CA ArcotID® PKI

Mobile Authentication Developer's Guide

r2.0.3

Contact CA Technologies

Contact CA Support

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

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

Providing Feedback About Product Documentation

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

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

Contents

Chapter 1: Introduction

Mobile phones now are not just used for wireless communication, they have also become a medium for home banking and performing financial transactions. As these transactions involve sensitive user data, relying just on user name for authentication is not sufficient.

To secure the mobile transactions from Man-in-the-Middle (MITM) or other related attacks, CA provides mobile applications, which must be downloaded to the user's mobile. CA mobile applications are based on ArcotID PKI and ArcotID OTP credentials, which are software credentials that provide two-factor authentication. These credentials use CA-patented **Cryptographic Camouflage** technique for securely storing 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 will decrypt it correctly, but many passwords will decrypt it to produce a key that looks valid enough to fool an attacker. As a result, this method protects a user's private key against fraudulent attacks, as a smart card does, but entirely in software.

To integrate ArcotID PKI mobile authentication with your application, you can either choose the ready-to-use *ArcotID PKI Application* available in the mobile vendor's store or build your own applications by using the Software Development Kit (SDK) that is shipped with the ArcotID PKI mobile authentication.

The *ArcotID PKI Mobile Authentication Developer's Guide* is designed to be a reference manual for you as you create mobile-based custom applications that use ArcotID PKI for authentication.

Notes:

- ArcotID PKI SDK is available in Java, JavaScript, and Objective C programming languages. The SDK format for all the supported programming languages is similar.
- This guide explains *only* Java SDK functions. If you are planning to use other SDKs, then you can use this guide as a reference.
- To get a better understanding of how to integrate the SDK with your client application, see the sample application shipped with the client package described in this guide.

Note: CA ArcotID PKI Mobile still contains the terms Arcot, WebFort and ArcotID in some of its code objects and other artifacts. Therefore, you will find occurrences of Arcot, WebFort and ArcotID in all CA ArcotID PKI Mobile 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.

ArcotID PKI Overview

ArcotID PKI is a secure software credential that provides two-factor authentication. An ArcotID PKI is a small data file that by itself can be used for strong authentication to a variety of clients such as Web or Virtual Private Networks (VPNs). ArcotID PKI is not vulnerable to "Brute Force" password attacks or "Man-in-the-Middle" attacks.

ArcotID PKI is a challenge-response type of authentication method, where AuthMinder Server sends the challenge to user, the client application that you build by using ArcotID PKI SDK signs the challenge by using user’s Personal Identification Number (PIN) and then sends it to the AuthMinder Server. The user is authenticated after verifying the signed challenge.

Chapter 2: Preparing for Integration

Before integrating with ArcotID PKI SDK, ensure that your integration environment meets the requirements described in this chapter. It contains the following sections

Integration Requirements

The following table lists the software required to develop ArcotID PKI client applications by using the ArcotID PKI SDK:

Mobile	Software
Apple iPhone	■ iOS 3.0 or later ■ iOS simulator 4.2, if you are using iPhone simulator for the developing the application
RIM BlackBerry	RIM Java Development Environment (JDE) 4.5 or later
Google Android	Android SDK 1.6 or later

The following table lists the mobile operating systems that are supported by ArcotID PKI SDK:

Operating System	Version
Apple iOS	3.0 or later
RIM BlackBerry	4.5 or later
Google Android	1.6 or later

Integration Notes

Before you start writing your code to integrate your application with ArcotID PKI SDK, ensure that:

- A release of CA AuthMinder that is supported by this release of the ArcotID PKI mobile application is installed and running.
- The ArcotID PKI profile is set by using the Administration Console.

  Note: The lifecycle management of ArcotID PKI credential is handled by AuthMinder. By default, these credentials have default settings, which will be used during issuance. If you want to change these settings, then configure the credential profiles by using Administration Console. Refer to *CA AuthMinder Administration Guide* for more information.
- You understand the AuthMinder Server SDK to be able to integrate it with the ArcotID PKI client SDK.

  Note: Refer to *CA AuthMinder Java Developer's Guide* and *CA AuthMinder Web Services Developer's Guide* for more information.

Chapter 3: Understanding ArcotID PKI APIs

This chapter discusses the ArcotID PKI Software Development Kit (SDK) that you can use for building mobile applications for authenticating users by using their ArcotID PKI. The most common tasks performed using this SDK are provisioning the ArcotID PKI account to the user's mobile and authenticating using that account. Other tasks that you can perform using the SDK include fetching and deleting the accounts from default location, storing accounts in a custom location, locking account to the device by using device parameters of your choice, and checking library version.

