CA ArcotID® OTP

Authentication Developer's Guide

r2.0.2

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

Computers and mobile devices are used as a medium for home banking and performing financial transactions. Because these transactions involve sensitive user data, relying on user name for authentication is not sufficient.

To secure online transactions from Man-in-the-Middle (MITM) and other related attacks, CA AuthMinder provides client applications that are based on CA AuthID and ArcotID OTP credentials. These software credentials provide two-factor authentication and are based on the patented Cryptographic Camouflage technique for securely storing keys.

To address your business requirements, you can develop your own client application by using the Software Development Kit (SDK) that is described in this guide. Use this guide as a reference manual when you create a custom client application for use with the ArcotID OTP for authentication on mobile devices and computers.

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

ArcotID OTP is a One-Time Password compliant to OATH standards. Once the user's account is provisioned, the client application that you build by using ArcotID OTP SDK takes the user’s PIN as an input and generates passcodes on the user’s device. The user uses this generated passcode at the Web application that is protected by ArcotID OTP authentication. Based on the authentication result, the user is granted access to the protected application.

ArcotID OTP also supports the Transaction Signing feature in the Sign mode of passcode generation. This feature conforms to the OATH Challenge-Response Algorithm (OCRA) defined by RFC 6287.

Passcode generation is an offline process, which means the client application need not connect to the authentication server for generating passcodes.

ArcotID OTP library supports industry-standard passcode generation methods such as counter-based passwords (HOTP), time-based passwords (TOTP), MasterCard Chip Authentication Program (CAP), and VISA Dynamic Passcode Authentication (DPA).

Chapter 2: Preparing for Integration

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

- A release of CA AuthMinder that is supported by this release of ArcotID OTP Client is installed and running on the required operating system.
- The ArcotID OTP profile is set using the Administration Console.

  Note: The lifecycle management of ArcotID OTP credential is handled by AuthMinder. By default, these credentials have default settings, which are used during issuance. If you want to change these settings, configure the credential profiles by using Administration Console. Refer to *CA AuthMinder Administration Guide* for more information.
- The application you are integrating with supports the operating systems listed in Platform Support Matrix.

Chapter 3: Understanding ArcotID OTP APIs

This chapter discusses the ArcotID OTP Software Development Kit (SDK) that you can use to build client applications for authenticating users by using their ArcotID OTP. The most common tasks performed using this SDK are provisioning the ArcotID OTP account to the user's device and generating passcodes. Other tasks are outlined in the following list:

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

- **Provisioning (Downloading) ArcotID OTP Accounts**

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

- **Generating Passcodes**

  To perform ArcotID OTP authentication, the users have to first generate a passcode for authentication. The Generating Passcodes (see page 14) section discusses the generateOTP() method in the API class that you use to generate passcodes.

- **Resetting ArcotID OTP PIN**

  The Resetting ArcotID OTP PIN (see page 17) section discusses the resetPin() method in OTP class that you use to change the user's ArcotID OTP PIN.

- **Managing Accounts**

  The Managing Accounts (see page 17) section discusses the methods of the API class that you use for reading and deleting ArcotID OTP accounts stored in the default location.

  If you choose to store the accounts in a custom location, then you have to implement the Store interface. Refer to the Choosing Custom Storage Medium (see page 21) section. To store the accounts in memory, use MemoryStore class. Refer to the Storing Accounts in Memory (see page 21) section.

- **Device Locking**

  Depending on the device that is being used, ArcotID OTP library supports default parameters for locking the account to the device. If you want to lock an account to the device by using the parameters of your choice, then implement the DeviceLock interface, as discussed in the Device Locking (see page 20) section.

- **Reading ArcotID OTP Details**

  The Reading ArcotID OTP Account Details (see page 22) section discusses the API class fields that hold the ArcotID OTP details such as, unique identifier for the account, timestamp when the account was used, number of times the account was used, and friendly name for the account. It also discusses the classes that are used to set and get additional ArcotID OTP attributes.

- **Checking ArcotID OTP SDK Version**

  The Checking Library Version (see page 25) section discusses the getVersion() method in OTP class that you need to use if you want to check the version of ArcotID OTP SDK.

Provisioning ArcotID OTP Accounts

To create and store the ArcotID OTP account on the user's device, you must invoke the provisionAccount() method in the API class. The location where the account is saved depends on the device to which the account is being downloaded. The following table lists the default storage location for different mobiles that ArcotID OTP SDK supports.

Mobiles	**Parameter Used**
Google Android-Based Mobiles	Database
Apple iOS-Based Mobiles	Database
RIM BlackBerry-Based Mobiles	Record Management Store (RMS)
J2ME-Based Mobiles	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**.

