC with Strong Authentication.

Modify the Database Script

You run the database scripts as a post-installation task in the Strong Authentication installation procedure. Before you run this script, modify it for Oracle RAC.

Follow these steps:

1. To determine the Oracle RAC shared datafile path, log in to the database and run the following command:

   ```
   SELECT file_name, tablespace_name FROM dba_data_files
   ```

   The following is sample output of this command:

   ```
   +DATA/qadb/datafile/users.259.797224649      USERS
   +DATA/qadb/datafile/undotbs1.258.797224649   UNDOTBS1
   +DATA/qadb/datafile/sysaux.257.797224647     SYSAUX
   ```

2. Open the arcot-db-config-for-common-8.0.sql file. This file is in the install_location/arcot/dbscripts/oracle/ directory.

3. Search for the following line in the file:

   ```
   filename varchar2(50) := 'tabspace_arreports_'|| to_char(current_timestamp,
   'YYYY-MM-DD-HH24-MI-SS') || '.dat';
   ```

4. Replace that line with the following line:

   ```
   filename varchar2(100) :=
   '+shared_location/service_name/datafile/tabspace_arreports_'||
   to_char(current_timestamp, 'YYYY-MM-DD-HH24-MI-SS') || '.dat';
   ```

   In the new line:

   - Replace shared_location with the shared datafile path that you determined by running the command given in the first step.
   - Replace service_name with the service name of the Oracle RAC installation.

   The following is a sample line:

   ```
   filename varchar2(100) := '+DATA/forwardinc/datafile/tabspace_arreports_'||
   to_char(current_timestamp, 'YYYY-MM-DD-HH24-MI-SS') || '.dat';
   ```

5. Save and close the script file, and then run it.

Configure the JDBC URL

Specify the JDBC URL in the format supported by Oracle RAC in the arcotcommon.ini file.

Follow these steps:

1. Open the arcotcommon.ini file in a text editor. This file is in the install_location/arcot/conf/ directory.
2. Specify a value for the URL parameter in the [arcot/db/primarydb] section and, if required, in the [arcot/db/backupdb] section of the INI file. Enter the URL in the following format:

   ```
   URL.1=jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=host_name)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=service_name)(SERVER=DEDICATED)))
   ```

   For example:

   ```
   URL.1=jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=172.30.250.18)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=forwardinc)(SERVER=DEDICATED)))
   ```

   Note: If Oracle RAC is client configured, then include all the nodes in this format.
3. If the database user that you specified while running the Strong Authentication installer is different from the database user in Oracle RAC, then:
   a. Change the database user credentials in the arcotcommon.ini file.
   b. Use DBUtil to change the database user credentials in the securestore.enc file. DBUtil is available in the ARCOT_HOME/tools/*<platform_name>* directory. Instructions on using DBUtil are given in Updating the securestore.enc File and Setting the TrustStore Password (see page 156).
4. Save and close the arcotcommon.ini file.

Update the odbc.ini File

The odbc.ini file contains connection parameters. For Oracle RAC, you must specify values pertaining to the Oracle RAC installation in the odbc.ini file.

Follow these steps:

1. Create a *.ora file on the system on which you have installed Strong Authentication. For example, /var/opt/tns.ora.
2. Enter the following lines in the file:

```
section_name =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = host_name_or_IP_address)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = service_name)
    )
  )
```

   For example:

```
fwdincrac =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = 172.30.250.18)(PORT = 1521))
    )
    (CONNECT_DATA =
      (SERVICE_NAME = forwardinc)
    )
  )
```

   Note: If Oracle RAC is client configured, then include all the nodes in this format.
3. Save the file.
4. Open the *ARCOT_HOME*/odbc32v60wf/odbc.ini file in a text editor.
5. For the required DSN sections, comment out the lines containing the following parameters:
   - HostName
   - PortNumber
   - SID

   For example:

```
#HostName=172.30.251.251
#PortNumber=1521
#SID=an
```

6. Add the following parameters:

```
TNSNamesFile=ARCOT_HOME/ora_file_name
ServerName=section_name
```

For example:

```
TNSNamesFile=/var/opt/tns.ora
ServerName=fwdincrac
```

7. Save and close the file.

Appendix E: Additional Configurations for CA Adapter 2.2.7

To integrate Strong Authentication with your Adapter 2.2.7 instance, you must first perform certain additional configurations. Perform the steps in this section only after all the Strong Authentication components are installed and running properly.

This section contains the following topics:

Update the CA Adapter 2.2.7 Instance

Strong Authentication includes the Arcot-Adpater-2.2.7-Compatibility-Package.zip file. You copy the contents of this file to your existing Adapter installation.

Follow these steps:

1. Extract the contents of the Strong Authentication-Adpater-2.2.7-Compatibility-Package.zip file to a temporary location.

   You see the following structure:

```
arcotsm
    WEB-INF
        lib
            arcot-common.jar
            log4j-1.2.15.jar

arcotafm
    client
        arcotjsclient_jso.js
      vpn
             controller_vpn.jsp
WEB-INF
        classes
            jspStrings_en.properties
dbscripts
    mssql
        arcot-db-config-for-adapter-statemanager.sql
        drop-adapter-statemanager.sql
    oracle
        arcot-db-config-for-adapter-statemanager.sql
        drop-adapter-statemanager.sql
```

2. Update State Manager as follows:
   a. Navigate to the location on the application server where you deployed State Manager 2.2.7, and delete the arcot-common-1.0.9.jar file.

      For example, in Apache Tomcat this location is *TOMCAT_HOME*/arcotsm/WEB-INF/lib.

   b. From the extracted file structure, copy the files arcot-common.jar and log4j-1.2.15.jar to this location.

   c. Restart the application server.
3. Update Authentication Flow Manager as follows:
   a. From the extracted file structure, copy the arcotjsclient_jso.js file to the arcotafm/client folder on the application server where you deployed Authentication Flow Manager 2.2.7.