The chapter first introduces you to the interfaces and classes that you will be using for different tasks mentioned in the preceding paragraph, and later explains the usage in detail.

- Provisioning (Downloading) ArcotID PKI Accounts

  To perform ArcotID PKI authentication, you need to first create an account for the user that contains the ArcotID PKI information and save it on their mobile. The Provisioning ArcotID PKI Accounts (see page 12) section discusses the provisionAccount method in AID class that you need to use to create ArcotID PKI accounts.

- Authenticating Using ArcotID PKI

  The Authenticating Using Account (see page 13) section discusses the AID class that you need to use for signing the challenge.

- Managing Accounts

  The Managing Accounts (see page 14) section discusses the methods of AID class that you need to use for reading and deleting ArcotID PKI accounts stored in the default location.

  If you choose to store the accounts in a custom location, then you have to implement Store interface. Refer to the Choosing Custom Storage Medium (see page 18) section. To store the accounts in the mobile memory, use MemoryStore class. Refer to the Storing Accounts in Memory (see page 18) section.

- Device Locking

  Depending on the mobile that is being used, ArcotID PKI library supports default parameters for locking the account to the device. If you want to lock an account to the mobile by using the device parameters of your choice, then implement the DeviceLock interface, as discussed in the Device Locking (see page 17) section.

- Reading ArcotID PKI Details

  The Reading ArcotID PKI Account Details (see page 19) section discusses the Account class fields that hold the ArcotID PKI details such as, unique identifier for the account, ArcotID PKI of the user, and friendly name for the account. It also discusses the classes that are used to set and get additional ArcotID PKI attributes.

- Checking ArcotID PKI SDK Version

  The Checking Library Version (see page 21) section discusses the getVersion method in AID class that you need to use if you want to check the version of ArcotID PKI SDK.

Provisioning ArcotID PKI Accounts

To create an account for the users and save the account on their mobile, you must invoke the provisionAccount() method in the AID class. The location where the account is saved depends on the mobile where the account is being downloaded to. The following table lists the default storage location for different mobiles that ArcotID PKI SDK supports:

Mobiles	**Parameter Used**
Google Android-based mobiles	Database
Apple iOS-based mobiles	Database
RIM BlackBerry-based mobiles	Record Management Store (RMS)
JavaScript-based mobiles	Web browser local storage **Note:** If the Web browser does not support local storage, then Accounts are stored in a Cookie.

Notes:

- Other than the default location, accounts can *also* be stored in a custom location. Refer to Choosing Custom Storage Medium (see page 18) for more information on how to store accounts in a custom location.
- ArcotID PKI SDK also provides built-in functions to store accounts in the mobile memory, see Storing Accounts in Memory (see page 18) for more information.

API Details

The following table lists the input and output parameters of the provisionAccount()method:

Parameter	Description
Input Parameters	
b64aid	The base64-encoded format of the ArcotID PKI.
provUrl	The URL of the AuthMinder Server that issued ArcotID PKI for the user.
Output Parameters	
Account	Object containing the ArcotID PKI.

Exception

The AIDException class is returned if there any errors while executing the provisionAccount() method. See "ArcotID PKI SDK Exceptions and Error Codes (see page 23)" for more information on the exception class and errors returned by ArcotID PKI SDK.

Authenticating Using Account

ArcotID PKI is a challenge-response type of authentication. The challenge is posted by the AuthMinder Server to the user who is trying to authenticate. The client signs this challenge by using the user's ArcotID PKI PIN.

To sign the challenge, you need use signWithAccount() method in the AID class.

API Details

The following table lists the input and output parameters of the signWithAccount()method:

Parameter	Description
Input Parameters	
b64chall	The base64-encoded challenge sent by the AuthMinder Server.
id	The unique identifier of the account.
pwd	ArcotID PKI PIN. If you are using Objective C programming language, then the password is a mutable string that can be changed later. For Java programming language, the password is a character array. After the API is used, the password parameter is filled with random data or zeros.
Output Parameters	
Digital signature in base64-encoded format.	

Exception

The AIDException class is returned if there any errors while signing the challenge. See "ArcotID PKI SDK Exceptions and Error Codes (see page 23)" for more information on the exception class and errors returned by ArcotID PKI SDK.

Managing Accounts

This section discusses the APIs that you need to use for managing the accounts in default storage:

- Fetching Accounts (see page 15)
- Deleting Accounts (see page 16)

Fetching Accounts

To fetch the accounts from the default storage, you need to use the AID class. This class provides different options to read accounts as mentioned in the following table:

Method	Description
getAccount()	Fetches the accounts based on the account identifier that is passed as an input.
getAllAccounts()	Fetches all the accounts that are present on the mobile. **Note:** You can also fetch the accounts based on the ArcotID PKI namespace. For this, you have to pass namespace as an input parameter to the getAllAccounts() method. This method fetches all the accounts that belong to a particular namespace and also the accounts that belong to other namespaces, which contain the search string in their name. For example, if you want to fetch the accounts that belong to ARCOT.COM domain, then the API returns the accounts belonging to ARCOT.COM, A.ARCOT.COM, and B.ARCOT.COM.