On a computer, the default storage location is whichever of the following is available (in this order of priority):

- CA AuthMinder plugin store
- userData store
- HTML5 local storage
- Cookie store

Note: Other than the default location, accounts can *also* be stored in a custom location. Refer to Choosing Custom Storage Medium (see page 21) for more information on how to store accounts in a location of your choice. ArcotID OTP SDK also provides built-in functions to store accounts in memory, see Storing Accounts in Memory (see page 21) for more information.

API Details

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

Parameter	Description
Input Parameters	
xml	The user information that is required to create an account. This information is obtained from the AuthMinder Server.
provUrl	The URL of the AuthMinder Server that issued ArcotID OTP for the user.
code	The one-time activation code that authenticates the user before provisioning the account for them. The one-time activation code that authenticates the user before provisioning the account for them. This activation code is obtained from the AuthMinder Server.
pin	The PIN that is to be used to protect the account. The same PIN will have to be provided when calling the generateOTP() method.
Output Parameters	
Account	Object containing the ArcotID OTP.

Exception

The OTPException class is returned if there any errors while executing the provisionAccount() method. See chapter, "ArcotID OTP SDK Exceptions and Error Codes" (see page 27) for more information on the exception class and errors returned by ArcotID OTP SDK.

Generating Passcodes

To perform ArcotID OTP authentication, users have to generate a passcode on their device and then submit it at the authenticating website to access the protected source. To generate the passcode, use the generateOTP() method in the API class.

ArcotID OTP SDK supports major industry-standard One-Time Password (OTP) generation algorithms. Based on the algorithm that you are using, you must prepare the input data. The following table lists the fields that hold the input data required for generating passcodes.

Note: MasterCard Chip Authentication Program (CAP) and VISA Dynamic Passcode Authentication (DPA) algorithms support different modes for generating passcodes. Refer to the vendor documentation for more information on these modes.

Field	Description
CAP and DPA Password Fields	
P_MODE	Specifies that modes are being used for generating passcodes. The possible values for this parameter are: ■ M_1 ■ M_2 ■ M_2_TDS ■ M_3
M_1	Specifies that Mode 1 is being used for generating passcodes. If you are using this mode, then you have to collect the following information: ■ P_AA ■ P_TRCC ■ P_UN
M_2	Specifies that Mode 2 is being used for generating passcodes. This mode does *not* require any other additional information.
M_2_TDS	Specifies that Mode 2 with TDS is being used for generating passcodes. This mode supports10 entries for passing any additional information.
M_3	Specifies that Mode 3 is being used for generating passcodes. If you are using this mode, then you have to collect P_UN.
P_AA	Specifies the amount that is used in the transaction.
P_TRCC	Specifies the type of currency that is used in the transaction.
P_UN	Specifies the challenge data for the OCRA Transaction Signing feature.

Field	Description
P_DATA	This field is used to pass additional information that is required by M_2_TDS mode.
Time-Based OTP Password Fields	
A_TIMELEFT	This is an output parameter. It specifies the time in seconds for which the TOTP is valid.

API Details

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

Parameter	Description
Input Parameters	
id	The unique identifier of the account. **Note:** Use the getId() method to fetch this identifier.
pwd	ArcotID OTP PIN. This is the PIN that was associated with the account when the provisionAccount() method was called.

Parameter	Description

	The parameters required for generating passcodes. You need to set the parameters based on the type of OTP to be generated. For example:
	■ HOTP
	params.put(OTP.P_UN, "0123456789");
	■ TOTP
	Hashtable params = new Hashtable();
	params.put(OTP.P_TIME, "123456789");
	params.put(OTP.P_UN, "0123456789");
	Note: As mentioned in the previous table, the P_TIME value needs to be provided only if an OTP is being generated for a time other than the current time.
	■ CAP or DPA Mode 1
	Hashtable params = new Hashtable();
	params.put(OTP.P_MODE, OTP.M_1);
	params.put(OTP.P_AA, "123.45");
	params.put(OTP.P_UN, "0123456789");
	■ CAP or DPA Mode 2 with TDS
	Hashtable params = new Hashtable();
	params.put(OTP.P_MODE, OTP.M_2_TDS);
	params.put(OTP.P_DATA + "0", "123");
	params.put(OTP.P_DATA + "1", "456");
	params.put(OTP.P_DATA + "2", "789");
	■ CAP or DPA Mode 3
	Hashtable params = new Hashtable();
	params.put(OTP.P_MODE, OTP.M_3);
	params.put(OTP.P_UN, "0123456789");

Output Parameters	
params	params.get(A_TIMELEFT) Here, the value of the A_TIMELEFT parameter is the time interval in seconds for which the TOTP is valid.

Exception

The OTPException class is returned if there any errors while signing the challenge. See chapter, "ArcotID OTP SDK Exceptions and Error Codes" (see page 27) for more information on the exception class and errors returned by ArcotID OTP SDK.

