ArcotID Client™ Reference Guide

Version 6.0.2

ARCOT®

455 West Maude Avenue, Sunnyvale, CA 94085

ArcotID Client Reference Guide
Version 6.0.2
April 2010
Part Number: AC-0602-00MG-10

Trademarks

Patents

Arcot Systems, Inc., 455 West Maude Avenue, Sunnyvale, CA 94085

Contents

Preface

Welcome to the ArcotID Client Reference Guide. This guide is designed to be a reference manual for you as you create custom applications that use ArcotID Client for authentication and digital signatures. The guide covers the following aspects:

- Introduction to various ArcotID Clients
- Feature description
- Installation
- API reference
- Integration with other applications

Intended Audience

This manual is intended for managers and administrators in organizations who are responsible for providing an online authentication and digital signature solutions. Many topics discussed in this manual are written for administrators who have the following skills: intermediate cryptography knowledge, experience with the applicable RDBMS, and familiarity with Web server administration.

Information Included in this Guide

This manual is organized as follows:

- Chapter 1, “Introduction”, provides the overview of different types of ArcotID Clients.
- Chapter 2, “What’s New”, describes the new features in ArcotID Client 6.0.
- Chapter 3, “Supported Client Environments”, lists all the supported environments.
- Chapter 4, “ArcotID Client Features”, lists all the features of ArcotID Client.
- Chapter 5, “Invoking the ArcotID Client”, describes methods for invoking the different ArcotID Clients.
- Chapter 6, “The ArcotID Client API Reference”, explains the Javascript APIs.

- Chapter 7, “Deploying the ArcotID Client”, describes the installation of ArcotID Clients on different platforms.
- Chapter 8, “ArcotID Client Tools”, describes the tools that are used with ArcotID Client.
- Chapter 9, “ArcotID Client Compatibility”, describes the compatibility issues.
- Chapter 10, “Integrating with Third-Party Applications”, describes the integration of ArcotID client with other software.
- Appendix A, “Source Code Samples”, provides the source code samples.
- Appendix B, “Glossary”, describes the most common terms used in the guide.

Conventions Used in This Book

The conventions, formats, and scope of this manual are described in the following paragraphs:

Typographical Conventions

This manual uses the following typographical conventions:

Italic	Emphasis, Guide names
Bold	User input, GUI screen text
`Fixed`	File and directory names, extensions, Command Prompt, CLI text, code
`Fixed Bold`	Target file or directory name in the path
`Fixed-Italic`	File or directory name that might be different from user to user
Link	Links within the guide, URL links

Formats

This manual uses the following formats to highlight special messages:

Note: Highlights information of importance or special interest.

💡	**Tip:** Highlights a procedure that will save time or resources.

⚠	**Warning:** Ignoring this type of note may result in a malfunction or damage to the equipment.

ℹ	**Important:** Information to know before performing an operation.

✋	**Caution:** Makes the user attentive of the possible danger.

📘	**Book:** Provides reference to other guides.

Contacting Support

If you need help, contact Arcot Support as follows:

Email	*support@arcot.com*
Web site	*http://www.arcot.com/support/index.html*

Chapter 1
Introduction

This chapter discusses the following topics:

- ArcotID and ArcotID Client
- Features and Usage Scenarios
- ArcotID Client Types
- Summary of Capabilities

ArcotID and ArcotID Client

ArcotID is a *secure software credential* that provides two-factor authentication and protects digital IDs using Arcot's patented *cryptographic camouflage* technology.

An ArcotID is a small data file that by itself can be used for strong authentication to a variety of applications such as Web or Virtual Private Networks (VPNs). ArcotID is not vulnerable to "brute force" password attacks or "man-in-the-middle" attacks.

The size of the ArcotID depends on the presence of the logo image. If the logo is present, then the ArcotID size can range between 3KB-100KB, otherwise it is typically between 2KB-3KB.

When an ArcotID is used to protect one or more Digital IDs, it provides the same features as a physical smartcard or USB security token such as, client-server strong authentication, digital signing, encryption, and decryption. The ArcotID Client software provides standard CSP and PKCS#11 protocol support for PKI-enabled applications.

Features and Usage Scenarios

This section describes the important features of ArcotID Client and the usage scenarios.

- **Web and VPN authentication**

Web applications and Web-based VPN requiring authentication can use the ArcotID Client to provide two-factor strong authentication. The ArcotID Client is available as a browser plugin, java applet, and flash client. This wide variety of deployment options enables the ArcotID Client to be used in most Web browsers and operating systems.

- **SSL client authentication using a digital ID in the ArcotID**

  The optional ArcotID *Key Vault* feature enables the ArcotID to protect additional *private key* portion of the digital ID. When the Key Vault feature is used, the ArcotID Client can be used for SSL client authentication with the help of Arcot Cryptographic Service Provider (CSP) or Arcot PKCS#11 module.

- **Digital signing by using a digital ID in the ArcotID**

  The ArcotID key vault feature also enables the digital IDs stored in an ArcotID to be used for digital signing in PKI enabled applications such as email, PDF files, and other document types. The ArcotID Client signing functionality is supported for most applications that support PKCS#11 and Microsoft cryptographic API standards.

- **Decrypting PKI encrypted files or emails by using a digital ID that is stored in the ArcotID**

  The ArcotID key vault feature enables the ArcotID Client to protect the decryption keys, which are used to decrypt the encrypted emails, PDF files, and other documents. The ArcotID Client decryption functionality is supported for most applications that support the PKCS#11 and Microsoft cryptographic API standards.

- **Roaming download of the ArcotID**

  The ArcotID Client has the ability to download an ArcotID. Depending on which client is used, the ArcotID can be temporarily downloaded into memory, saved on the user's hard disk, or saved on a USB storage device.

- **Signing a PDF by using a Roaming ID from Arcot SignFort in Adobe Reader**

  A subset of ArcotID Client is embedded in Adobe Acrobat 8 (and higher) as well as Adobe Reader 8 (and higher). This functionality enables the use of an ArcotID for two-factor authentication when digitally signing a PDF file by using a roaming digital ID that is stored in the Arcot SignFort server product.

ArcotID Client Types

To support a wide variety of application environments, the ArcotID Client is available in a variety of deployment types. These include:

- Flash Client
- Unsigned Java Applet
- Signed Java Applet
- Native Client for Windows and Mac OS X
- Embedded Client in Adobe Reader and Adobe Acrobat

Flash Client

The Flash client is an implementation of the ArcotID Client that can run in a Web browser that has the Adobe Flash Player (version 9 or higher). The Flash client can be used only for strong authentication to Web applications or SSL VPNs. An ArcotID downloaded by using the ArcotID Flash client can be stored persistently. However, they cannot be accessed by other ArcotID client types.

Note: The Flash Client supports the functionality of ArcotID Client 4.6.1. The new features provided in the 5.0 version are not yet supported by the Flash Client.

Unsigned Java Applet

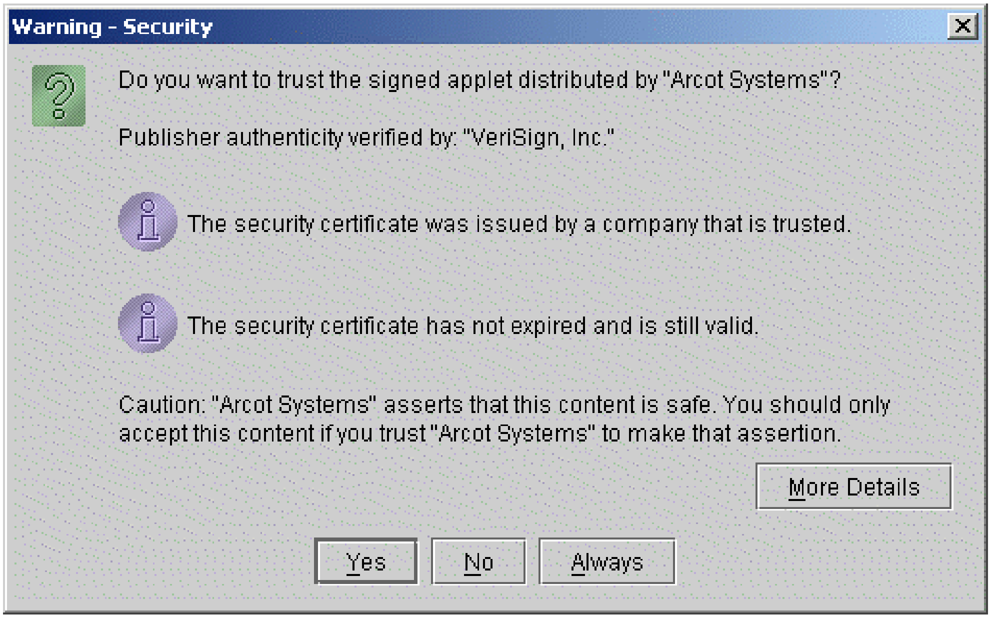
The unsigned Java applet is an implementation of the ArcotID Client that can run in any Web browser that contains a Java Virtual Machine (JVM) from Sun Microsystems or Microsoft. ArcotIDs downloaded by using the unsigned Java applet cannot be stored persistently. ArcotIDs can only be downloaded temporarily into memory for the duration of the browser session.

Signed Java Applet

The signed Java applet is an implementation of the ArcotID Client that can run in any Web browser that contains a Java Virtual Machine (JVM) from Sun Microsystems or Microsoft. When using the Arcot signed Java applet, the user will be presented a security message that requires the user to accept the signed applet before it is invoked. ArcotIDs downloaded by using the signed Java applet can be stored persistently. ArcotIDs downloaded by using the signed Java applet can also be accessed by the Native client for Windows.

A pop-up window is displayed when the signed Java applet is invoked for the first time. Click **Yes** on the window to continue using Java applet client.

Figure 1-1 Pop-up Window



Warning - Security

Do you want to trust the signed applet distributed by "Arcot Systems"?

Publisher authenticity verified by: "VeriSign, Inc."

The security certificate was issued by a company that is trusted.

The security certificate has not expired and is still valid.

Caution: "Arcot Systems" asserts that this content is safe. You should only accept this content if you trust "Arcot Systems" to make that assertion.

Native Client for Windows and Mac OS X

The native client for windows is an install package that includes the Arcot browser plug-in, Arcot Cryptographic Service Provider (CSP), and Arcot PKCS#11 module. The native client offers more functionality that the other clients. When installing the native client, the user is presented with a security message that requires the user to accept the installation of the client software.

Native client is available on Mac OS X platform in the form of Netscape Plug-ins for Mozilla Firefox and Apple Safari browsers.

Embedded Client in Adobe Reader and Adobe Acrobat

ArcotID Client functionality is embedded in the shipping versions of Acrobat 8 (and higher) and Adobe Reader 8 (and higher). This functionality enables ArcotIDs to be used for authentication to digitally sign PDF files by using a Roaming Digital ID. This functionality is dependent on another

Arcot product SignFort (available separately). ArcotIDs downloaded for use with SignFort are fully compatible with ArcotID Native Client and Signed Java applet, and can be used for other non-Adobe ArcotID client functionality.

Summary of Capabilities

The table below provides the quick overview of various client's functionality.

Table 1-1. ArcotID Client Features

		Flash Client	Unsigned Java Applet	Signed Java Applet	Native Windows	Native Mac OS X	Embedded in Adobe Reader
ArcotID Roaming Download		✓	✓	✓	✓	✓	✓
ArcotID Saved to Desktop		☑		✓	✓	✓	✓
ArcotID Saved to USB Drive					✓		✶
Authentication	Web	✓	✓	✓	✓	✓	
	VPN	✪	✪	✪	✓		
Digitally Sign Web Forms		✓	✓	✓	✓		
Digital Signing and Encryption CSP (MS office) and PKCS#11 applications (PDF)					✓		
Digital Signing with Roaming IDs							✓
Device Locked ArcotID				✓	✓		

☑Stored in flash secure object store, not available to copy to USB, floppy, CD, or other storage devices.