API Details

The following table lists the input and output parameters of the getAccount() method:

Parameter	Description
Input Parameters	
id	The unique identifier of the account that has to be fetched.
Output Parameters	
account	The account of the user.

The following table lists the input and output parameters of the getAllAccounts() method:

Parameter	Description
Input Parameters	
None.	
Output Parameters	
account	An array of all the accounts present in the storage.

The following table lists the input and output parameters of the getAllAccounts() method, which takes namespace as the input:

Parameter	Description
Input Parameters	
ns	The namespace of the ArcotID PKI.
Output Parameters	
account	Array of accounts belonging to a specified namespace (domain) and also accounts from other namespaces that contain the search string in their name.

Exception

The AIDException class is returned if there any errors while reading the account from the storage location. See "ArcotID PKI SDK Exceptions and Error Codes" (see page 23) for more information on the exception class and errors returned by ArcotID PKI SDK.

Deleting Accounts

To delete the accounts from the default storage, you need to invoke the deleteAccount() method in the AID class.

API Details

The following table lists the input and output parameters of the deleteAccount() method:

Parameter	Description
Input Parameters	
id	The unique identifier of the account that has to be deleted.
Output Parameters	
Deletes the account from the default storage location.	

Exception

The AIDException class is returned if there any errors while deleting the account from the storage location. See "ArcotID PKI SDK Exceptions and Error Codes (see page 23)" for more information on the exception class and errors returned by ArcotID PKI SDK.

Device Locking

Device locking enables an account to be locked to a specific mobile, so that the account is unusable if it is copied to another mobile. The account is locked to the system when it is downloaded to the user's mobile. By default, the device locking feature is enabled.

Based on the mobile operating system, ArcotID PKI SDK supports different parameters to derive the unique identifier of the device for locking the account. If you want to use other mobile parameters for device locking, then see the Device Locking by Using Non-Default Parameters (see page 17) section for more information.

Note: You can disable this feature by passing a NULL value to the setDeviceLock() method.

Device Locking by Using Non-Default Parameters

To lock an account to a mobile by using parameters other than the default parameters supported by ArcotID PKI SDK:

1. Implement the getKey() method in the DeviceLock interface.

   The getKey() method returns the unique identifier of the mobile that you have requested for.

2. Invoke the setDeviceLock() method in the AID class.

   The setDeviceLock() method locks the account to the mobile by using the parameters that are fetched by the getKey() method.

API Details

The following table lists the input and output parameters of getKey() method:

Parameter	Description
Input Parameters	
The unique identifier of the mobile that has to be fetched. **Note:** For devices using Apple iOS operating system, the unique identifier used for device locking is a string that is generated for each application using CFUUIDCreateString. These generated strings are stored in the key chain.	
Output Parameters	
device identifier	The unique identifier of the mobile.

The following table lists the input and output parameters of setDevicelock() method:

Parameter	Description
Input Parameters	
lock	The unique identifier of the mobile.
Output Parameters	
Locks the account to the mobile.	

Exception

The AIDException class is returned if there any errors while locking the account to the device See "ArcotID PKI SDK Exceptions and Error Codes" (see page 23) for more information on the exception class and errors returned by ArcotID PKI SDK.

Choosing Custom Storage Medium

ArcotID PKI library enables you to store the accounts in a location of your choice, for this you have to implement the Store interface to define the storage medium, and then set that as default.

Perform the following steps to set up a custom storage:

1. Implement the Store interface to use the custom storage.
2. Invoke the setStore() method in the AID class to initialize the storage medium.

Storing Accounts in Memory

As an example for storing accounts in a location other than default storage, ArcotID PKI library provides methods to store the accounts in the mobile memory.

To store accounts in memory, invoke the setStore() method and pass the MemoryStore class as an object.

Reading ArcotID PKI Account Details

This section walks you through the following topics related to Account class:

ArcotID PKI Details

The following table lists the fields of the Account class that hold the ArcotID PKI information:

Field	Description
accountId	The unique identifier of the account.
logoUrl	The URL of the logo image, this image is displayed on the application. **Note:** This field is for future use.
name	A user friendly name for the account.
ns	The namespace of the ArcotID PKI. It is typically the domain name to which the ArcotID PKI belongs.
org	The organization to which the user for whom the account is being created belongs.
provUrl	The URL of the AuthMinder Server.