      For example, in Apache Tomcat this location is *TOMCAT_HOME*/arcotafm/client.

   b. From the extracted file structure, copy the jspStrings_en.properties file to the arcotafm/WEB-INF/classes folder on the application server where you deployed Authentication Flow Manager 2.2.7. For example, in Apache Tomcat this location is *TOMCAT_HOME*/arcotafm/WEB-INF/classes.
4. Perform the following steps on the database schema:
   a. If the State Manager and Strong Authentication instances are using different databases, then drop the table ARCMNDBERRORCODES in the database that is used by State Manager.

   b. Run the part of the script that creates ARCMNDBERRORCODES table and inserts rows into it.

LDAP Plugin Registration

Perform the steps in this section only in the following scenarios:

- You performed a new installation of Strong Authentication and want to use it with Adapter 2.2.7.
- You upgraded from versions 6.2.x to 7.x, did not register the LDAP plugin, and then upgraded.
- You upgraded from 6.2.x to 7.1.01, but did not register the LDAP plugin in the previous releases.

CA Adapter 2.2.7 uses a plugin to enable LDAP authentication using Strong Authentication. To verify that the LDAP plugin used in Adapter 2.2.7 works with Strong Authentication, you must register it.

Register the LDAP Plugin

Register the LDAP plugin using the Administration Console.

Follow these steps:

1. Log in to the Administration Console as a master administrator.
2. Click the Services and Server Configurations tab.
3. Click the Strong Authentication subtab, and on the left pane, under Extensible Configurations, select Plug-In Registration.

   The Register Plug-In screen appears in the right pane.
4. Provide appropriate values for the following fields:
   - Name: A name for the plugin.
   - Handler Path: arwfldapauthplugin.dll.
   - Configuration template: Select the path to the file ldapauthplugin-config.xml in the file system. Typically, this file is located in the *Install_Location*/arcot/samples/xml/webfort directory.
   - Move UP_AUTH from the Available Events list to the Supported Events list.
5. Click the Register button.

Configure the Plugin for an Organization

You can now configure the registered plugin for different organizations.

Follow these steps:

1. Log in to the Administration Console as a global administrator.
2. Click the Organizations tab, and search for the organization for which you want to use the plugin.

   Note: The organization that you select here must be mapped to LDAP in the AFM Wizard.

   The Organization information screen appears.
3. Click the Strong Authentication Configuration subtab, and on the left pane, under Extensible Configurations, select Plug-In Configurations.

   The Configure Plug-In screen appears.
4. Select the registered LDAP authentication plugin from the Name list.
5. Move UP_AUTH from the Supported Events list to the Selected Events list.
6. Click Submit.

   A message appears indicating that the plugin was configured successfully.

Appendix F: Configuring Alternate Schema for IBM DB2 Universal Database

IBM DB2 Universal Database (UDB) enables you to use a different schema other than the default schema. This schema is referred to as alternate schema. This appendix explains the steps that you must perform to set up and use the alternate schema. It covers the following topics:

Creating Schema

Perform the following steps to set up the alternate schema:

1. Log in to IBM DB2 UDB database.

   If the login name is arcotuser, then the default schema for this user is arcotuser.

2. Create an alternate schema.

   For example, the alternate schema can be arcotalternateuser.

   Note: See IBM DB2 UDB database documentation for more information how to create the alternate schema.

3. Run the following queries to set up the alternate schema (arcotalternateuser):
   ```
   set current schema ARCOTALTERNATEUSER
   set current path ARCOTALTERNATEUSER
   ```

   Note: The alternate schema *must* be specified in upper case.

4. Navigate to the following location:
   `<install_location>/arcot/dbscripts/db2`

5. Run the *scripts in the following order*:

   a. arcot-db-config-for-common-2.0.sql

   b. arcot-db-config-for-webfort-7.1.01.sql

   Note: If you encounter any error while executing the scripts, then check with your database administrator whether you have the required privileges.

6. Ensure that the database user name (arcotuser)and the corresponding password are set in the securestore.enc file. You can use the DBUtil tool to insert the database user name and password.

   See chapter, "Tools for System Administrators" in *CA AuthMinder Administration Guide* for more information.

Editing Configuration File

If you plan to use alternate schema for IBM DB2 UDB, then edit the arcotcommon.ini file to configure these settings.

Perform the following steps:

1. Navigate to the following locations:
   `<install_location>/arcot/conf`
2. Open the arcotcommon.ini file in a text editor.
3. In the [arcot/db/primarydb] and [arcot/db/backupdb] sections, set the following parameters:
   - **URL.1**: Set this parameter to the JDBC data source, as follows:
     `jdbc:db2://<server>:<database_port>/<database>:currentSchema=<alternateID>;currentFunctionPath="SYSIBM","SYSFUN","SYSPROC","SYSIBMADM","<alternateID>";`

     For example
     `jdbc:db2://db2server:50000/arcotdb:currentSchema=ARCOTALTERNATEUSER;currentFunctionPath="SYSIBM","SYSFUN","SYSPROC","SYSIBMADM","ARCOTALTERNATEUSER";`
   - **Username.1**: Set this parameter to the user name that is used to access the database. For example, arcotuser.
4. Save and close the arcotcommon.ini file.

Configuring ODBC DSN

AuthMinder Server uses the Open Database Connectivity (ODBC) to connect to the database. Edit the ODBC settings for AuthMinder Server to connect to the database using the alternate schema.

Perform the following steps to edit the ODBC settings:

1. Navigate to the following location:

   <install_location>/arcot/odbc32v70wf
2. Open the odbc.ini file in a file editor.
3. In the [*<Database_name>*] section that corresponds to the database you are using, edit the parameters that are listed in the following table:

Parameter	Description
AlternateID	The alternate schema created in Step 2.

Parameter	Description
CurrentFuncPath	Set this field to SYSIBM, SYSFUN, SYSPROC, SYSIBMADM, *<Alternate_ID>*. For example, SYSIBM, SYSFUN, SYSPROC, SYSIBMADM, ARCOTALTERNATEUSER.