✶ Can use an ArcotID stored on a USB drive, but cannot save to USB.

✪ Only for SSL VPNs through a Web browser.

Chapter 2
What’s New

This chapter describes the new features in the ArcotID Client.

- New Features in 6.0.2
- New Features in 6.0.1
- New Features in 6.0

New Features in 6.0.2

This release is mainly focused on enabling the user to login using the same ArcotID with different aliases for the `USERNAME`. To support this feature, following new APIs are added:

- isArcotIDAvailableEx()
- SignChallengeEx2()
- RemoveArcotIDEx()

New Features in 6.0.1

This section provides a quick glance at the new features and enhancements provided in 6.0.1 for ArcotID Signed Java Applet client:

- **Device Locking**

  > **Note:** Previously, this feature was available only for Native client.

  This feature enables an ArcotID to be locked to one system, such that the ArcotID is not usable if it is copied to another system.

See section, "Machine PIN for Signed Java Applet Client" on page 4-19 for more information.

- **Support for Linux Platforms**

  The following flavors of Linux are now supported:

  - Ubuntu 9.10
  - SUSE 11.2
  - Fedora 12

  The default ArcotID storage location on these Linux distributions is same as Mac OS, which is:

  `$HOME/.arcot/ids`

- **Methods Currently *Not* Supported**

  The following methods are currently *not supported* for the **Signed Java Applet** client:

  - `SignChallenge()`
  - `RegisterCSPCertificates()`
  - `SignChallengeNonBlocking()`

  See Chapter 6, "The ArcotID Client API Reference" for more information.

New Features in 6.0

This section provides a quick glance at the new features and enhancements in 6.0:

- **Support for New Platforms**

  - This version extends the support of ArcotID Client plug-in on Mozilla Firefox and Apple Safari browsers on Mac OS X platform. Refer to "Deploying on Mac OS X" for more information on the installation procedure.
  - This version extends the support of ArcotID Client plug-in on Mozilla Firefox on Windows operating system.

- **API Changes**

  The following APIs are changed in this release to pass the organization name as an additional input parameter:

  - `SignChallengeEx()`
  - `RemoveArcotID()`

- **ArcotID File Name Changes**

  ArcotID files were previously saved as `username_domainname.aid`. The file format is now changed. The new file format is derived by hex-encoding the hash of `<username><organization><domain>` appended together. For example:

  ```
  97A6D55BE5375DEFDA0CC825302E2E868234B438.aid
  ```

- **ArcotID Default Storage Location**

  On Windows: The new default location for storing an ArcotID on Windows is `%USERPROFILE%\My Documents\ArcotIDs`.

  - For Windows XP users, the ArcotIDs will be stored at:

    ```
    C:\Document and Settings\<username>\My Documents\ArcotIDs
    ```

  - For Windows Vista users, the ArcotIDs will be stored at:

    ```
    C:\Document and Settings\<username>\Documents\ArcotIDs
    ```

  > **Note:** Any ArcotID stored in the legacy directory `%APPDATA%\arcot\ids` will continue to be recognized.

  On Mac OS X: The new default location for storing an ArcotID on Mac OS X is `$HOME/.arcot/ids` directory.

Chapter 3
Supported Client Environments

This chapter lists all the Operating System, Web Browsers, and JVMs supported by ArcotID clients.

Operating Environment

The section lists the supported operating environment for Native Plugin, Java Applet, and Flash client.

Table 3-1. Supported Client Environment

Operating Systems	Native Client	Java Applet	Flash Client
Windows			
Windows 7		✓	
Windows Vista Business	✓	✓	✓
Windows Vista Enterprise	✓	✓	✓
Windows Vista Home Premium	✓	✓	✓
Windows Vista Home Basic	✓	✓	✓
Windows Vista Ultimate	✓	✓	✓
Windows XP Home, SP1	✓	✓	✓
Windows XP Home, SP2	✓	✓	✓
Windows XP Professional, SP2	✓	✓	✓
Windows XP Media Center Edition 2005	✓	✓	✓
Windows 2000 Server, SP4	✓	✓	✓
Windows 2000 Professional, SP4	✓	✓	✓
Windows Server 2003, SP2	✓	✓	✓
Windows 98	✓	✓	✓

Table 3-1. Supported Client Environment

Operating Systems	Native Client	Java Applet	Flash Client
Linux			
Ubuntu 9.10		✓	
SUSE 11.2		✓	
Fedora 12		✓	
Mac OS			
Mac OS X Tiger 10.4✶	✓	✓	✓
Mac OS X Leopard 10.5✶	✓	✓	✓
Web Browsers			
IE 7.0	✓	✓	✓
IE 6.0	✓	✓	✓
IE 6.0 SP1	✓	✓	✓
IE 6.0 SP2	✓	✓	✓
Firefox 1.5		✓	✓
Firefox 2.0	✓✶✶	✓	✓
Firefox 3.0	✓✶✶	✓	✓
Apple Safari 1.2		✓	✓
Apple Safari 3.x	✓	✓	✓
Google Chrome 4.0		✓	✓
Java Virtual Machines			
Sun Java VM 1.4.2		✓	
Sun Java VM 1.5		✓	
Sun Java VM 1.6		✓	
Flash Player Versions			
Adobe Flash Player 9.0			✓

*ArcotID storage on USB device is not supported on Mac OS X platform.

**Supported on Windows and Mac OS X platforms.

Chapter 4
ArcotID Client Features

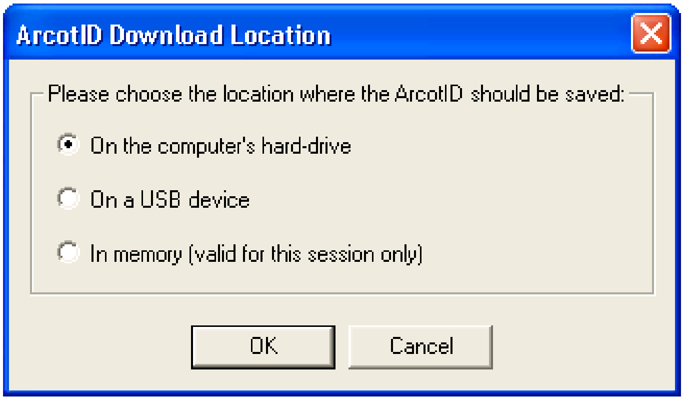
This chapter discusses the following ArcotID Client features in detail:

- ArcotID Storage
- ArcotID Authentication
- JavaScript APIs
- DNS Restrictions
- Device Locking
- Client Logging
- Customizing ArcotID Client Behavior
- Precedence Logic
- Partial Hash
- ArcotID Extension Directories
- Cryptographic Service Provider
- PKCS#11 Module

ArcotID Storage

The download of an ArcotID can be done either for the current session or permanently. This is identified by the storage medium that will be selected for storing the ArcotID.

Figure 4-1 ArcotID Download Location for Native Plug-In



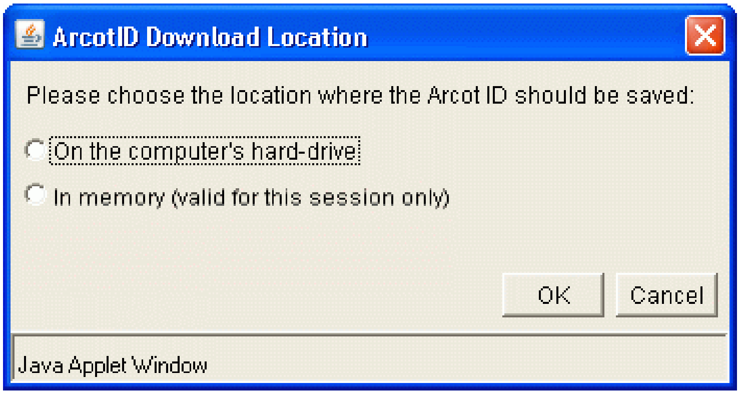
Please choose the location where the ArcotID should be saved:

● On the computer's hard-drive

○ On a USB device

○ In memory (valid for this session only)

Figure 4-2 ArcotID Download Location for a Java Applet



Please choose the location where the Arcot ID should be saved:

○ On the computer's hard-drive

○ In memory (valid for this session only)

- **Hard Disk**

  The ArcotID is directly downloaded to user’s hard disk. The ArcotID can be downloaded to either `%APPDATA%` directory or the user configured directory.

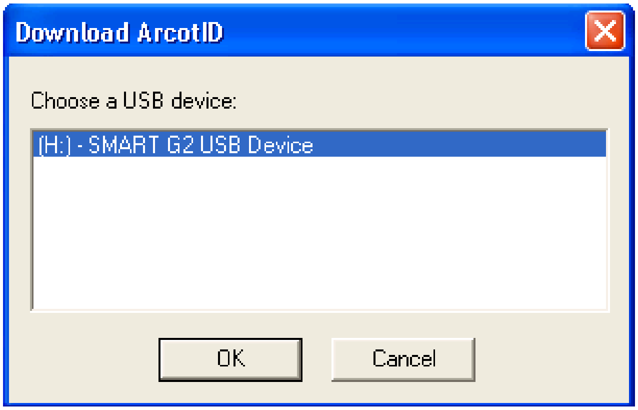
- **Memory**

  The ArcotID is downloaded to memory, and this is available only for the current session. The user has to re-download the ArcotID if required for the successive sessions.

- **(*Native Client Only*) Universal Serial Bus (USB)**

  The ArcotID is downloaded to the USB that is plugged to the system. If the ArcotID Client detects more than one device, then a pop-up window appears, which lists all the available external storage medium. The user has to select a device from the list.

Figure 4-3 Choosing the USB Device (*Native Client Only*)



Download ArcotID

Choose a USB device:

(H:) - SMART G2 USB Device

If the machine is not connected to any external device, then the user will be displayed with appropriate message.

ArcotID Authentication

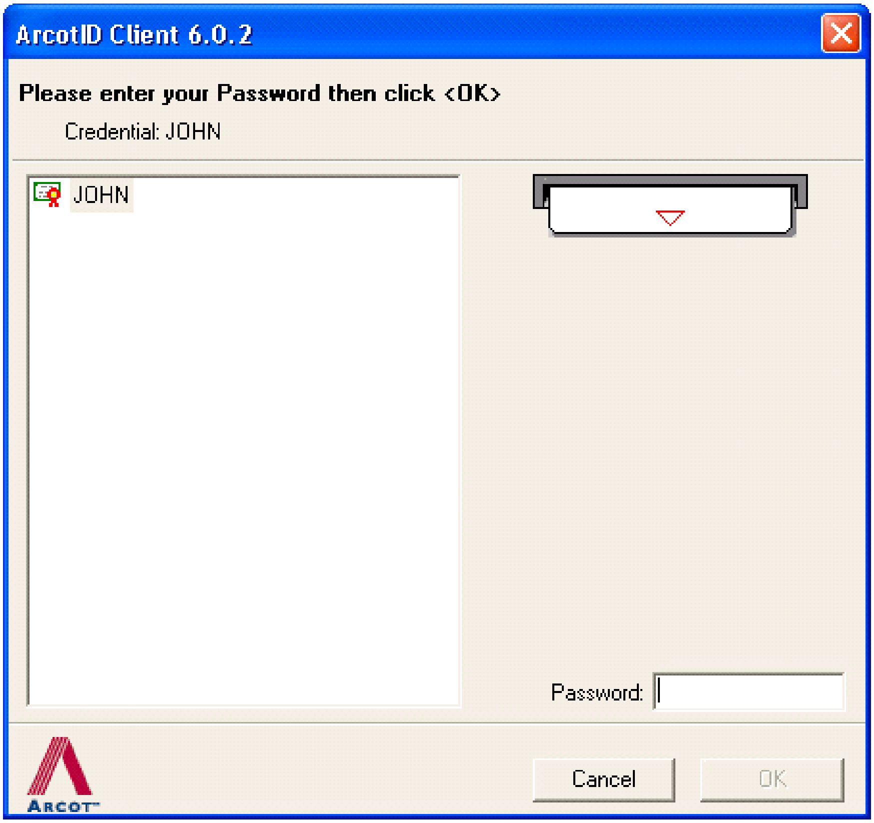
The ArcotID authentication process begins when a user connects to a Web page that requires authentication. The ArcotID authentication process is done in the following steps:

- Send Appropriate Challenge
- Generate Signature
- Verify Response
- Verify Authentication Token

The ArcotID Client collects the user password for ArcotID to access the application or the resource that is protected by WebFort. An ATM GUI is provided to collect the ArcotID password. The password can either be entered by using keyboard or the pin-pad.