Resetting ArcotID OTP PIN

The ArcotID OTP SDK provides functions that you can use to reset the user's ArcotID OTP PIN. Before resetting the PIN, you should prompt the users to perform secondary authentication to prove their identity. Typically, Security Questions and Answers or One-Time Passwords are used as secondary authentication mechanisms.

To reset the PIN, you need to use the resetPin() method in the API class.

Important! The current PIN cannot be verified. When a user is resetting a PIN, if an incorrect current PIN is passed to the resetPin() method, the account will become unusable. Therefore, do not use the resetPin() method in a context in which the user has forgotten the PIN.

API Details

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

Parameter	Description
Input Parameters	
id	The unique identifier of the account.
oldPin	User's current ArcotID OTP PIN.
newPin	New PIN that the user wants to set.
Output Parameters	
None.	

Exception

The OTPException class is returned if there any errors while resetting the ArcotID OTP PIN. See chapter, "ArcotID OTP SDK Exceptions and Error Codes" (see page 27) for more information on the exception class and errors returned by ArcotID OTP SDK.

Managing Accounts

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

- Fetching Accounts (see page 18)
- Deleting Accounts (see page 19)

Fetching Accounts

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

Method	**Description**
getAccount()	Fetches the account based on the account identifier that is passed as an input. Use the getId() method to fetch the account identifier. This method is described later in this document.
getAllAccounts()	Fetches all the accounts that are present on the device. **Note:** You can also fetch accounts based on the ArcotID OTP namespace. To do this, pass the namespace as an input parameter to the getAllAccounts() method. This method fetches all the accounts whose domains match the namespace passed to the method. For example, if you pass ARCOT.COM as a namespace to the method, then it returns accounts belonging to ARCOT.COM, A.ARCOT.COM, B.ARCOT.COM, and so on.

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 requested account.

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 store.

The following table lists the input and output parameters of the getAllAccounts() method when a namespace is passed as input:

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

Exception

The OTPException class is returned if there any errors while reading the account from the storage location. See chapter, "ArcotID OTP SDK Exceptions and Error Codes" (see page 27) for more information on the exception class and errors returned by ArcotID OTP SDK.

Deleting Accounts

To delete accounts, use the deleteAccount() method in the API 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	
None.	

Exception

The OTPException class is returned if there any errors while deleting the account from the storage location. See chapter, "ArcotID OTP SDK Exceptions and Error Codes" (see page 27) for more information on the exception class and errors returned by ArcotID OTP SDK.

Device Locking

Device locking enables an account to be locked to a specific device, so that the account is unusable if it is copied to another device. It is done at the time when an account is stored on the user's device. By default, the device locking feature is enabled.

Based on the operating system, ArcotID OTP SDK supports different parameters to derive the unique identifier of the device for locking the account. For example, for iOS mobiles, the CFUUID parameter is used to lock the device. If you want to use other parameters for device locking, then see Device Locking Using Non-Default Parameters (see page 20) for more information.

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

Device Locking Using Non-Default Parameters

To lock an account to a device by using parameters other than the default parameters supported by ArcotID OTP SDK, implement custom logic as explained in the following steps:

1. Implement the DeviceLock interface to use the custom device locking parameters.
2. Invoke the setDeviceLock() method in the API class.

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

API Details

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

Parameter	**Description**
Input Parameters	
None.	
Output Parameters	
device identifier	The unique identifier of the device.

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

Parameter	Description
Input Parameters	
lock	The unique identifier of the device.
Output Parameters	
None.	

Exception

The OTPException class is returned if there are any errors while locking the account to the device. See chapter, "ArcotID OTP SDK Exceptions and Error Codes" (see page 27) for more information on the exception class and errors returned by the ArcotID OTP SDK.

Choosing Custom Storage Medium

ArcotID OTP 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 API class to initialize the storage medium.

Storing Accounts in Memory

ArcotID OTP library provides a sample implementation for storing the accounts in device memory.

Note: The sample implementations that are provided with the ArcotID OTP library must be used for reference only.

Perform the following steps to store accounts in memory:

1. Invoke the MemoryStore class to use memory as a storage medium.
2. Invoke the setStore() method in the OTP class to initialize the storage medium.

Reading ArcotID OTP Account Details

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

- ArcotID OTP Details (see page 22)
- Fetching ArcotID OTP Details (see page 23)
- Managing Additional ArcotID OTP Attributes (see page 24)
- Saving Additional ArcotID OTP Attributes (see page 24)

ArcotID OTP Details

The following table lists the attributes that hold the basic ArcotID OTP account information. Use the getAttribute() method to use any of these attributes.