4. Save and close the odbc.ini file.

Appendix G: Default Port Numbers and URLs

This appendix lists the default port numbers that AuthMinder uses. It contains the following sections:

Default Port Numbers

AuthMinder supports four protocols that are configured at different ports. The following table lists the default port numbers that are used by AuthMinder:

Protocol	Default Port Number	Default Status	Description
Server Management Web Services	9743	Enabled	This protocol is used for managing AuthMinder server. Administration Console and arwfutil clients communicate using this port for server management activities.
Transaction Web Services	9744	Enabled	This protocol is used by the Authentication and the Issuance Web Services clients to connect to AuthMinder Server.
Transaction Native	9742	Enabled	This is a proprietary, binary protocol used by AuthMinder for the purpose of authentication and credential issuance. This port is used by Authentication and Credential Management SDKs.
Administration Web Services	9745	Enabled	This protocol is used to create and manage configurations, such as profiles, policies, SAML, and ASSP configurations.

Protocol	Default Port Number	Default Status	Description
RADIUS	1812	Disabled	This is used to support the Remote Authentication Dial In User Service (RADIUS) protocol. When configured to support RADIUS protocol, AuthMinder server acts as a RADIUS server.
ASSP	9741	Disabled	This protocol is used with Adobe® Reader and Adobe® Acrobat® to authenticate user for server-side digital signing of the PDF documents.
Transaction HTTP	9746	Disabled	This protocol is used to transfer the HTTP request packets from the HTTP Client to the AuthMinder Server.

If some other service is already running on the default port, then set a new port for the protocols as follows:

- **To set a new port number for Server Management Web Services protocol**, use webfortserver tool.

  Note: See the topic titled "Tools for System Administrators" in *CA AuthMinder Administration Guide* for more information about the tool.

- **To set a new port number for other protocols**, use Protocol Configuration screen in the Administration Console.

  Note: See the topic titled "Managing WebFort Server Instances" in *CA AuthMinder Administration Guide* for more information about changing the port numbers.

URLs for AuthMinder Components

Use the URLs listed in the following table to access AuthMinder components after installation:

Component or Service	URL
Administration Console URL for Master Administrator	*http://<Apphost>:<app_server_port>/arcotadmin/masteradminlogin.*htm **Note: Apphost** refers to the system where Administration Console is deployed. **App_Server_Port** refers to the port number of the application server on which the Administration Console is deployed.

Component or Service	URL
Administration Console URL for other administrators	*http://<Apphost>:<app_server_port>/arcotadmin/adminlogin.htm* **Note: Apphost** refers to the system where Administration Console is deployed. **App_Server_Port** refers to the port number of the application server on which the Administration Console is deployed.
Organization Management Web Service	http://*<Apphost>:<app_server_port>/arcotuds/services/ArcotUserRegistryMgmtSvc* **Note: Apphost** refers to the system where User Data Service (UDS) Console is deployed. **App_Server_Port** refers to the port number of the application server on which UDS is deployed.
User Management Web Services	http://*<Apphost>:<app_server_port>/arcotuds/services/ArcotUserRegistrySvc* **Note: Apphost** refers to the system where User Data Service (UDS) Console is deployed. **App_Server_Port** refers to the port number of the application server on which UDS is deployed.
Configuration Registry Web Service	http://*<Apphost>:<app_server_port>/arcotuds/services/ArcotConfigRegistrySvc* **Note: Apphost** refers to the system where User Data Service (UDS) Console is deployed. **App_Server_Port** refers to the port number of the application server on which UDS is deployed.
Credential Management Web Services	http://*<Apphost>:<Protocol_port>/WebFortIssuanceSvc* **Note: Apphost** refers to the system where AuthMinder Server is installed. **Protocol Port** refers to the port number of the Transaction Web Services protocol. By default, this value is 9744.
Administration Web Service	http://*<Apphost>:<Protocol_port>/ArcotWebFortAdminSvc* **Note: Apphost** refers to the system where AuthMinder Server is installed. **Protocol Port** refers to the port number of the Administration Web Services protocol. By default, this value is 9745.

Component or Service	URL
Authentication Web Services	http://*<Apphost>:<Protocol_port>/WebFortAuthSvc* **Note: Apphost** refers to the system where AuthMinder Server is installed. **Protocol Port** refers to the port number of the Transaction Web Services protocol. By default, this value is 9744.
Bulk Operation Web Service	http://*<Apphost>:<Protocol_port>/WebFortBulkOperationsSvc* **Note: Apphost** refers to the system where AuthMinder Server is installed. **Protocol Port** refers to the port number of the Administration Web Services protocol. By default, this value is 9745.
Sample Application	*http://<Apphost>:<app_server_port>/webfort-7.1.01-sample-application/* **Note: App_Server_Port** refers to the port number of the application server on which the Sample Application is deployed.

Appendix H: Configuring Application Server

This appendix covers the following topics:

Enabling Database Connection Pooling

Typically, accessing the database may not be a bottleneck, but setting up a new connection for each request can be an overhead and can reduce the performance of the system. By creating a database connection pool, you can avoid potential bottlenecks. This is because connection pooling helps you avoid the overhead of making a new database connection every time a AuthMinder component that is deployed on the application server requires access to the database.

This section covers the configuration steps for the following application servers:

Apache Tomcat

This section provides the steps to enable Apache Tomcat for JNDI-based database operations.

Perform the following steps to create a JNDI connection in Apache Tomcat:

1. Install Apache Tomcat, and test the installation using the following URL:

   http://localhost:8080/

   The preceding URL must open the Apache Tomcat home page.

2. Open the server.xml file present in the <*TOMCAT-HOME*>/conf directory.

3. Collect the following information for defining a data source:

   - JNDI Name

     The JNDI name that is used by the product components. This name must match with the AppServerConnectionPoolName.*N* in arcotcommon.ini (without the java:comp/env/ prefix).

   - User ID

     The database user ID.

   - Password

     The database password.

   - JDBC Driver Class

     The JDBC driver class name. For example, oracle.jdbc.driver.OracleDriver.

   - JDBC URL

     The JDBC URL for the database server. For example, if you are using Oracle driver, then URL would be: jdbc:oracle:thin:@<*server*>:<*database_port*>:<*sid*>.