See “Issuer Preferences” for more information on the PIN input method.

Figure 4-4 ATM GUI for Native Plug-In



ArcotID Client 6.0.2

Please enter your Password then click <OK>

Credential: JOHN

JavaScript APIs

ArcotID client provides a rich set of APIs for customizing the ArcotID display and behavior. See Chapter 6, “The ArcotID Client API Reference” for more information on Java APIs.

Note that the following methods are currently *not supported* for the **Signed Java Applet** client:

- `SignChallenge`
- `RegisterCSPCertificates`
- `SignChallengeNonBlocking`

DNS Restrictions

When an ArcotID is downloaded, the primary DNS domain where it was downloaded from is recorded in the ArcotID. The user will not be permitted to use that ArcotID in any other primary DNS domain.

For example, if an ArcotID is downloaded from `A.ARCOT.COM`, then it can be used for authentication at `B.ARCOT.COM` and `C.ARCOT.COM`, or any server in the `ARCOT.COM` domain. However, it cannot be used at Web servers in other primary domains, for example, `A.ARCOTSYSTEMS.COM`.

Device Locking

The Device Locking feature enables an ArcotID to be *locked* to a specific machine, so that the ArcotID is not usable if it is copied to another machine.

The feature works by camouflaging (protecting) an ArcotID using two passwords.

1. The password selected by the user when the ArcotID is issued.
2. A new Machine PIN, which is derived from unique machine-specific information derived from the hardware characteristics of the client machine.

When device locking is enabled, the ArcotID is cryptographically camouflaged twice, once with the user password and once with the Machine PIN.

The device locking is done at the time when an ArcotID is downloaded to the user's machine. After an ArcotID is locked to the user's machine, it is not usable if you copy it to another machine.

Typically when device locking is enabled, the server will not enable Roaming of the ArcotID, so that the user will not be able to download the ArcotID to another machine. To enable both roaming on the server and device locking, the ArcotID is device locked separately to each machine that it is downloaded to.

Machine PIN for Native Client

The Machine PIN is derived from the computer-specific information. Not all the information type is present on all the computers. For example, the motherboard serial number is not present on all motherboards. The ArcotID Client uses such machine characteristics available on the computer.

The following are the set of categories that are considered for generating the Machine PIN:

- `mem`

  Physical memory size.

- `vol`

  Boot partition volume ID.

- `bios`

  BIOS information such as, serial number and manufacturer name. This information is not always present.

- `mac`

  MAC addresses of all fixed Network Interface Cards (NIC). NICs that are not fixed, for example, PCMCIA, USB, docking station and NIC that is virtual, for example, VPN adapters are skipped. All other NICs are included. Adding or removing a NIC will cause Machine ID to change.

- `moth`

  Motherboard information such as, the serial number and manufacturer name. This information is not always present.

- `hd`

  Hard Disk (HD) information such as, model number and manufacturer's name. This is not HD serial number and therefore it would be same for all identical HDs. The removable HDs such as external USB, memory cards, are not included. Only the fixed HDs are included and therefore adding or removing a HD will cause Machine ID to change.

- `proc`

  Main Central Processing Unit (CPU) information such as, model and clock speed. For multiprocessor machine the information from all CPUs is included. This is not the CPU serial number.

- `encl`

Some computers, such as those manufactured by Dell, have their service tag encoded in the enclosure information.

Machine PIN for Signed Java Applet Client

The **Signed Java Applet** client currently uses the following category for generating the Machine PIN:

- `vol`

  The volume ID of the Boot partition. If this attribute is not found, then the ArcotID download will fail.

Client Logging

ArcotID Client supports logging for all the user actions. It also supports file logging for the server and other component logs. Logging can be done in multiple steps with a fine-grain configuration during startup and system management.

Logging may be enabled for the ArcotID Client by following these steps:

1. Copy the `ArcotLog2FileSC.dll` to the <*%SYSTEMDRIVE%*>:\system32\\ directory or some other directory which you will indicate in the configuration settings.

	Note: `ArcotLog2FileSC.dll` is not a part of the standard client package. It is only required for advanced troubleshooting purposes and is not required for most users. To obtain this DLL, please contact Arcot Technical Support.

2. Create the <*%SYSTEMDRIVE%*>:\Program Files\Common Files\Arcot Shared\logs\\ directory if it does not exist already.
3. Optionally set the `LoggingLevel` parameter to control the logging details.
4. Create a text file called `clientlog.ini` in the directory <*%SYSTEMDRIVE%*>:\Program Files\Common Files\Arcot Shared\conf\\.

   The text file must contain the configuration settings as shown in the `clientlog.ini File Example`.

	Note: The lines preceded by a # are comments that explain what each setting means.

clientlog.ini File Example

```
[arcot/WebClient]

# Count of loggers to send messages to. If set to 0 no logging

# is performed

LogSupported=1


# One sub-section must be created  below for each LogSupported, that is,
# LogLibrary<n> is a separate section for each n in LogSupported for 1 to N


[arcot/WebClient/LogLibrary1]

# DLLName is the name of the shared library which does the

# logging. The  name is provided without the suffix. You may provide a full
# path, if without path then the PATH variable will be searched to find the name
# provided. The suffix is automatically added depending on the operating system:
# Win32=".dll", AIX=".a", OtherUnix=".so"

DLLName=ArcotLog2FileSC


# HandleType indicates what type of messages should be routed to this logger.
# Possible values are ALL, APPLICATION, SYSTEM, AUDIT, STATUS, TRACE. Mulitple
# choices can be used seperated by commas.
# Including TRACE will capture debuging info when troubleshooting problems.
# Including AUDIT is will capture messages listed in the XMLMessagesToLog
# setting above.

HandleType=TRACE,APPLICATION,SYSTEM,STATUS


# HandleLevel indicates whether detailed log messages should be included. The
# number provided indicates that messages of the given severity and higher
# should be logged. Supported values are 1=debug, 2=info, 3=notice, 4=warning,
# 5=error, 6=alert, 7=fatal. We suggest a level of 3 or lower be used.

HandleLevel=1


# EntryPoint is the function within the library which can be called to get a
# handle to the logging object. This is fixed for a given log handler DLL.

EntryPoint=CreateFileLogHandler
```

```
# ParamSupported is the count of parameters to pass into the logging object.
# This will depend on the specific logging object and what it needs to know.

ParamSupported=4


# Param<M> gives a parameter in name=value format.

# Starting a path with ARCOT_HOME/ will substitute the installed directory from
# the environment; otherwise use a full pathname. For example, for Win32
# "LOG_FILE_NAME=D:\MyDocuments\ArcotClient.txt" or for Unix
# "LOG_FILE_NAME=/usr/spool/logs/ArcotClient.txt"

# LOG_FILE_NAME is for the ArcotLog2File logger. You can change the location and
# names of the log file by updating this value.

Param1=LOG_FILE_NAME=ARCOT_HOME/logs/ArcotClientLog.txt


# LOG_FILE_ROLLOVER_INTERVAL tells how often you want the log file to rollover
# to the backup file. The values recognized are HOURLY, DAILY, WEEKLY and
# MONTHLY. DAILY results in the file rolling over when the first log message is
# received with a time after midnight. The time check is based on the logged
# time; by default the local timezone is used for logging. The default is no
# setting which means the MAX_LOG_FILE_SIZE will be used to determine the
# rollover.

Param2=LOG_FILE_ROLLOVER_INTERVAL=DAILY


# MAX_LOG_FILE_SIZE is an alternative way to indicate rollover if the rollover
# interval is not set. It is expressed in bytes so 10000000 is about 10 MB. If
# this is set to 0, the log file will never be rolled over and will continue to
# grow indefinitely.

# Param3=MAX_LOG_FILE_SIZE=10000000

Param3=MAX_LOG_FILE_SIZE=


# BACKUP_LOG_FILE_LOCATION is the directory where the log files are rolled over
# to when they get to the max size. This directory must already exist and have
# permissions allowing the MS to create files in it.

Param4=BACKUP_LOG_FILE_LOCATION=ARCOT_HOME/logs
```

```
# LOG_LINE_FORMAT provides a string that indicates what attributes should be
# logged on each log line. If this parameter is not set then the legacy format
# is used. The typically used data tags are LTZ=local timezone date&time,
# SEV=severity, PID=processID, TID=threadID, MID=messageIDNumber, MSG=log
# message text, LID=loggingID. See the admin guide for a list of all the data
# elements that can be used in the format string.
Param5=LOG_LINE_FORMAT=$$TS1L$$ $$SEV$$ pid $$PID$$ tid $$TID$$: $$MID$$
$$MSG$$ ($$LID$$)
```

Customizing ArcotID Client Behavior

The ArcotID Client provides the following three tiers of preferences for customizing the behavior of the Arcot Client

- Issuer Preferences
- Application Preferences
- User Preferences

Issuer Preferences

Issuer preferences are name-value pairs stored in an ArcotID. These preferences are also referred to as ArcotID attributes. The supported attributes in the ArcotID Client include:

url_main

URL of the ArcotID issuer (for informational purposes only.)

Banner

Binary data of the logo image present on ArcotID.

url_help

URL where the user can look for help related to this ArcotID.

input_method

Determines the way how the ArcotID password can be received by the ATM-style GUI. Possible values include:

- `mouse`

Input can only be entered by using the mouse. Pinpad is visible but password edit box is disabled.

- `keyboard`

  Input can only be entered by using the keyboard. Pinpad is hidden and password edit box is enabled.

- `dual`

  Both mouse and keyboard can be used for entering the input. Pinpad is visible and password edit box is enabled.

- `not set`

  Required to be set. If not set, an undefined error appears.

userID

User ID of the ArcotID. The User ID and the card name of the ArcotID are stored in an extension in the Arcot certificate.

Org

Organization name of ArcotID.

Domain

Domain name of DNS domain where the ArcotID was downloaded. This attribute is set by the client software at the time when the ArcotID is downloaded, for example, `arcot.com`.

Key Store

The binary value of the *key bag*. Additional application keys protected by using the Key Authority features of WebFort.

KeyFortURL

The URL of the WebFort Key Authority (optional).

KeyFortUseWinInet

Determines whether or not native WinInet or WinHTTP, or both the libraries should be used for HTTP communication for Key Authority.

KeyFortSecret

It can be used to have an additional secret to protect the keybag, beyond the normal process.

scrstyle

Determines how often the pinpad should be scrambled in the ATM GUI. This is ignored if `input_method` is set to `keyboard`. Possible values include:

- `never`

  Pinpad is never scrambled.

- `once`

  Scramble the pinpad only once, when pinpad is initially displayed.

- `always`

  Scramble the pinpad every time a pinpad key is clicked or pressed.

- `not set`

  The default value *once* will be set.

scrorder

Determines the manner in which the pinpad is scrambled. This attribute is ignored if `input_method` is set to `keyboard` or if `scrstyle` is set to `never`. Possible values include:

- `random`

  Scrambling is done randomly. For example, "7253081496".

- `sequential`

  Scrambling is sequential. For example, "4567890123". The sequences "9012345678", "0123456789" are not allowed because the values are not scrambled completely.