Field	**Description**
A_DLTA	Specifies the time difference between the client and the AuthMinder Server. This attribute is used in the case of time-based OTPs.
A_IAF_AA	Specifies whether the amount is required to perform transaction.
A_IAF_TRCC	Specifies whether currency is required to perform transaction.
A_IAF_UN	Specifies whether challenge is required to perform transaction.
A_MPL	Specifies the minimum length of the ArcotID OTP PIN.
A_PROTOCOLVER	Specifies the version of the protocol that is used between the client and the AuthMinder Server.
A_RESETSUPPORT	Specifies whether the AuthMinder Server supports re-synchronization if the client and server clocks are out of synchronization. This attribute is used in case of time-based OTPs.

The following table lists the fields that hold basic ArcotID OTP account information:

Field	**Description**
accountId	The account identifier of the ArcotID OTP account.
algo	The algorithm that is used generate passcodes.
creationTime	The timestamp when the ArcotID OTP account was created.
expiryTime	The timestamp when the ArcotID OTP account expires.

lastUsed	The timestamp when the ArcotID OTP account was last used.
logoUrl	The URL of the logo image, this image is displayed on the application.
name	A user friendly name for the account.
ns	The namespace of the ArcotID OTP. It is typically the domain name to which the ArcotID OTP belongs.
org	The organization to which the user for whom the account is being created belongs.
provUrl	The URL of the AuthMinder authentication server.
uses	The number of times ArcotID OTP has been used.

Fetching ArcotID OTP Details

Use the getId() method to fetch the identifier of the ArcotID OTP account. You can then use this identifier to fetch other information about the account:

Parameter	Description
Input Parameters	
None.	
Output Parameters	
Identifier of the ArcotID OTP account.	

Managing Additional ArcotID OTP Attributes

To set ArcotID OTP information that cannot be passed by using the fields listed in ArcotID OTP Details (see page 22), 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 OTP information that cannot be passed by using the fields listed in ArcotID OTP Details (see page 22). 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 client application, then you can set a new attribute called *Copyright*. ■ The value (in string format) that has to be set for the attribute.
getAttribute()	This method is used to read the value of ArcotID OTP attributes. **Input Parameters:** ■ The name of the attribute, whose value has to be fetched. **Output Parameters:** ■ Attribute value in string format.

Saving Additional ArcotID OTP Attributes

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

Parameter	Description
Output Parameters	
	None.

Exception

The OTPException class is returned if there any errors while checking the library version. See chapter, "ArcotID OTP SDK Exceptions and Error Codes" (see page 27) for more information on the exception class and errors returned by ArcotID OTP SDK.

Fetching Library Version

To fetch the version of the ArcotID OTP library that you are using, use the getVersion() method in the API class.

API Details

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

Parameter	Description
Input Parameters	
None.	
Output Parameters	
Fetches the version number of the ArcotID OTP library.	

Exception

The OTPException class is returned if there any errors while checking the library version. See chapter, "ArcotID OTP SDK Exceptions and Error Codes" (see page 27) for more information on the exception class and errors returned by ArcotID OTP SDK.

Chapter 4: ArcotID OTP SDK Exceptions and Error Codes

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

- Exceptions (see page 27)
- Error Codes (see page 27)

Exceptions

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

To fetch the error code for a particular error, the OTPException class provides getcode() method, which returns the error code. See the "Error Codes (see page 27)" section for the list of error codes and error messages returned by ArcotID OTP APIs.

Error Codes

The following table lists the error codes returned by ArcotID OTP 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_XML	The XML response from the server is invalid.
33	E_BAD_ID	The user identifier is invalid.

Code	Code Message	Description
34	E_BAD_ACCOUNT	The account is invalid.
35	E_BAD_PIN	The format of the PIN is invalid.
36	E_BAD_ALGO	The algorithm used for generating passcode is invalid.
37	E_BAD_CS	The ArcotID OTP card string is invalid.
38	E_BAD_ATTR	The attributes of the ArcotID OTP card passed are invalid or reserved attributes are being set.
Processing Errors		
41	E_PROC_SERVER	The AuthMinder Server returned an error.
42	E_PROC_XML	There was an error while processing the input XML.
43	E_PROC_DEVLOCK	There was an error while locking the ArcotID OTP to the device.
ArcotID OTP Card Errors		
51	E_TOTP_TIME	The timestamp passed for OTP generation is invalid.
52	E_CAP_MODE	The Client Authentication Program (CAP) mode used for generating passcode is invalid.
53	E_CAP_AA	The amount key is either not passed or the key length is incorrect.
54	E_CAP_TDS	The key to specify Mode2TDS mode is either not passed or the key length is incorrect.
55	E_CAP_TRCC	The currency challenge key is either not passed or the key length is incorrect.
56	E_CAP_UN	The challenge key is either not passed or the key length is incorrect.