4. Add the following entry to define the datasource within the <*GlobalNamingResources*> tag:

```
<Resource name="SampleDS"
auth="Container"
type="javax.sql.DataSource"
factory="org.apache.tomcat.dbcp.dbcp.BasicDataSourceFactory"
username="<userid>"
password="<password>"
driverClassName="<JDBC driver class>"
url="<jdbc-url>"
maxWait="30000"
maxActive="32"
maxIdle="8"
initialSize="4"
timeBetweenEvictionRunsMillis="300000"
minEvictableIdleTimeMillis="30000"/>
```

5. Open the context.xml file present in the *<TOMCAT-HOME>*/conf directory.
6. Add the following entry to define the datasource within the <Context> tag:
   `<ResourceLink global="SampleDS" name="SampleDS" type="javax.sql.DataSource"/>`
7. Copy the following database connection pooling (DBCP) dependencies to *<TOMCAT-HOME>*/common/**lib** directory.
   - commons-dbcp-1.2.2.jar
   - ojdbc14-10.2.0.1.0.jar (for Oracle Database)
   - sqljdbc.jar (Microsoft JDBC driver for Microsoft SQL Server 2005 - version 1.2.2828)
   - db2jcc-9.5.jar (for IBM DB2 UDB)
   - mysql-connector-java-5.1.22-bin.jar (for MySQL)

Configuration changes for Tomcat8 (apache-tomcat-8.0.24) and JDK8 (1.8.0_51)

1. Create JNDI connection

   There are few configuration attribute names that have been updated in tomcat 8. Couple of them are listed in the sample here. Verify your attribute names from tomcat 8 documentation if you are using any others which are not shown in the sample.

   <Resource name="< data source_name >" auth="Container"

   type="javax.sql.DataSource" username="USER_ID" password="PASSWORD"

   driverClassName="JDBC_Driver_Class " url="JDBC_url" maxWaitMillis="30000"

   maxTotal="32" maxIdle="4" initialSize="4"

   timeBetweenEvictionRunsMillis="600000"

   minEvictableIdleTimeMillis="600000"/>

   Note :

   The **maxActive** configuration option has been renamed to **maxTotal**

   The **maxWait** configuration option has been renamed to **maxWaitMillis**
2. Make sure you use the latest database jar.
   - For sql server - sqljdbc4.jar
   - For oracle ojdbc6.jar
3. Enable SSLv3

   Java 8 disables SSLv3 by default. To enable it follow these steps:

   1. Go to “<JRE_HOME>\lib\security” folder used by tomcat.

2. Open java.security file.
3. Check if the property “jdk.tls.disabledAlgorithms” has SSLv3, if yes remove the SSLv3 value.

IBM WebSphere Application Server

This section provides the steps to enable IBM WebSphere for JNDI-based database operations:

Perform the following steps to configure IBM WebSphere for deploying Java-dependent components of AuthMinder:

1. Log in to WebSphere Administration Console.
2. Click **Resources** and expand the **JDBC** node.
3. Click **JDBC Providers**.

   The JDBC Providers page appears.
4. In the **Preferences** section, click **New**.

   The Create a new JDBC Provider page appears.
5. Perform the following steps to create a JDBC provider:

   Note: See http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/topic/com.ibm.websphere.base.iseries.doc/info/iseries/ae/tdat_ccrtprov.html for more information about JDBC providers.

   a. Specify the **Database Type** and **Provider Type**.

   b. Select **Connection pool data source** from the **Implementation Type** drop-down list.

   c. Enter a **Name** for the JDBC provider. You can also enter a **Description** for the JDBC provider.

   d. Click **Next**.

      The Enter database class path information screen appears.

   e. Enter the absolute path for the JAR file.

   f. Click **Next**.

      The Summary screen appears.

   g. After reviewing the summary of the information that you have entered, click **Finish**.
6. Set the CLASSPATH for the JDBC provider that you created in Step 5.

   a. Click **Resources** and expand the **JDBC** node.

   b. Click **JDBC Providers**.

      The JDBC Providers page appears.

   c. Click the JDBC provider that you created in Step 5.

   d. Set the **Class Path** for the JDBC JAR.

   e. Click **Apply** to save the changes.

7. Create a Data Source, as follows:
   a. Go to **Resources**, and then click **JDBC**.
   b. Under **JDBC**, open **Data Sources** and click **New**. Perform the following steps to create a data source:
   c. Specify the **Data source name**.
   d. Specify the **JNDI name**. This name must match with the AppServerConnectionPoolName.*N* in arcotcommon.ini.
   e. Click **Next.**
   f. Select the JDBC provider that you created in Step 3.
   g. Click **Next.**

      The Enter database specific properties for the data source screen appears.
   h. Depending on the database, enter the following information:
      - **For Oracle**:

        Specify the **Value** for JDBC URL. This URL would be of the following type:
        `jdbc:oracle:thin:@<server>:<oracle_port>:<sid>`

        Select the **Data store helper class name**.
      - **For Microsoft SQL Server:**
        `jdbc:sqlserver://<server>:<sql_port>;databaseName=<database name>;selectMethod=cursor`
      - **For IBM DB2:**
        `jdbc:db2://<server>:<db2_port>/<database>`
      - **For MySQL:**
        `jdbc:mysql://<server>:<mysql_port>/<database>`
   i. Click **Next**.

      The Setup Security aliases screen appears.
   j. Click **Next** to view the Summary screen, and then click **Finish**.
8. Click the data source that you created in Step 7.
9. In the **Related Items** section, click **JAAS - J2C authentication data**.
10. Click **New** to create a credential.
11. Enter login credentials that are used to connect to the database and save the credential.
12. Click **Apply**, and then click **OK** to save the changes.
13. Click **Data Sources** and select the data source that you created in Step 7.
14. Under **Security Settings** -> **Component-managed authentication alias**, select the JAAS credential that you created in Step 11 and click **Apply**, and then **OK**.

15. Click **Data Sources** and select the check box for the data source you created in Step 7.
16. Click **Test connection** to verify that you have specified the connection correctly.