- `not set`

  The default value `random` is set.

storage_type

Configures allowable storage medium for the ArcotID. Possible values include `hd_usb_memory`, or any combination of `hd`, `usb`, and `memory` delimited by the an *underscore* character. The default behavior is `any`. See "ArcotID Storage" for more information.

devlock_required

Configures whether the device locking feature is required for the ArcotID. Possible values are `yes` and `no`. The default value for the `devlock_required` attribute (if it is not present or holds an invalid setting) is `no`.

> **Note:** If the value is set to `no`, then the device locking can still be enabled by JavaScript using `SetAttribute (DeviceLocking)` because of the higher precedence.

devlock_type

A sequence of parameters delimited by underscores. For example, `moth_bios_hd`. The value `all` means all parameters are included for device locking. If no valid attribute is provided, then the default value `all` is used (assuming device locking is required or enabled by Application preferences).

> **Note:** A value of `no` does not rule out the use of device locking if the application JavaScript sets a preference using `SetAttribute(DeviceLocking)`.

Application Preferences

Application preference enables a Web application to customize the behavior of the ArcotID Client at run-time within the application. Application preferences are set using the `SetAttribute()` JavaScript API function. See “SetAttribute()” for more information about the API and the supported attributes.

User Preferences

The ArcotID native plugin client currently supports two user preferences. The user preferences are supported only in the native client, as shown in Figure 4-5.

Set default credential

In the **Arcot Digital Identity Plugin Options** dialog box, the user can select the default credential that is selected in the ATM GUI. This feature is for user convenience only.

Show all ArcotIDs

In the **Arcot Digital Identity Plugin Options** dialog box of the native plugin, the user can select the ***Show all ArcotIDs*** option to display all the ArcotIDs, even if they belong to different domain or filtering options.

Precedence Logic

The ArcotID Client behavior is determined by the settings that can be configured in four main ways:

- By the client itself
- Inside an ArcotID
- From a Web page through the JavaScript API
- Through choices made by the user at runtime

The precedence logic feature formalizes the precedence rules of how the run-time value of each setting is determined from the four sources of configuration information.

Note: The precedence rules themselves cannot be customized.

In general, the highest priority preferences are the issuer preferences. The issuer preferences specify the allowable behaviors for the ArcotID. If an issuer preference is not set, then the preference is for the application preferences and then the user preferences.

The precedence rules are described in the table below:

Table 4-1. Precedence logic

Features	Client Configuration	Issuer Preferences	Application Preferences	User Preferences
Storage type for ArcotID storage during download	Default values	Allowable values	Refinement of allowable or default values. Decider if no settings on ArcotID	Refinement of allowable values.
Location of ArcotID (during usage)	Default value		Overrides default value using credential filtering attributes	

Table 4-1. Precedence logic

Features	Client Configuration	Issuer Preferences	Application Preferences	User Preferences
Temporary or Permanent Download during Roaming	Permanent allowed only on some clients	Allowable values	Refinement of allowable or default values. Decider if no settings on ArcotID	Refinement of allowable values
Device Locking	Enabled on some clients only	Allowable mechanisms for device locking	Refinement of allowable or default values. Decider if no settings on ArcotID	N/A*
ArcotID Image displayed (applicable only when ATM GUI is used)	Default value	Overrides default value when present	N/A*	N/A*
PIN PAD scramble method	Default value	Overrides default value	Overrides ArcotID values, but only within the set of possible values allowed inside the ArcotID	

*The feature is not applicable.

Partial Hash

Within the ASN.1 structure of an ArcotID (not in the ArcotID attributes), an ArcotID may store a partial hash of the password. This is an optional feature and it is controlled by the server that creates the ArcotID.

The presence of the partial hash enables a portion of the invalid password to be tested on the client-side. This prevents user lockouts due to typos. However, systematic attempts like brute-force attack on the password will still be prevented.

The size of the partial hash is also configurable when the partial hash is stored in the ASN.1 structure. Configuring the size of the partial hash can help balance the trade-off between convenience (less user lockouts) and security (preventing attackers who are attempting to guess the password.)

Note: The partial hash functionality may be deprecated in the future release.

ArcotID Extension Directories

The ArcotID extension directories feature (previously knows as wallet extension directories) is still partially supported in the ArcotID Client 6.0. However, the feature is deprecated and may not be supported in future releases.

ArcotID extension directories enable ArcotIDs to be read from directories other than the default hard disk and USB drive locations.

To configure an ArcotID Extension Directory:

1. Create a text file with `.wlx` extension with the following parameters:
   - `path = <%SYSTEMDRIVE%>:\<extra_dirname>`

     Path specifies the location provided by the user where the ArcotIDs have to be stored.
   - `accessmode = ro`

     Permission on the file accessibility.
2. Store `.wlx` file in the default directory `%APP_DATA%:\arcot\`**`ids`**`\`.

In the above example, the WLX file points to the path `C:\extra_dirname`. This means that the ArcotID client will use this path when reading ArcotIDs. The access mode parameter must be `ro`. No other access mode values are supported at this time.

Note: If the same `wlx` file is stored in both legacy and default directory, the ArcotID will be detected twice by the application.

Cryptographic Service Provider

The ArcotClient.dll file provides the Arcot CSP functionality. This library file is signed by Microsoft as a CSP that is trusted by the Windows operating system. The Arcot CSP is registered by the Arcot Client installer if the user doing the installation has the appropriate permission. If the Arcot CSP is not registered during installation, then use the ArcotCSPInstall.exe tool to register the CSP at a later time.

PKCS#11 Module

The ArcotPK11_5_1_3.dll library file provides the Arcot PKCS#11 functionality. This file is recognized by most applications that support PKCS#11 modules. For example, Mozilla Firefox, Adobe Reader, and Adobe Acrobat.

The Arcot PKCS#11 module conforms to PKCS#11 version 2.01. The Arcot PKCS#11 module does not support all PKCS#11 API functions. Below is the list of PKCS#11 functions that are supported by the Arcot module:

- C_Initialize
- C_Finalize
- C_GetInfo
- C_GetFunctionList
- C_GetSlotList
- C_GetSlotInfo
- C_GetMechanismList
- C_GetTokenInfo
- C_GetMechanismInfo
- C_OpenSession
- C_CloseSession
- C_CloseAllSessions
- C_GetSessionInfo
- C_Login
- C_Logout
- C_CreateObject

- `C_GetAttributeValue`
- `C_SetAttributeValue`
- `C_FindObjectsInit`
- `C_FindObjects`
- `C_FindObjectsFinal`
- `C_DecryptInit`
- `C_Decrypt`
- `C_SignInit`
- `C_Sign`
- `C_Verify`
- `C_VerifyInit`
- `C_GenerateKey`
- `C_GenerateKeyPair`

The following PKCS#11 functions are not supported by the Arcot module:

- `C_InitToken`
- `C_InitPIN`
- `C_SetPIN`
- `C_GetOperationState`
- `C_SetOperationState`
- `C_CopyObject`
- `C_DestroyObject`
- `C_GetObjectSize`
- `C_EncryptInit`
- `C_Encrypt`
- `C_EncryptUpdate`
- `C_EncryptFinal`
- `C_DecryptUpdate`
- `C_DecryptFinal`
- `C_DigestInit`
- `C_Digest`
- `C_DigestUpdate`

- `C_DigestKey`
- `C_DigestFinal`
- `_SignUpdate`
- `C_SignFinal`
- `C_SignRecoverInit`
- `C_SignRecover`
- `C_VerifyFinal`
- `C_VerifyUpdate`
- `C_VerifyRecoverInit`
- `C_VerifyRecover`
- `C_DigestEncryptUpdate`
- `C_DecryptDigestUpdate`
- `C_SignEncryptUpdate`
- `C_DecryptVerifyUpdate`
- `C_WrapKey`
- `C_UnwrapKey`
- `C_DeriveKey`
- `C_SeedRandom`
- `C_GenerateRandom`
- `C_GetFunctionStatus`
- `C_CancelFunction`
- `C_WaitForSlotEvent`

Arcot has defined vendor-specific codes as permitted in the PKCS#11 standard. These codes are used in the Arcot PKCS#11 module.

Vendor-Specific PKCS#11 Mechanisms

CKM_ARCOT_SIGN_CHALL 0x87000000

This mechanism is:

- Returned by `C_GetMechanismList`
- Supported by `C_GetMechanismInfo`

- Used as argument to C_SignInit to request an Arcot proprietary-format signature by using the Arcot Private Key. The returned signature format is the DER encoding of SEQUENCE{ version, Certificate, X509Signature}.

Vendor-Specific PKCS#11 Attributes

The below attributes are supported by C_GetAttributeValue():

- CKA_ARCOT_CARD_IMAGE 0x87000000
- CKA_ARCOT_WALLET_NAME 0x87000001
- CKA_ARCOT_CARD_NAME 0x87000002
- CKA_ARCOT_WALLET_PATH 0x87000003
- CKA_ARCOT_CARD_DOMAIN 0x87000005

The below attribute is only supported when the object is an internal Arcot certificate:

- CKA_ARCOT_CARD_DOMAIN: card attribute domain

The below attributes are only supported when the object is a key vault certificate stored in an ArcotID:

- CKA_ARCOT_CARD_IMAGE: card attribute banner
- CKA_ARCOT_WALLET_NAME: ArcotID name
- CKA_ARCOT_CARD_NAME: ArcotId card name
- CKA_ARCOT_WALLET_PATH: Configuration value for the folder where ArcotIDs are stored by default.

Vendor-Specific Flags

The following are the vendor specific flags used by Arcot PKCS#11 module:

- CKF_ARCOT_NO_KEY_AUTHORITY_SESSION 0x01000000

  Supported as an argument to C_OpenSession. Indicates that no attempt to login to the key authority server should be made. This means the slot can be used for functions with the Arcot Certificate only, and the certificates in the card's KeyVault (if any) will not be visible.

- CKF_ARCOT_ONLY_KEY_AUTHORITY_SESSION 0x02000000

  Supported as an argument to C_OpenSession. Indicates that operations in the session will only refer to certs in the card's Key Vault. The Arcot Certificate will not be visible. This flag is ignored if CKF_ARCOT_NO_KEY_AUTHORITY_SESSION is present.

- CKF_ARCOT_TOKEN_KEYFORT_ENABLED 0x04000000

Returned by C_GetTokenInfo for a given slot when the slot represents and ArcotID that contains a KeyStore.

Chapter 5
Invoking the ArcotID Client

This chapter describes invocation process for different ArcotID Client from a Web page.

Invoking ArcotID Client on Web Page

This section describes the process of invoking ArcotID Client on a web page. Refer to the Appendix A, “Source Code Samples” appendix for the working sample code that is implemented by using these instructions.

To invoke the ArcotID Client from a Web page:

1. Include `arcotclient.js` in your page using the `SCRIPT` tag as follows:

   ```
   <SCRIPT type="text/javascript"
   src="arcotclient.js"></SCRIPT>
   ```

2. Create a `DIV` tag inside the body of your page.

   This identifies where dynamically generated code will be inserted. Depending on the type of client invoked (Native, Java, or Flash), different code may be generated in JavaScript. The code that is generated will be automatically inserted at the location of this `DIV` tag.

   The `DIV` tag needs to include a unique identifier such as `ArcotIDClient`. This unique identifier is used in the next step.