Fetching ArcotID PKI Details

The following table lists the methods that are used to fetch the ArcotID PKI information:

Method	Description
getBase64Aid	This method is used to fetch the ArcotID PKI in the base64-encoded format. Input Parameters: ■ None. Output Parameters: ■ ArcotID PKI in the base64-encoded format.

Method	Description
getId()	This method is used to fetch the unique identifier of the ArcotID PKI account. Input Parameters: ■ None. Output Parameters: ■ Identifier of the ArcotID PKI account.

Managing Additional ArcotID PKI Attributes

To set the ArcotID PKI information that cannot be passed by using the fields listed in the "ArcotID PKI Details (see page 19)" section, you need to use the setAttribute() method and pass that information as name-value pairs, and to read that information use getAttribute() method. The following table explains these methods:

Method	Description
setAttribute()	This method is used to set the ArcotID PKI information that cannot be passed by using the fields listed in the "ArcotID PKI Details (see page 19)" section. The additional information is passed as name-value pairs. Input Parameters: ■ The name of the attribute. For example, if you want to display your organization copyright information along with the user details on the mobile application, then you can set a new attribute called *Copyright*. Output Parameters: ■ The value (in string format) that has to be set for the attribute.
getAttribute()	This method is used to read the value of ArcotID PKI attributes. Input Parameters: ■ The name of the attribute, whose value has to be fetched. Output Parameters: ■ Attribute value in the string format.

Saving Additional ArcotID PKI Attributes

After you set a new ArcotID PKI attribute or change any existing ArcotID PKI attribute, you need to save these changes by invoking the saveAccount() method in the AID class.

Perform the following steps to save the changes made to accounts:

1. Prepare an instance of the Account object that has to be saved.
2. Invoke the saveAccount() method of the AID class to save the account.

API Details

The following table lists the input and output parameters of the saveAccount() method

Parameter	Description
Input Parameters	
acc	The account that has to be saved.
Output Parameters	
Saves the changes made to the ArcotID PKI account.	

Exception

The AIDException class is returned if there any errors while storing the account. See "ArcotID PKI SDK Exceptions and Error Codes (see page 23)" for more information on the exception class and errors returned by ArcotID PKI SDK.

Checking Library Version

To check the version of the ArcotID PKI library that you are using, you need to use the getVersion() method in the AID class.

API Details

The following table lists the input and output parameters of the getVersion() method:

Parameter	Description
Input Parameters	
None.	
Output Parameters	
Returns the version number of the ArcotID PKI library.	

Exception

The AIDException class is returned if there any errors while resetting the ArcotID PKI PIN. See "ArcotID PKI SDK Exceptions and Error Codes (see page 23)" for more information on the exception class and errors returned by ArcotID PKI SDK.

Chapter 4: ArcotID PKI SDK Exceptions and Error Codes

This chapter lists all exceptions and error codes that are returned by ArcotID PKI SDK. It covers the following topics:

Exceptions

If there are any errors while processing the ArcotID PKI APIs, then the AIDException class is returned. This class provide an constructor class AIDException, which takes error code, error message, and throwable as input.

To fetch the error code for a particular error, the AIDException class provides getcode() method, which returns the error code. See the "Error Codes (see page 23)" section for the list of error codes and error messages returned by ArcotID PKI APIs.

Error Codes

The following table lists the error codes returned by ArcotID PKI APIs:

Code	Code Message	Description
Default Errors		
1	E_UNKNOWN	Internal error.
Storage Errors (10-19)		
11	E_STORE_WRITE	There was an error while saving the account.
12	E_STORE_READ	There was an error while reading the account.
13	E_STORE_DELETE	There was an error while deleting the account.
14	E_STORE_ACCESS	There was an error while accessing the account.
User Input Errors		
31	E_BAD_NS	The namespace is invalid.
32	E_BAD_B64AID	The base64-encoded ArcotID PKI is invalid.
33	E_BAD_ID	The user identifier is invalid.

Code	Code Message	Description
34	E_BAD_ACCOUNT	The URL of AuthMinder Server is not configured correctly.
35	E_BAD_PIN	The ArcotID PKI PIN entered by the user is invalid.
36	E_BAD_B64CHALL	The base64-encoded challenge sent by the AuthMinder Server is invalid.
Processing Errors		
41	E_PROC_DEVLOCK	There was an error while locking the ArcotID PKI to the device.
42	E_PROC_CRYPTO	There was an error while performing the cryptographic operation.
43	E_ASN_READ	There was an Abstract Syntax Notation (ASN) error while reading the data.
44	E_ASN_WRITE	There was an ASN error while writing the data.