Note: This test only checks the connection to the database server, not necessarily the correct definition of the data source.

Oracle WebLogic Server

This section provides the steps to enable Oracle WebLogic for JNDI-based database operations.

Perform the following steps to create a data source in Oracle WebLogic:

1. Log in to WebLogic Administration Console.
2. Click the **Lock & Edit** button, if it is not done.
3. Go to **Resources**, and then click **JDBC**.
4. Under **JDBC**, open **Data Sources** and click **New** to create a data source. Perform the following steps to create a data source.
5. Set the following JNDI and the database information:
    a. Set **Name** = ArcotDB.
    b. Set **JNDI Name** = ArcotDB.
    c. Choose a suitable **Database Type**, for example, Oracle.
    d. Select a suitable **Database Driver**, for example, Oracle Thin Driver.
6. Click **Next**, retain the default values, and click **Next**.
7. In the Connection Properties page, set the database details. The following values are for Oracle Database:
    - **Database**: SID or service name of the DB server
    - **Hostname:** The DB server's IP address or host name
    - **Port**: 1521 or any other port the DB server is running
    - **Database User Name**
    - **Database Password / Confirm Password**
8. Click **Test Configuration** to verify that you have specified the correct database parameters.
9. Click **Next** and set the data source target to the preferred WebLogic server instance.
10. Click **Finish** to return to the data source list page.
11. Click **Activate** to enable the data source settings.

JBoss Application Server

This section provides the steps to enable JBoss Application Server for JNDI-based database operations.

1. Navigate to the location where you have deployed the WAR files. For example: <*JBOSS_HOME*>/server/default/deploy
2. Create a data source descriptor file called arcotdatabase-ds.xml.
3. Collect the following information for defining a data source in the arcotdatabase-ds.xml file:
   - JNDI Name

     The JNDI name that is used by the Arcot components. This name must match with the AppServerConnectionPoolName.*N* in arcotcommon.ini (without the java:comp/env/ prefix).
   - User ID

     The database user ID.
   - Password

     The database password.
   - JDBC Driver Class

     The JDBC driver class name. For example, oracle.jdbc.driver.OracleDriver.
   - JDBC URL

     The JDBC URL for the database server. For example, if you are using Oracle driver, then URL would be: jdbc:oracle:thin:<*server*>:<*database_port*>:<*sid*>.
   - Exception Sorter Class

     Class that implements the org.jboss.resource.adapter.jdbc.ExceptionSorter interface, which determines whether the exception indicates a connection error.

     Use this parameter for Oracle Database *only*. Set it to org.jboss.resource.adapter.jdbc.vendor.OracleExceptionSorter.
4. Open the arcotdatabase-ds.xml in a text editor.
5. Add the following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<datasources>
<local-tx-datasource>
<jndi-name>SampleDS</jndi-name>
<connection-url><jdbcurl></connection-url>
<driver-class><JDBC Driver class></driver-class>
<user-name><database_userid></user-name>
<password><database_password></password>
<exception-sorter-class-name><Exception Sorter
Class></exception-sorter-class-name>
```

```
</local-tx-datasource>
</datasources>
```

6. Save and close the file.

Enabling LDAP Connection Pooling

This section covers the configuration steps for enabling LDAP connection pool for the following application servers:

- Apache Tomcat (see page 247)
- IBM WebSphere Application Server (see page 248)
- Oracle WebLogic Server (see page 248)
- JBoss Application Server (see page 250)

Apache Tomcat

Perform the following steps to create an LDAP connection pool:

1. Install the Apache Tomcat application server and test the installation using the following URL:

   http://localhost:8080/

   The preceding URL must open the Apache Tomcat home page.

2. Navigate to the following location:
   <TOMCAT-HOME>`/conf`
3. Open the catalina.properties file in a text editor.
4. Add the following entries to the file:
   - com.sun.jndi.ldap.connect.pool.protocol=plain ssl
   - com.sun.jndi.ldap.connect.pool.authentication=simple
   - com.sun.jndi.ldap.connect.pool.maxsize=64
   - com.sun.jndi.ldap.connect.pool.prefsize=32
   - com.sun.jndi.ldap.connect.pool.timeout=240000
   - com.sun.jndi.ldap.connect.pool.initsize=8
5. Save and close the file.
6. Restart the application server.

IBM WebSphere Application Server

Perform the following steps to create an LDAP connection pool:

1. Log in to WebSphere Administration Console.
2. Navigate to **Servers** > **Server Types** > **WebSphere application servers**.
3. The Application servers page appears.
4. Click the Server that you want to configure.
5. In the **Server Infrastructure** section, click **Java and Process Management**.
6. Click the **Process Definition** link.
7. In the **Additional Properties** section, click **Java Virtual Machine**.
8. In the **Additional Properties** section, click **Custom Properties.**
9. Click **New** to add custom properties.

   The **General Properties** section appears.
10. Add the configurations that are listed in the following table as name-value pairs in the **General Properties** section. Repeat the process for every name-value pair:

Name	Value
com.sun.jndi.ldap.connect.pool.maxsize	64
com.sun.jndi.ldap.connect.pool.prefsize	32
com.sun.jndi.ldap.connect.pool.initsize	8
com.sun.jndi.ldap.connect.pool.timeout	240000
com.sun.jndi.ldap.connect.pool.protocol	plain ssl
com.sun.jndi.ldap.connect.pool.authentication	simple

11. Click **Apply**.
12. Restart WebSphere.

Oracle WebLogic Server

This section covers the following topics for enabling an LDAP connection pool for Oracle WebLogic Server:

- Including LDAP Options in Startup Script (see page 249)
- Specifying LDAP Pool Options Using Managed Server (see page 250)

Including LDAP Options in Startup Script

This section provides the steps to include the LDAP connection pool parameters in WebLogic server startup script:

1. Log in to the system
2. Create a backup copy of the WebLogic Server startup script. This script is available at the following location:
   `domain-name/bin/startWebLogic.sh`
3. Open the script in a text editor.
4. Add the following entries in the section that is used to start the WebLogic server.
   - -Dcom.sun.jndi.ldap.connect.pool.maxsize=64
   - -Dcom.sun.jndi.ldap.connect.pool.prefsize=32
   - -Dcom.sun.jndi.ldap.connect.pool.initsize=8
   - -Dcom.sun.jndi.ldap.connect.pool.timeout=240000
   - -Dcom.sun.jndi.ldap.connect.pool.protocol="plain ssl"
   - -Dcom.sun.jndi.ldap.connect.pool.authentication=simple

   The following code snippet shows a sample script with LDAP connection pool parameters:

```
# START WEBLOGIC
echo "starting weblogic with Java version:"
${JAVA_HOME}/bin/java ${JAVA_VM} -version
if [ "${WLS_REDIRECT_LOG}" = "" ] ; then
echo "Starting WLS with line:"
echo "${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS}
-Dweblogic.Name=${SERVER_NAME} -Djava.security.poli
cy=${WL_HOME}/server/lib/weblogic.policy ${PROXY_SETTINGS} ${SERVER_CLASS}"
${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS}
-Dcom.sun.jndi.ldap.connect.pool.maxsize=64
-Dcom.sun.jndi.ldap.connect.pool.prefsize=32
-Dcom.sun.jndi.ldap.connect.pool.initsize=8
-Dcom.sun.jndi.ldap.connect.pool.timeout=240000
-Dcom.sun.jndi.ldap.connect.pool.authentication=simple
-Dcom.sun.jndi.ldap.connect.pool.protocol="plain ssl"
-Dweblogic.Name=${SERVER_NAME}
-Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy
${PROXY_SETTINGS} ${SERVER_CLASS}
else
echo "Redirecting output from WLS window to ${WLS_REDIRECT_LOG}"
${JAVA_HOME}/bin/java ${JAVA_VM} ${MEM_ARGS} ${JAVA_OPTIONS}
-Dweblogic.Name=${SERVER_NAME}
-Djava.security.policy=${WL_HOME}/server/lib/weblogic.policy
${PROXY_SETTINGS} ${SERVER_CLASS} >"${WLS_REDIRECT_LOG}" 2>&1
fi
```

5. Save and close the file.
6. Restart Oracle WebLogic Server.

Specifying LDAP Pool Options Using Managed Server

1. Log in to WebLogic Administration Console.
2. Click the **Lock & Edit** button.
3. In the **Domain Structure** pane, navigate to **Environment** > **Servers**.
4. Click the server that you want to configure.
5. In the right pane, click **Server Start**.
6. In the **Arguments** field, include the following JVM options:
   - -Dcom.sun.jndi.ldap.connect.pool.maxsize=64
   - -Dcom.sun.jndi.ldap.connect.pool.prefsize=32
   - -Dcom.sun.jndi.ldap.connect.pool.initsize=8
   - -Dcom.sun.jndi.ldap.connect.pool.timeout=240000
   - -Dcom.sun.jndi.ldap.connect.pool.protocol="plain ssl"
   - -Dcom.sun.jndi.ldap.connect.pool.authentication=simple
7. Click **Save** and then **Activate Changes**.
8. Restart Oracle WebLogic Server.

JBoss Application Server

Perform the following steps to create an LDAP connection pool:

1. Navigate to the following location:
   `<JBOSS_HOME>/server/<Profile>/deploy/`
2. Open properties-service.xml file in a text editor.
3. Add the following properties to the <attribute name="Properties"> section:
   - com.sun.jndi.ldap.connect.pool.protocol=plain ssl
   - com.sun.jndi.ldap.connect.pool.authentication=simple
   - com.sun.jndi.ldap.connect.pool.maxsize=64
   - com.sun.jndi.ldap.connect.pool.prefsize=32
   - com.sun.jndi.ldap.connect.pool.timeout=240000
   - com.sun.jndi.ldap.connect.pool.initsize=8
4. Save and close the file.
5. Restart JBoss Application Server.

Enabling Apache Tomcat Security Manager

Perform the following steps to enable Tomcat Security Manager:

1. Add the security manager entries to the **JAVA_OPTS** environment variable, as follows:
   ```
   export CATALINA_OPTS="-Djava.security.manager
   -Djava.security.policy=<Tomcat_Home>/conf/catalina.policy"
   ```
2. Navigate to the following Apache Tomcat installation location:
   `<Tomcat_Home>/conf/`
3. Open catalina.policy file in a text editor.
4. Add the following code in the **WEB APPLICATION PERMISSIONS** section.
   ```
   grant {
   permission java.io.FilePermission
   "${catalina.base}${file.separator}webapps${file.separator}arcotuds${file.sepa
   rator}-", "read";
   permission java.util.PropertyPermission "adb.converterutil", "read";
   permission java.lang.RuntimePermission "accessDeclaredMembers";
   permission java.security.SecurityPermission "putProviderProperty.BC";
   permission java.security.SecurityPermission "insertProvider.BC";
   permission java.security.SecurityPermission "putProviderProperty.SHAProvider";
   permission java.io.FilePermission "${arcot.home}${file.separator}-",
   "read,write";
   permission java.net.SocketPermission "*:1024-65535", "connect,accept,resolve";
   permission java.net.SocketPermission "*:1-1023", "connect,resolve";
   };
   ```
5. Add the following section to grant permissions for Administration Console (arcotadmin) and User Data Service (arcotuds).
   ```
   grant codeBase "file:${catalina.home}/webapps/arcotuds/-" {
   permission java.lang.RuntimePermission "getenv.ARCOT_HOME", "";
   permission java.lang.RuntimePermission
   "accessClassInPackage.org.bouncycastle.asn1.*";
   permission java.security.AllPermission;
   };
   grant codeBase "file:${catalina.home}/webapps/arcotadmin/-" {
   permission java.lang.RuntimePermission "getenv.ARCOT_HOME", "";
   permission java.security.AllPermission;
   };
   ```
6. Save and close the file.
7. Restart Apache Tomcat.

Appendix I: Deploying Administration Console on IBM WebSphere 7.0

If you plan to deploy Administration Console on IBM WebSphere 7.0, then perform the following steps:

1. Change the working directory to:
   `<install_location>/arcot/sbin`
2. Type source arwfenv and press **Enter** to set the $ARCOT_HOME environment variable.