   The `DIV` tag is coded as follows:

   ```
   <div id="ArcotIDClient"></div>
   ```

3. Add JavaScript initialization method. Create a JavaScript method that initializes the ArcotID client when the page loads. For example:

   ```
   var arcotClient;
   function initClient() {
   ```

```
arcotClient = new ArcotClient();
arcotClient.setAttribute("clientBaseURL", "client");
arcotClient.write("ArcotIDClient");
   }
```

In the preceding example, `clientBaseURL` is an optional attribute. The full list of optional client invocation attributes are listed in the table below:

Table 5-1. Optional Invocation Attributes

Attribute	Possible values	Description	Default value
`clientType`	ActiveX Applet Flash	Sets which client should be used. Can be invoked several times to specify an order of preference for which client should be used. For example, if it is called first with Flash and then with Applet, then when `write()` is invoked, the library will first try to use the Flash version and if a Flash player is not available (or not the right version of Flash), it will instead use the Applet.	The default behavior is that the library detects the most appropriate client for the user's browser and OS combination.
`clientBaseURL`	Any URL (relative or absolute). If absolute, it must start with `http://` or `https://`	Sets the URL of the directory where the different ArcotID Client package files (`.cab`, `.jar`, `.swf`) can be found.	The default value is `""`, which means the client files are expected to be found at the same directory level as the invoking Web page.

Table 5-1. Optional Invocation Attributes (Continued)

Attribute	Possible values	Description	Default value
`clientReadyCallback`	The name of a JavaScript method on the current page.	This defines a JavaScript callback method that is invoked when the ArcotID Client is fully initialized and ready to process API calls such as `SignChallenge()` and `GetGlobalAttribute()`. This callback can be used to ensure login buttons are disabled on a Web page until after the client loads. This is especially useful for the Java applet because the JVM can take time to initialize during the first invocation of the applet. It is highly recommended to use this callback to avoid errors that result if ArcotID Client API functions are invoked before the client is initialized and ready.	None
`flashInstallURL`	URL (relative or absolute). If absolute, it must start with `http://` or `https://`	URL where the user is redirected if Flash is not installed in the user's browser.	*http://www.macromedia.com/go/getflashplayer*
`flashUpdateURL`	URL (relative or absolute). If absolute, it must start with `http://` or `https://`	URL the user is redirected to if Flash is available in the user's browser but is not the required version	*http://www.adobe.com/products/flash/about/*
`javaInstallURL`	URL (relative or absolute). If absolute, it must start with `http://` or `https://`	URL the user is redirected to if Java is not available in the user's browser	*http://www.java.com/en/download/*

Table 5-1. Optional Invocation Attributes (Continued)

Attribute	Possible values	Description	Default value
`signedApplet`	`true` or `false`	Whether the signed applet should be used instead of the unsigned applet.	`true`
`ActiveXMinVersion`	A version string in the following format: `5,0,0,0`	Sets the minimum version of the ArcotID ActiveX plugin that is required by the Web page. If the user has an older version of the ActiveX plugin, then the user will be prompted to upgrade to the newest version.	`5,0,0,0`

4. Add onload event.

   Add a call to the `initClient` method (created in Step 3) to the onload event of the body tag as follows:

   ```
    <body onload="initClient();">
   ```

   The onload event only happens after all the resources of the pages have been loaded. Pages that have large images might take more time to load, causing the client to be instantiated only after that extra delay. An alternative to speed-up the loading process is to instead use the ready event (which is triggered as soon as the page structure is available). Because of browser incompatibilities, Arcot suggests you to use a third-party package such as *jquery*. Using `jquery`, you can replace the onload event with the following JavaScript lines:

   ```
   $(document).ready(function(){
   arcotClientInit();
   }
   ```

Chapter 6
The ArcotID Client API Reference

This chapter contains reference material for the client-side development with Arcot WebFort. All of the client-side controls for WebFort are provided in the ArcotID Client Javascript API. This API allows you to program the functionality of the client using various Web programming languages including Java and JavaScript.

ArcotID Client Javascript APIs

The ArcotID Client Javascript API controls all the client-side functions related to managing and authenticating ArcotIDs. These APIs are common for both ArcotID Client browser plug-in and ArcotID Client Applet. These include signing challenges and managing error messages. Before the API can be accessed by the client, the Arcot WebFort browser plug-in must be installed on the client system.

Table 6-1. ArcotID Client Javascript APIs

Methods	Description
`AddCurrentCardToWallet()`	Adds the current ArcotCard to the ArcotID.
`AttachCertToCurrentCard()`	Attaches the new certificate to the ArcotCard.
`GetErrorCode()`	Returns an error code for the last unsuccessful call to the browser plug-in.
`GetErrorMessage()`	Returns a readable string that describes the last error encountered.
`GetVersion()`	Requests the version number of the currently loaded plug-in or applet.
`GetVersionEx()`	Gets the extended version information of the native client or the applet.
`GetGlobalAttribute()`	Requests a previously set plug-in attribute value.
`SetAttribute()`	Sets the value of predefined plug-in attributes and creates new plug-in attributes.
`SignChallenge()`	Signs a challenge previously obtained from the authentication server.

Table 6-1. ArcotID Client Javascript APIs

Methods	Description
`SignChallengeEx()`	Signs a challenge previously obtained from the authentication server.
`SetCurrentCardByIndex()`	Sets the current card to be the one with the given index in the current wallet.
`SetCurrentWalletFromEncoding()`	Sets the current wallet to be the one that is passed in as a string parameter.
`ImportArcotID()`	Downloads the ArcotID.
`RemoveArcotID()`	Removes the ArcotID.
`RegisterCSPCertificates()`	Registers the certificates with Microsoft CAPI.
`SignChallengeNonBlocking()`	Enables to support the ATM GUI applet on Mac OS X. The API call receives callback functions for success and error handling.
`RefreshArcotIDs()`	Instructs the ArcotID client to re-read the storage areas for ArcotIDs.
`isArcotIDAvailableEx()`	Checks if ArcotID of the user is present based on the lookup mode.
`SignChallengeEx2()`	Signs a challenge based on the lookup mode.
`RemoveArcotIDEx()`	Deletes the ArcotID of the user based on the lookup mode.

AddCurrentCardToWallet()

This method adds the current Arcot card to the named wallet stored in the client machine. If the mentioned file is not present, a new one is created.

Note: This is a deprecated function, use ImportArcotID() instead.

Syntax

```
boolean AddCurrentCardToWallet(WalletName)
```

Parameters

The following are the parameters of this method:

Parameter	Type	Description
`WalletName`	string	File name for the new wallet file (without the `.wlt` extension.)

Returns

If the method is successful, then it returns `TRUE`. If the method is unsuccessful, then it returns `FALSE`.

Example

```
var arcotClient = new ArcotClient();
var walletnameString = "GuestUser";

if (arcotClient.AddCurrentCardToWallet(walletnameString))
{
    document.write("<P>Card added to wallet successfully</P>");
}
else
{
    document.write("<P>Failed to add card to wallet</P>");
}
```

AttachCertToCurrentCard()

This method is used as part of the client-side key generation. `AttachCertToCurrentCard()` is used to attach the newly generated certificate to the card, where the keys were generated from GetErrorCode().

After this method, AddCurrentCardToWallet() needs to be invoked to add the new card to a wallet and store the wallet to the user's machine.

Syntax

```
string AttachCertToCurrentCard(certificateEncoding)
```

Parameters

The following are the parameters of this method:

Parameter	Type	Description
`certificateEncoding`	string	Base-64 encoded certificate you wish to attach to the current card.

Returns

If the method is successful, then it returns `TRUE`. If the method is unsuccessful, then it returns `FALSE`.

Example

```
var arcotClient = new ArcotClient();
var b64certString = "MIIFeQIBAAwHcGlucGFkMTCCBWcwggVjAgEBDAR
                     jYXJkoIHFMB8GCmCGSAGG+UYJAQEwEQQAAgEBAg
                     ILAgICAegCAgMYBIGhMIGeAgEAMA4GCmCGSAGG+
                     UYJAgAFAASBiDCBhQJBAM4mHRH5WMYDjDPUZD16...";

if (arcotClient.AttachCertToCurrentCard(b64certString))
{
    document.write("<P> AttachCertToCurrentCard succeeded</P>");
}
else
{
    document.write("<P> AttachCertToCurrentCard failed</P>");
}
```

GetErrorCode()

Retrieves the error code for the last error encountered by the SDK.

Syntax

`int GetErrorCode()`

Returns

An error code.

Discussion

Possible values include:

Error Code	Description
ERR_NONE (0)	
ERR_BAD_PIN (1)	The user entered an invalid PIN.
ERR_GUI_CANCEL (2)	The user cancelled authentication before it completed.
ERR_MISSING_WALLET (3)	The specified Arcot card was not found.
ERR_MISSING_CARD (4)	The specified card was not found.
ERR_GUI_RENEW (5)	The user choose to renew instead of authentication.
ERR_BAD_WALLET (6)	The Arcot card is invalid.
ERR_INVALID_PUBLIC_KEY (7)	The public key you are attempting to use is invalid.
ERR_MISSING_SERVER_NAME (8)	The authentication server could not be found.
ERR_INITCONN (9)	The connection between the client and the application server failed to initialize.
ERR_CONNECT (10)	The client failed to connect to the authentication server.
ERR_CREATE_SYSTEM_ CONTEXT (11)	Could not create an Arcot system context.
ERR-CREATE_CONTEXT (12)	Could not create an Arcot context.
ERR_PROTOCOL (13)	An Arcot authentication protocol error.
ERR_AAPLIB (14)	An Arcot authentication library error.
ERR_AUTHENTICATE (15)	The authentication failed.
ERR_COOKIE (16)	The cookie placed in the browser failed to authenticate.
ERR_ILLEGAL_CHALLENGE (17)	The challenge sent to the authentication server failed.
ERR_UNKNOWN (18)	An unknown error occurred during the authentication process.
ERR_BAD_WALLET_FOLDER (19)	Error in the folder containing the wallet.
ERR_ILLEGAL_DOMAIN_ACCESS (20)	An error occurred while accessing the domain.
ERR_INVALID_CREDENTIAL_FILTER (21)	The user credential was invalid after the filtering result.
ERR_BAD_APPCTX (22)	The application context is invalid.
ERR_BAD_ALIAS (23)	The alias is invalid.
ERR_BAD_WALLET_OR_ALIAS (24)	The alias or the wallet name is invalid.

Error Code	Description
ERR_INVALID_STORAGE (30)	Storage medium mentioned is invalid.
ERR_STORAGE_UNAVAILABLE (31)	Storage mentioned is not available.
ERR-STORAGE_ERROR (32)	Error occurred while storing the ArcotID.
ERR_NO_SUCH_WALLET_ERROR (33)	Mentioned wallet does not exist.

Example

```
<HTML>
<HEAD>
<TITLE>Authentication Failure</TITLE>
</HEAD>

<BODY>
<H2>Authentication Failure!</H2>

<SCRIPT LANGUAGE="JavaScript">

var arcotClient = new ArcotClient();
var errorcode = arcotClient.GetErrorCode();
if (errorcode == 1)
{
    document.write("<P>Invalid PIN</P>");
}
if (errorcode == 3)
{
    document.write("<P>Invalid User Name</P>");
}
```

GetErrorMessage()

Returns a text string that describes the last error encountered.

Syntax

```
string GetErrorMessage()
```

Returns

A readable string that describes the last error encountered.

Example

```
<HTML>
<HEAD>
<TITLE>Authentication Failure</TITLE>
</HEAD>

<BODY>
<H2>Authentication Failure!</H2>

<SCRIPT LANGUAGE="JavaScript">

var arcotClient = new ArcotClient();
var reason = arcotclient.GetErrorMessage();
document.write("<P>" + reason + "</P>");
```

GetVersion()

Gets the version number of the currently loaded Arcot plug-in or applet. This function can be used to check if the installed plug-in is latest or old.

Syntax

```
string GetVersion()
```

Returns

The version of the Arcot browser plug-in installed on the client system. For example, 4.6.0.0.

Example

```
<HTML>
<HEAD>

<SCRIPT LANGUAGE="JavaScript">

var arcotClient = new ArcotClient();
var version = arcotClient.GetVersion();
```

GetGlobalAttribute()

Retrieves attribute values from the plug-in, applet, or from a specific ArcotID. Not all ArcotID attributes can be retrieved by using this API.

Syntax

`string GetGlobalAttribute(`*`attributeName`*`)`

Parameters

The following are the parameters of this method:

Parameter	Type	Description
`attributeName`	string	The attribute you are requesting.

Returns

If successful, then it returns the Arcot card attribute you requested.

Discussion

Attributes allow you to set information that will persist in the plug-in till the browser session is closed. You can create your own attributes by using `SetAttribute()` or you can use special pre-defined attributes to obtain specific information about Arcot wallets and cards. Pre-defined attributes include:

Attribute	Description
SelectedWalletName	Set by the plug-in when the user attempts to renew their card.
SelectedCardName	Set by the plug-in when the user attempts to renew their card.
walletn:count	Indicates the number of ArcotIDs present on the client.
walletn:*x*	Selects a specific Arcot card within the wallet. The cards are numbered 0 - X.
walletn:*x:name*	Indicates the name of specific wallet.
walletn:*x:cardn:count*	Indicates the number of cards in the specific wallet. **Note:** Usually only one card is present in an wallet.
walletn:*x:cardn:x:name*	Indicates the name of the specific card in the particular wallet.
wallet:*walletname*:card:*cardname*::walletDER	Returns the DER encoding of the wallet with the specified wallet name and card name. **Note:** This requires a domain check to be done on the client to ensure the Web page in use is from the correct domain.
walletn:*x:cardn:x:serialnumber*	Fetches the serial number from the Arcot certificate present in the specified card of the wallet. The serial number is always in the hexadecimal format. **Note:** This function will succeed if the DNS domain of the current Web page matches the DNS domain of the given ArcotID.

Example

```
var arcotClient = new ArcotClient();
var i = 0;
var walletname;

// Get the number of wallets
var walletcount = arcotClient.GetGlobalAttribute("walletn.count");

// Print out the names of all the wallets
for (i = 0; i < walletcount; i++)
{
    walletname = arcotClient.GetGlobalAttribute("wallet." + i +  ".name");
    document.write("<P>" + walletname + "</P>");
}
```

SetAttribute()

Sets an attribute that will persist in the plug-in or applet till the browser session is closed. This function is used to set the Application Preferences.

Syntax

`boolean SetAttribute(`*`attributename, attributeValue`*`)`

Parameters

The following are the attributes that can be set by using this API function, all the attributes are of *string* type.

- WalletInMemory
- StorageType
- CredentialFilter
- AID_LOOKUP_MODE
- DeviceLocking
- ScrambleStyle

- ScrambleOrder

WalletInMemory

`WalletInMemory` is an attribute that instructs the client to download the ArcotID to memory only if set to `yes`, or to disk (permanent) if set to `no`.

Note: If the old APIs are used, then use this attribute is used to store the wallet in memory or in the hard disk.

Note: This is a deprecated attribute, use StorageType instead.

StorageType

This attribute specifies the storage location for the subsequent downloaded ArcotIDs. A user interface is also provided for choosing the storage location. If the user specifies any option other than `MEMORY`, then the `WalletInMemory` attribute will be deprecated.

The following are the possible values or the combination of any:

- `HD`

  Stores the ArcotID in the hard disk.

- `USB`

  Stores the ArcotID in the USB flash drive.

- `MEMORY`

  Stores the ArcotID in the memory for the current browser session.

CredentialFilter

This specifies how the clients should filter the credentials while querying an ArcotID, during authentication or during any other use of an ArcotID. Filtering criteria can include parameters such as, storage medium or issuing CA.

A credential filter is a set of expressions containing *`<attribute><operator><value>`*. Each expression is separated by an Ampersand (`&`).

For example, CertSubject=~OU%3DTesting&storagetype==hd is a credential filter, which displays only ArcotIDs that are stored in hard disk and containing the substring OU=Testing. Equal (=) signs that appear in the values need to be encoded as %3D.

There are four supported operators:

- == exact string match
- != not exact string match
- =~ substring match
- !~ not substring match

The case-sensitivity of the match is controlled by the case of the <*attribute*>. If it is all lower-case then the match is case-insensitive. Therefore in the example, the CertSubject match is case-sensitive while the storage type value is not.

The supported <*attribute*> values are:

- userid ArcotID attribute
- org ArcotID attribute
- alias ArcotID attribute
- input_method ArcotID attribute
- scrstyle ArcotID attribute
- scrorder ArcotID attribute
- ShowPinKeys Y/N - derived: input_method != keyboard
- ScramblePinPad Y/N (Y when scrstyle=once or always, N when scrstyle=never)
- ScrambleAlways Y/N (Y when scrstyle=always)
- ScrambleRandom Y/N (Y when scrorder=random)
- banner ArcotID attribute
- bannerURL ArcotID attribute
- domain ArcotID attribute
- url_main ArcotID attribute
- url_help ArcotID attribute
- name same as the card name
- CardName name of card

- `WalletName` name of wallet
- `CertSubject` Arcot certificate attribute
- `CertIssuerSubject` Arcot certificate attribute
- `CertSerialNumber` Arcot certificate attribute
- `CertNotBefore` Arcot certificate attribute
- `CertNotAfter` Arcot certificate attribute
- `KeyTypeCode` key type of Arcot private key, for example, `Arcot`/`RSA`.
- `cardImage` base64-encoded ArcotID attribute
- `label` ArcotID attribute
- `KeyFortEnabled Y/N` (`Y` when ArcotID attribute `KeyFortURL` is present)
- `StorageType` wallet attribute - device of locally stored ArcotID - value domain (`hd`, `mem`, `usb`, `cdrom`)
- `storage_type`
- `serialnumber` ArcotID attribute

The value portion of the expression should be URL-encoded if it contains any of the following special characters:

`'~', '!', '=', '&','<', '>'`

AID_LOOKUP_MODE

This attribute specifies how the clients should search the ArcotID of the users in the database. The ArcotIDs are searched using `username`, `orgname`, `alias`, or a combination of these parameters.

Note: ArcotID `username` aliases are stored as unsigned attributes inside the ArcotID. These unsigned attributes are of the format `Alias.<Application Context>=<alias>`. For example, `Alias.safebank-online=jdoe-sb`.

The following lookup modes are supported:

- `USERNAME_ONLY_MODE`

  Clients searches the ArcotID of the user by their `username` and `orgname`.

- `ALIAS_ONLY_MODE`

  Client searches the ArcotID of the user by their `alias` and `orgname`.

- USERNAME_AND_ALIAS_MODE

  In this mode, the client first searches the ArcotID of the user by their username and orgname. If a matching ArcotID is not found, then it searches the ArcotID based on the orgname and alias.

  Note: This is also the default mode.

DeviceLocking

This attribute configures the device locking mechanism in various ways. If no valid attributes are provided, then the default value all is used. Attributes are delimited by Underscore, when two or more are used.

The following are the possible values for this attribute:

- all

  Uses all the below mentioned device locking techniques.

- mem

  Physical memory size of the client's machine.

- vol

  Identifies the partition that houses the volume to be locked. The volume is identified by the volume identifier.

- mac

  The distinctive address that identifies a Network Interface Card (NIC)is called the Media Access Control (MAC) address.

  A MAC address is a unique character string that identifies a specific physical device, which means one individual NIC. Therefore, the MAC address does not change for the life of the NIC. Because your NIC's MAC address is permanent, it is often referred to as the real or physical address of a computer.

- moth

  Mother board serial number and manufacturer name. This information is not always present.

- bios

  BIOS serial number and manufacturer name. This information is not always present.

- hd

Hard Disk model number and manufacturer name. Only fixed hard disks are included but not the removable hard disks such as external USB or memory card. Changing or removing the hard disk will change the machine identifier.

- `proc`

  CPU information such as model number or clock speed. If the machine in use is a multiprocessor machine then the information from all the CPUs is included.

- `enc`

  Enclosure information is an unique information provided by the manufacturer, such as **Service Tag** provided by Dell for all its computers.

ScrambleStyle

This attribute facilitates the scrambling of pin pad, which is used to enter the ArcotID password. The following are the different values used to set the frequency of scrambling:

- `Never`

  The pin pad is never scrambled.

- `Once`

  The pin pad is scrambled only once, when it is initially displayed. This is the default option.

- `Always`

  Pin pad is scrambled every time a key is clicked or pressed.

ScrambleOrder

This attribute defines the order in which the pin pad is scrambled. It is ignored if the password is entered using keyboard or the "ScrambleStyle" is set to `Never`. Following are the different values for this attribute:

- `random`

  Scrambling is done in random manner. For example, *8403172695*

- `sequential`

  Scrambling is done in sequential fashion. For example, *4567890123*

Returns

If the method is successful, then it returns a `TRUE`.

Example

```
var arcotClient = new ArcotClient();

// Set the PinPad scrambling order to SEQUENTIAL scrambling (shifting)
arcotClient.SetAttribute("ScrambleOrder", "sequential");

// Set the PinPad scrambling style to ALWAYS scramble
arcotClient.SetAttribute("ScrambleStyle", "always");
```

SetCurrentCardByIndex()

The client sets the current card to be the one with the given index in the current wallet.

Note: This is a deprecated function use ImportArcotID() instead.

Syntax

`boolean SetCurrentCardByIndex(`*`index`*`)`

Parameters

The following are the parameters of this method:

Parameter	Type	Description
`index`	integer	The index number of the card you wish to set. Usually “0” since there is only one card in a wallet.

Returns

If the method is successful, it returns `TRUE`. If the method is unsuccessful, it returns `FALSE`.

Example

```
var arcotClient = new ArcotClient();
var walletnameString = "GuestUser";
var b64walletString  = "MII0yAIBAAwHY3Rnb29kMTCCNDcwgjQzAgEBDAR
                        jYXJkoIHFMB8GCmCGSAGG+UYJAQEwEQQAAgEBAg
                        ILAgICAegCAgMYBIGhMIGeAgEAMA4GCmCGSA GG+
                    UYJAgAFAASBiDCBhQJBAMzNUdGcaBZsiAt/88Dk9...";

// Wallet will be saved to either hard disk or USB
arcotClient.SetAttribute("WalletInMemory", false);

// Set the current wallet
arcotClient.SetCurrentWalletFromEncoding(b64walletString);

// Set the card at index 0 as the current (and only) card in the wallet
var returnValue = arcotClient.SetCurrentCardByIndex(0);
if (returnValue)
{
    if (arcotClient.AddCurrentCardToWallet(walletnameString))
    {
        document.write("<P>Card added to wallet successfully</P>");
    }
    else
    {
        document.write("<P>Failed to add card to wallet</P>");
    }
}
```

SetCurrentWalletFromEncoding()

Sets the current wallet to be the wallet passed in as a string parameter. This function is typically called before calling `AddCurrentCardToWallet()` during a roaming download.

Note: This is a deprecated function use ImportArcotID() instead.

Syntax

```
boolean SetCurrentWalletFromEncoding(walletEncoding)
```

Parameters

The following are the parameters of this method:

Parameter	Type	Description
`walletEncoding`	string	A base-64 encoded string of the wallet you wish to set.

Returns

If the method is successful, then it returns `TRUE`. If the method is unsuccessful, then it returns `FALSE`.

Example

```
var arcotClient = new ArcotClient();
var walletnameString = "GuestUser";
var b64walletString  = "MII0yAIBAAwHY3Rnb29kMTCCNDcwgjQzAgEBDAR
                        jYXJkoIHFMB8GCmCGSAGG+UYJAQEwEQQAAgEBAg
                        ILAgICAegCAgMYBIGhMIGeAgEAMA4GCmCGSA GG+
                   UYJAgAFAASBiDCBhQJBAMzNUdGcaBZsiAt/88Dk9... ";

// Wallet will be saved to either hard disk or USB
arcotClient.SetAttribute("WalletInMemory", false);

// Set the current wallet
arcotClient.SetCurrentWalletFromEncoding(b64walletString);

// Set the card at index 0 as the current (and only) card in the wallet
var returnValue = arcotClient.SetCurrentCardByIndex(0);
if (returnValue)
{
    if (arcotClient.AddCurrentCardToWallet(walletnameString))
    {
        document.write("<P>Card added to wallet successfully</P>");
    }
    else
    {
        document.write("<P>Failed to add card to wallet</P>");
    }
}
```

AddCurrentCardToWallet()

Adds the current card to the named wallet.