3. Restart the application server for the changes to take effect.
4. Navigate to the following location where the Administration Console WAR file is located:
   `<install_location>/arcot/java/webapps`
5. Copy the arcotadmin.war file to a temporary directory. For example, opt/arcot_temp.
6. Extract the arcotadmin.war file contents.

   Of the JARs that are extracted to the /opt/arcot_temp/WEB_INF/lib directory, the following JARs are used to create the shared library in IBM WebSphere:

   - axiom-api-1.2.10.jar
   - axiom-impl-1.2.10.jar
   - axis2-java2wsdl-1.5.2.jar
   - backport-util-concurrent-3.1.jar
   - commons-httpclient-3.1.jar
   - commons-pool-1.5.5.jar
   - axiom-dom-1.2.10.jar
   - axis2-adb-1.5.2.jar
   - axis2-kernel-1.5.2.jar
   - commons-codec-1.3.jar
   - commons-logging-1.1.1.jar
   - log4j-1.2.16.jar
   - axis2-transport-http-1.5.2.jar
   - axis2-transport-local-1.5.2.jar
7. Log in to IBM WebSphere Administration Console.

8. Click Environment, and then click Shared Libraries.
    a. From the Scope drop-down, select a valid visibility scope. The scope must include the target server/node on which the application is deployed.
    b. Click New.
    c. Enter the Name, for example, ArcotAdminSharedLibrary.
    d. Specify the Classpath. Enter the path and file name for all the JAR files that are extracted in Step 3.

       For example, /opt/arcot_temp/WEB_INF/lib/axiom-api-1.2.10.jar
    e. Click Apply to save the changes.
9. Navigate to the following location where the Administration Console WAR file is located:
   `<install_location>/arcot/java/webapps`
10. Deploy arcotadmin.war in the application server.
11. Configure shared library, as follows:
    a. Click Applications, and then click WebSphere enterprise applications.
    b. Click arcotadmin_war.
    c. In the References section, click Shared library references.
    d. Select arcotadmin_war and click Reference shared libraries.
    e. Select the ArcotAdminSharedLibrary from the Available list and move it to the Selected list.
    f. Click OK to save the configurations.
12. Configure the class loader order and policy as follows:
    a. Click Applications, Application Types, and then click WebSphere enterprise applications.
    b. Click arcotadmin_war.
    c. Click Class loading and update detection link.
    d. In the Class loader order section, select the Classes loaded with local class loader first (parent last) option.
    e. In the WAR class loader policy section, select the Single class loader for application option.
    f. Click OK to save the configurations.
13. Ensure that the application is restarted.

Appendix J: Troubleshoot Strong Authentication Errors

This appendix describes the troubleshooting steps, which will help you resolve the errors that you may face while using Strong Authentication. The troubleshooting topics are classified based on different Strong Authentication components as follows:

- Installation Errors (see page 256)
- Database-Related Errors (see page 261)
- Strong Authentication Server Errors (see page 263)

Before you perform any troubleshooting tasks, check the Strong Authentication log files to see if there were any errors. By default, all the log files are saved in the *<install_location>*/arcot/logs/ directory. The following table lists the default log file names of the Strong Authentication components.

Strong Authentication Component	File Name	Description
Strong Authentication Server	arcotwebfortstartup.log	This file records all the start-up (or boot) actions. The information in this file is very useful in identifying the source of the problems if the Strong Authentication service does not start up.
	arcotwebfort.log	This file records all requests processed by the server.
Administration Console	arcotadmin.log	This file records the Administration Console operations.
User Data Service	arcotuds.log	This file records the User Data Service operations.

Note: See the topic titled "Strong Authentication Logging" in the *Strong Authentication Installation and Deployment Guide* for detailed information about the Strong Authentication log files.

Installation Errors

Problem:

I do not find arcotadmin.war in <*install_location*>/arcot/java/webapps directory.

Cause:

The file may not have been created during installation.

Solution:

Perform the following steps to create the arcotadmin.war file:

1. Open a command window.
2. Ensure ARCOT_HOME environment variable is set.
3. Navigate to <*install_location*>/arcot/tools/common/bundlemanager directory.
4. Run bundlemanager with the following command:
   ```
   java -jar bundle-manager.jar
   ```
   The preceding command generates the arcotadmin.war file in <*install_location*>/arcot/java/webapps directory.

Problem:

I cannot start the Strong Authentication Server (Strong Authentication Service). I see the following error in arcotwebfortstartup.log:

```
Failed to initialize DB Pool Manager
```

or

```
Data source name not found and no default driver specified
```

Cause:

The possible causes for this issue may be:

- The DSN for your database was not created as System DSN.
- You are using a 64-bit platform. As a result, the DSN was created by using the 64-bit ODBC Manager.

Solution:

You can verify the DSN-related issues in the arcotcommon.ini file. If the problem is DSN-related, then:

1. To resolve the first cause, ensure that the DSN is a System DSN. To do so:
    a. Open the Control Panel, navigate to Administrative Tools, and Data Sources (ODBC).
    b. Activate the System DSN tab, and verify that the DSN exists. If it does not exist, re-create the DSN with the same name as earlier.
    c. Restart the service.
2. To resolve the second issue (if you are using a 64-bit platform), use the 32-bit version of the ODBC Manager.

Note: For detailed information about arcotcommon.ini and other configuration files, see the topic titled "Configuration Files and Options" in *Strong Authentication Installation and Configuration Guide*.

Problem:

I cannot start the Strong Authentication Server (Strong Authentication Service). The error message indicates that the service starts and stops automatically.

Cause:

A possible cause for this issue may be that you specified details for a database during installation, but the data source was not created successfully.

Solution:

To resolve this issue:

1. Verify if there is a corresponding entry for the DSN in arcotcommon.ini.
   - If entry not found, manually create the DSN.
2. If you found the entry, then clean up the database (see "Uninstalling Strong Authentication Schema" (see page 171)) and reseed the database, as described in section, "Running Database Scripts" (see page 75).
3. Restart the Strong Authentication Server.

Problem

When I launch the Administration Console for the first time (section, "Performing the Bootstrapping Tasks") as the Master Administrator, I see the following message:
`The server encountered an internal error that prevented it from fulfilling this request."`

I see the following error in the arcotadmin.log file:
`adminLog: java.lang.UnsatisfiedLinkError: no ArcotAccessKeyProvider in java.library.path`

Cause:

The JAVA library does not include the path to one of the following files:

- libArcotAccessKeyProvider.so
- arcot-crypto-util.jar

Solution:

Perform the following steps:

1. Ensure that the LD_LIBRARY_PATH variable includes the absolute path to the following files:
   - libArcotAccessKeyProvider.so
   - arcot-crypto-util.jar
2. Restart the application server.

Problem:

I do not see the log file (arcotadmin.log, arcotuds.log, or webfortserver.log) in the logs directory in ARCOT_HOME.

Cause:

Some of the probable causes for this issue may be:

- ARCOT_HOME may not be set correctly during installation.
- The application server JAVA HOME may be pointing to JRE instead of the JDK HOME.

Solution:

To resolve these issues, you must:

- Ensure that you reset the ARCOT_HOME to point to the correct location. Typically, this is *<installation_location>*/arcot/.

  As a result of this, when you use the cd $ARCOT_HOME command in the command prompt window, your current directory must change to *<installation_location>*/arcot/.
- Ensure that you copy the libArcotAccessKeyProvider.so and arcot-crypto-util.jar files in the application server JAVA HOME location.

Problem:

I deployed the UDS, but it is not coming up.

Cause:

One of the possible causes may be that the application server JAVA HOME may be pointing to JRE instead of the JDK HOME.

Solution:

Ensure that you have copied the libArcotAccessKeyProvider.so and arcot-crypto-util.jar files in the application server JAVA HOME location.

Problem:

Connection to the UDS fails. I see the following error message:

```
Unable to contact User Data Service
```

Cause:

The possible causes may be:

- You may not have specified the UDS **Host**, **Port**, and **Application Context** information for UDS correctly.
- The UDS service may not be initialized.

Solution:

To resolve this issue:

1. Verify whether the UDS information that you have provided in the "User Data Service Configuration" page of the Administration Console is correct. The details in **Host**, **Port**, and **Application Context** fields must be correct.
2. Check the UDS log file to ensure that the service was initialized successfully.

Database-Related Errors

Problem:

Connection to the database fails with the following entry in the Strong Authentication Server log file:

```
ReportError: SQL Error State:08001, Native Error Code: 30FD, ODBC Error: [DataDirect][ODBC Oracle driver][Oracle]ORA-12541: TNS:no listener
```

Solution:

Check for the following:

- Listener service on your database server.
- The TNSnames.ora file settings on the system where Strong Authentication Server is installed.

Problem:

Connection to the database fails with the following entry in the Strong Authentication Server log file:

```
TNS:listener could not resolve SERVICE_NAME given in connect descriptor
```

Solution:

Check for the following:

- Database is started. If it is not, this message appears.
- If the database is running, probably the database has not registered yet with the listener. This occurs when the database or listener just starts up. Normally this problem should be solved by waiting a minute or so.
- If you are using static registration, ensure that the SERVICE_NAME entry that is used in the connection string (TNSNAMES.ORA, NAMES, OID, ...) matches a valid service know by the listener.
- You can use C:>tnsping SERVICE_NAME - to check the status or C:>lsnrctl services - to verify all the services that are known to the listener.

Problem:

Connection to the database fails with the following entry in the Strong Authentication Server log file:

```
Database password could not be obtained from securestore.enc
```

Cause:

The database details may not be available in the securestore.enc file.

Solution:

Use the DBUtil tool to update the securestore.enc file with the database details. See *Strong Authentication Administration Guide* for more information about how to use DBUtil.

Problem:

Connection to the database fails with the following entry in the Strong Authentication Server log file:

```
ORA-03113: end-of-file on communication channel
```

Cause:

This is a generic error that indicates that the connection has been lost. This can be caused by reasons such as:

- Network issues or problems
- Forceful disconnection of a Server session
- Oracle Database crash
- Database Server crash
- Oracle internal errors, such as ORA-00600 or ORA-07445, causing aborts
- Oracle Client or TNS layer inability to handle the connections

Solution:

Check for the possible causes that are mentioned in the preceding list.

Strong Authentication Server Errors

Problem:

I am trying to start Strong Authentication Server, but it is not coming up. The last lines in arcotwebfortstartup.log show the following error:

```
WARN  STARTUP      -161388848 00WFMAIN - [11]: Protocol module
[SVRMGMT_WS] received portType error [bind: Address already in use]
```

Cause:

The possible cause for this issue is that the Server Management Port (Default Port Number: 9743) is already open on the host by some other process. While, Strong Authentication Server requires a minimum of Server Management port to start up.

Solution:

Perform the following steps:

1. Open the command prompt window.
2. Navigate to $ARCOT_HOME/bin.
3. Run the following command:

   arwfserver -i
4. Enter setsvrmgmtport *<new_port_number>*

   After the Server Management port is set, the Master Administrator can log in to the Administration Console and configure the other ports.

Problem:

RADIUS requests fail with the "Access-Reject" message.

Cause:

Check for the following:

- Shared secret is configured correctly.
- Strong Authentication Server logs for any errors.

Solution:

Log in to the Administration Console as Global Administrator or Organization and set the shared secret using the RADIUS Configuration page.

Problem:

CA AuthID authentication fails with the following entry in the Strong Authentication Server log file:

```
[Arcot Exception,No valid issuer certificate is available for this
certificate: unknown or invalid certificate issuer in ArcotVerifier],
Challenge verification failed -
```

Cause:

The Domain Key may have expired.

Solution:

Log in to the Administration Console as Global Administrator or Organization and configure the Domain Key using the Credential Key Management Configuration page.