Syntax

`boolean AddCurrentCardToWallet(`*`data`*`)`

Parameters

The following are the parameters of this method:

Parameter	Type	Description
`data`	string	Name of wallet.

Returns

If the method is successful, it returns `TRUE`. If the method is unsuccessful, it returns `FALSE`.

SignChallenge()

Important: This method is currently *not* supported for Signed Java Applet client.

Triggers the ArcotID plug-in or applet to sign the challenge from the authentication server. User is provided with an interface to select the ArcotID and enter the password.

Syntax

`string SignChallenge(`*`challenge`*`)`

Parameters

The following are the parameters of this method:

Parameter	Type	Description
`challenge`	string	Base-64 encoded challenge string that is to be signed.

Returns

Base-64 encoded string that includes the digital signature and the Arcot certificate, which contains the encrypted public key.

Discussion

When a client sends a signed challenge to WebFort, it is authenticated by the authentication server. If the server authenticates the challenge a response is generated.

The `SignChallenge()` method checks for this response. If a response is generated, then the method passes this response back to the client.

You obtain the challenge that is passed to this function by using one of the server-side APIs that are available for this purpose.

Example

```
var arcotClient = new ArcotClient();
var challengeString = "gCcBwHe/XkIxMjM0";

var response = arcotClient.SignChallenge(challengeString);
```

SignChallengeEx()

Triggers the plug-in or applet to sign a challenge string. The name of the ArcotID and the password are passed in as parameters. The plug-in or applet does not present a GUI for this function.

Syntax

`string SignChallengeEx(`*`challenge`*`, `*`userID`*`, `*`cardname`*`, `*`PIN, orgName`*`)`

Parameters

The following are the parameters of this method:

Parameter	Type	Description
`challenge`	string	The encoded challenge string.
`userID`	string	The unique user identifier associated with the Arcot card.
`cardname`	string	The card name associated with the Arcot card.
`PIN`	string	The ArcotID password.
`orgName`	string	The name of the WebFort organization to which the user belongs.

Returns

Base-64 encoded string that includes the digital signature and the Arcot certificate, which contains the encrypted public key.

Discussion

Use this method if you are creating a custom UI.

You obtain the challenge that is passed to this function by using one of the server-side APIs that are available for this purpose.

Example

```
var arcotClient     = new ArcotClient();
var challengeString = "gCcBwHe/XkIxMjM0";
var useridString    = "GuestUser";
var cardnameString  = "card";
var pinString       = "123456";
var orgName         = "Acme Vendor";
// Wallet will be saved to either hard disk or USB
var response = arcotClient.SignChallengeEx(challengeString,
                                           useridString,
                                           cardnameString,
                                           pinString, orgName);
```

ImportArcotID()

This method is used to download the ArcotID from the server. The wallet name will be generated based on the internal wallet name of the ArcotID. If the internal wallet name is not present, then the wallet's User ID will be used as the wallet name. This single API call replaces the previous sequence of:

- Javascript call to `SetAttribute("WalletInMemory", true | false)`. This indicates if the download will be temporary (memory only) or permanent (ArcotID saved to disk).
- Javascript call to `SetCurrentWalletFromEncoding(walletEncoding)`. This function takes the wallet encoding as a base-64 encoded string and saves it to memory as the current wallet.
- Javascript function, `SetCurrentCardByIndex(0)`. This sets the "current card" to be the first card in the current wallet.
- Javascript function, `AddCurrentCardToWallet(walletName)`. This stores the previously specified "current card" to a new wallet stored on the hard disk or memory. The parameter `walletName` is used as part of the file name for a wallet that is stored to disk.

Syntax

`boolean ImportArcotID(`*`walletEncoding, StorageType`*`)`

Parameters

The following are the parameters of this method:

Parameter	Type	Description
`walletEncoding`	string	Base-64 encoded string of the wallet to import.
`StorageType`	string	Specifies the medium, where the ArcotID is stored. See "StorageType" for more information.

Returns

If the method is successful, then it returns `TRUE`. If the method is unsuccessful, then it returns `FALSE`.

Example

```
var arcotClient = new ArcotClient();
var b64wallet = "MII0yAIBAAwHY3Rnb29kMTCCNDcwgjQzAgEBDAR
                 jYXJkoIHFMB8GCmCGSAGG+UYJAQEwEQQAAgEBAg
                 ILAgICAegCAgMYBIGhMIGeAgEAMA4GCmCGSA GG+
                 UYJAgAFAASBiDCBhQJBAMzNUdGcaBZsiAt/88Dk9... ";

// Import wallet and allow it to be saved on hard disk or memory only
var returnValue = arcotClient.ImportArcotID(b64wallet, "hd_memory");
```

RemoveArcotID()

This method removes the ArcotID with the specified wallet name and storage types. The function does a domain check of the calling Web page to ensure it matches the domain attribute stored in the ArcotID. For the ArcotID browser plug-in, this function will also un-register any key bag certificates that were registered in CAPI.

Syntax

`boolean RemoveArcotID(`*`walletname, StorageType, orgName`*`)`

Parameters

The following are the parameters of this method:

Parameter	Type	Description
`walletname`	string	The name of the wallet to be removed.
`StorageType`	string	Specifies the medium, where the ArcotID is stored. See "StorageType" for more information.
`orgName`	string	The name of the WebFort organization to which the user belongs.

Returns

None.

Example

```
var arcotClient = new ArcotClient();
var walletname  = "GuestUser";
var orgName     = "Acme Vendor";

// Remove the ArcotID
var returnValue = arcotClient.RemoveArcotID(walletname, "hd", orgName);

// Reload all the ArcotIDs
arcotClient.RefreshArcotIDs();

// Get the number of wallets
var walletcount = arcotClient.GetGlobalAttribute("walletn.count");
```

RegisterCSPCertificates()

Important: This method is currently *not* supported for Signed Java Applet client.

For each ArcotID, any key vault certificates will be registered into Microsoft CAPI (Crypto API) so that they can be used by Internet Explorer, Microsoft Outlook, and any other application that supports CAPI.

If `ImportArcotID()` was used to download the ArcotID by using the Native Client, it is not necessary to invoke *`RegisterCSPCertificates()`*, because `ImportArcotID()` registers any key vault certificates if it has the permission to do so.

On the other hand, if an ArcotID is imported to a machine using a USB token, `RegisterCSPCertificates()` needs to be invoked to ensure the key vault certificates are made available to CAPI.

Syntax

```
string RegisterCSPCertificates(walletname)
```

Parameters

The following are the parameters of this method:

Parameter	Type	Description
`walletname`	string	Wallet name of an ArcotID whose certificates have to be registered.

Returns

None.

Example

```
var arcotClient = new ArcotClient();
var walletname = "GuestUser";

// Register the certificates in this ArcotID into MS CAPI
arcotClient.RegisterCSPCertificates(walletname);
```

SignChallengeNonBlocking()

Important: This method is currently *not* supported for Signed Java Applet client.

This function resolves GUI issues related to the Arcot applet, and enables to support the ATM GUI applet on Mac OS X. The API call receives callback functions for success and error handling.

Note: This function is not implemented in the native plugin. If called on the native plugin, the native plugin will return the error code `ERR_FUNCTION_NOT_IMPLEMENTED`.

Syntax

`SignChallengeNonBlocking(`*`challenge`*`, `*`browser`*`, `*`onComplete`*`, `*`onFailure`*`)`

Parameters

The following are parameters of the method:

Parameter	Description
`challenge`	The challenge that has to be signed.
`browser`	This option decides on where the callback functions should be invoked. This is an optional parameter. If set, the functions are invoked on a different window. If the value is null, the current browser is used.
`onComplete`	This is a JavaScript function that is invoked after the challenge has been successfully signed. The parameters passed are challenge and the signed challenge.
`onFailure`	A Javascript function that is invoked when the signing has failed. The parameters passed are challenge and the error message.

Example

```
arcotClient.SignChallengeNonBlocking(challenge, document,
challengeSigned,

// Here we inline the error method, we could have called an external
function instead

function (challenge, error) {
alert("Failed to sign the challenge, error: " + error);});

function challengeSigned(challenge, signedChallenge) {
// Set the signed challenge in the form
document.getElementByID("SignedChallenge").value = signedChallenge;
// Submit the form
document.getElementByID("LoginForm").submit();
}
```

GetVersionEx()

Gets the extended version information of the currently loaded Arcot plug-in or applet. In addition to the client version number, this function will also return the build number that is based on the current date. For example: "6.0 (200905211523)".

Syntax

```
string GetVersionEx()
```

Returns

The version of the ArcotID Client installed on the client system.

Example

```
<HTML>
<HEAD>

<SCRIPT LANGUAGE="JavaScript">

var arcotClient = new ArcotClient();
var versionEx = arcotClient.GetVersionEx();
```

RefreshArcotIDs()

This API call will instruct the ArcotID Client to re-read the storage areas for ArcotIDs. This is very useful for ArcotIDs stored on removable USB flash drives.

Syntax

`RefreshArcotIDs()`

Returns

`None`

Example

```
var arcotClient = new ArcotClient();

// Reload all the ArcotIDs
arcotClient.RefreshArcotIDs();

// Get the number of wallets
var walletcount = arcotClient.GetGlobalAttribute("walletn.count");
```

isArcotIDAvailableEx()

This API checks whether the ArcotID for the specified user is present. The ArcotID is searched based on the AID_LOOKUP_MODE attribute. The following table explains how the input parameters (`userNameOrAlias`, `appctx`, `orgname`) are interpreted and used by the 3 lookup modes.

Table 6-2. Input Parameter Interpretation

Lookup Mode	Input Parameter Interpretation
`USERNAME_ONLY_MODE`	The `userNameOrAlias` is treated as `username` and the `appctx` is ignored. The ArcotID is searched based on `username` and `orgname`.
`ALIAS_ONLY_MODE`	All three parameters are used to search an ArcotID and the `userNameOrAlias` is treated as `alias`. If no matching ArcotID is found, then an error code is returned. If `orgname` is NULL, then this parameter is ignored and the other two parameters are used to find a matching ArcotID.
`USERNAME_AND_ALIAS_MODE`	The search is first based on `USERNAME_ONLY_MODE`, if the ArcotID is not found then `ALIAS_ONLY_MODE` is used.

Syntax

```
isArcotIDAvailableEx(userNameOrAlias, appctx, orgname);
```

Parameters

Parameter	Type	Description
`userNameorAliasID`	string	User name and the alias, which are the unique user identifier associated with the Arcot card. **Note:** The alias is obtained from the user and stored in the ArcotID as an attribute with the specified application context.
`appctx`	string	The name of the application context to which the user is logging in to. For example, the user with a single ArcotID can access different portals (savings account, insurance, and credit card) of a bank account. Application context is supplied by the ArcotID Issuance application during ArcotID issuance.
`orgName`	string	The name of the WebFort organization to which the user belongs.

Returns

If the method is successful, then it returns `TRUE`. If the method is unsuccessful, then it returns `FALSE`.

Example

```
var arcotClient = new ArcotClient();
var orgName = "safebank";
var appctx = "safebank-online";

var userAlias" = "jdoe-sb";

var response = arcotClient.isArcotIDAvailableEx(userAlias, appctx, orgName);
```

SignChallengeEx2()

This API selects an ArcotID from a list of ArcotIDs available to it, based on the input parameters (`userNameOrAlias`, `appctxName`, `orgName`) and uses it to sign the incoming challenge. ArcotID selection is based on the lookup mode that is set, see Table 6-2.

Note: The content and format of the signed challenge returned by `SignChallengeEx2()` is same as that returned by `SignChallengeEx()`. If you want to use the `appctx` and `alias` for logging, then you must pass this information to WebFort.

Syntax

```
SignChallengeEx2(challenge, PIN, userNameOrAlias, appctxName, orgName)
```

Parameters

Parameter	Type	Description
`challenge`	string	The encoded challenge string.
`PIN`	string	The ArcotID password.
`userNameorAlias`	string	User name and the alias, which are the unique user identifier associated with the Arcot card. **Note:** The alias is obtained from the user and stored in the ArcotID as an attribute with the specified application context.

Parameter	Type	Description
`appctxName`	string	The name of the application context to which the user is logging in to. Application context is supplied by the ArcotID Issuance application during ArcotID issuance.
`orgName`	string	The name of the WebFort organization to which the user belongs.

Returns

Base-64 encoded string that includes the digital signature and the Arcot certificate, which contains the encrypted public key.

Example

```
var arcotClient = new ArcotClient();
var orgName = "safebank";
var appctx = "safebank-online";
var challengeString = "gCcBwHe/XkIxMjM0";
var userAlias" = "jdoe-sb";
var pinString = "123456";

var response = arcotClient.SignChallengeEx2(challengeString, pinString,
userAlias, appctx, orgName);
```

RemoveArcotIDEx()

This API selects an ArcotID from a list of ArcotIDs available to it, based on the input parameters (`userNameOrAlias`, `appctxName`, `orgName`) and deletes it. ArcotID selection is based on the lookup mode that is set, see Table 6-2. After the ArcotID is deleted it is not available for authentication.

Syntax

`RemoveArcotIDEx(userNameOrAlias, appctx, orgname, storageType)`

Parameters

Parameter	Type	Description
`userNameorAlias`	string	User name and the alias, which are the unique user identifier associated with the Arcot card. **Note:** The alias is obtained from the user and stored in the ArcotID as an attribute with the specified application context.
`appctx`	string	The name of the application context to which the user is logging in to. Application context is supplied by the ArcotID Issuance application during ArcotID issuance.
`orgName`	string	The name of the WebFort organization to which the user belongs.
`storageType`	string	Specifies the medium, where the ArcotID is stored. See "StorageType" for more information.

Returns

If the method is successful, then it returns `TRUE`. If the method is unsuccessful, then it returns `FALSE`.

Example

```
var arcotClient = new ArcotClient();
var orgName = "safebank";
var appctx = "safebank-online";


var userAlias" = "jdoe-sb";


var storageType="hd";


var response = arcotClient.RemoveArcotIDEx(userAlias, appctx, orgName,
storageType);
```

Chapter 7
Deploying the ArcotID Client

This chapter describes the installation process for the ArcotID Clients. The following topics are included in this chapter.

- Deploying Flash Client
- Deploying Java Client
- Deploying Native Plugin
- Upgrading ArcotID Client
- Uninstalling the Native Plug-In
- Installation Directory
- Registry Changes

Note: Check for the supported operating environment, before performing the client installation. See “Operating Environment” for more information.

Deploying Flash Client

The ArcotID flash client is in the `ArcotIDClient.swf` file. This file must be stored on a Web server or application server.

Place the `ArcotIDClient.swf` file in a directory anywhere on the Web server or application server. The URL of this directory must be added as the clientBaseURL attribute in the JavaScript that instantiates the client on the Web page.

Deploying Java Client

The following are the files for deploying different ArcotID Java applets:

- `ArcotApplet.jar`

  Signed applet for Sun JVM.

- `ArcotAppletRaw.jar`

  Unsigned applet for Sun JVM.

Place the above files in a directory anywhere on the Web server or application server. The URL of this directory should be added as the clientBaseURL attribute in the JavaScript that instantiates the client on the Web page.

Deploying Native Plugin

This section discusses the process to install ArcotID browser plug-in on Windows and Mac operating systems.

- Deploying on Windows
- Deploying on Mac OS X

Deploying on Windows

The following two types of installation are covered for Windows:

- Web-Based Installation of Native Plug-In on Internet Explorer
- Local Installation of Native Plug-In on Internet Explorer
- Web-Based Installation of Native Plug-In on Mozilla Firefox

Web-Based Installation of Native Plug-In on Internet Explorer

The `arcotplugin_win32.cab` file contains the ArcotID native client. This file must be stored on a Web server or application server.

Place the `arcotplugin_win32.cab` file in a directory anywhere on the Web server or application server. The URL of this directory must be added as the clientBaseURL attribute in the JavaScript that instantiates the client on the Web page.

Local Installation of Native Plug-In on Internet Explorer

The ArcotID native client is also packaged as an EXE file that can be installed directly on a user's Windows operating system.

To install the ArcotID Client on Windows:

1. Navigate to the appropriate directory and double-click the `arcotplugin_win32.exe` file.

   The Welcome to the ArcotID Client 6.0 Setup screen appears.

2. Click **Next** to proceed with the installation.

   The License Agreement screen appears.

3. Read the agreement carefully and select the **I accept the terms in the license agreement** option and click **Install**.

   The Installing ArcotID Client 6.0 screen appears and installs the software on the client's machine with required settings and files, see "Installation Directory" for more information on the files.

   After the successful installation, Installation Complete screen appears.

4. Click **Finish** on this screen to exit the Wizard.

Web-Based Installation of Native Plug-In on Mozilla Firefox

The `arcotplugin.xpi` file contains the ArcotID native client. This file must be stored on a Web server or application server. Place the `arcotplugin.xpi` file in a directory anywhere on the Web server or application server. The URL of this directory must be added as the clientBaseURL attribute in the JavaScript that instantiates the client on the Web page.

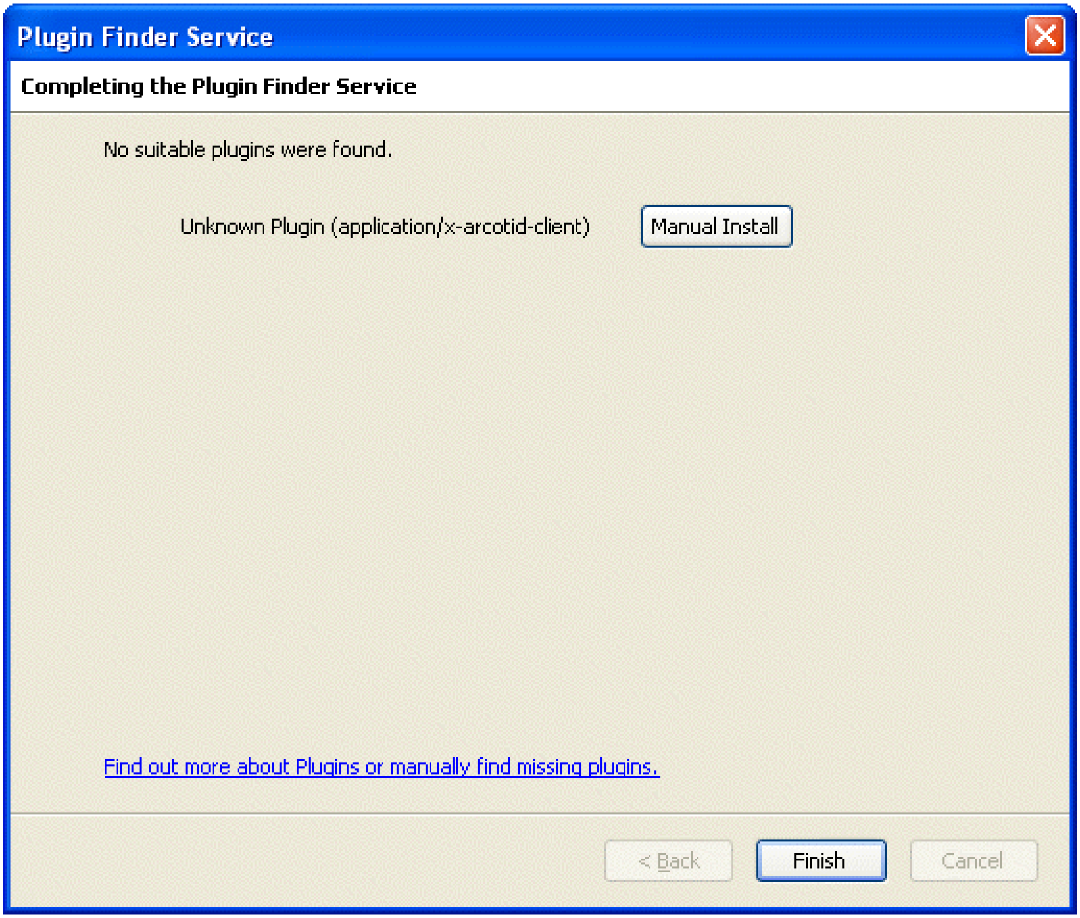
When the user browses to the application pages, the `arcotiplugin.xpi` is automatically downloaded by the browser and opened for installation on the user machine.

Perform the following steps to install the ArcotID Native Client plug-in on your system:

1. Click to install the new plug-in detected by the application.

   The Plugin Finder Service screen (Figure 7-1) appears.

Figure 7-1 ArcotID Plug-In



Plugin Finder Service

Completing the Plugin Finder Service

No suitable plugins were found.

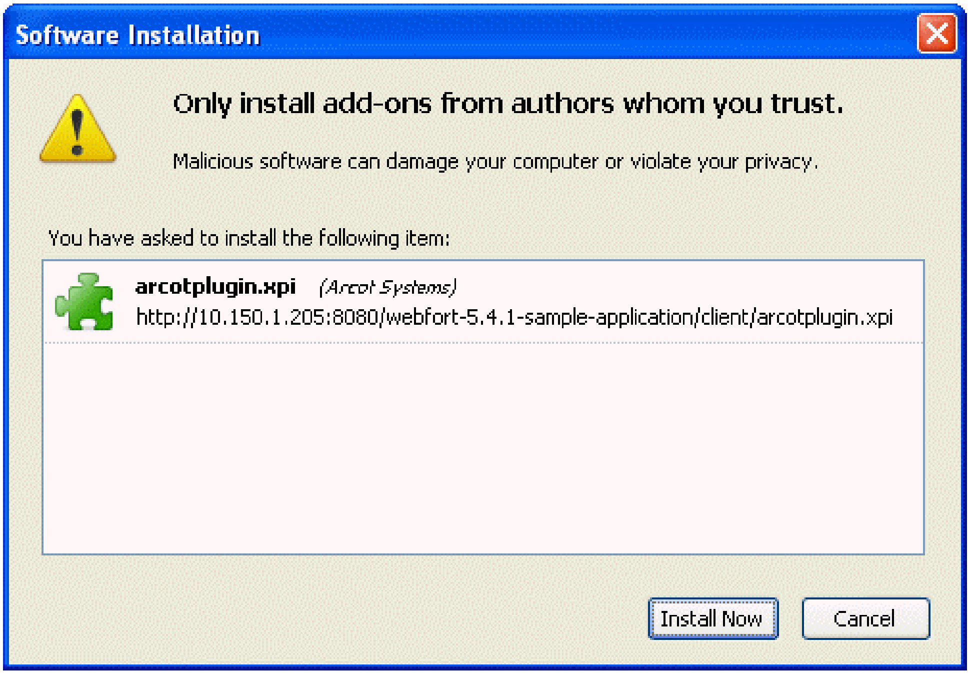
Unknown Plugin (application/x-arcotid-client)

Find out more about Plugins or manually find missing plugins.

2. This dialog lists the ArcotID Client plug-in as **x-arcotid-client**. Click **Manual Install** to install the plug-in.

   The Software Installation screen (Figure 7-1) appears.

Figure 7-2 ArcotID Plug-In



Software Installation

Only install add-ons from authors whom you trust.

Malicious software can damage your computer or violate your privacy.

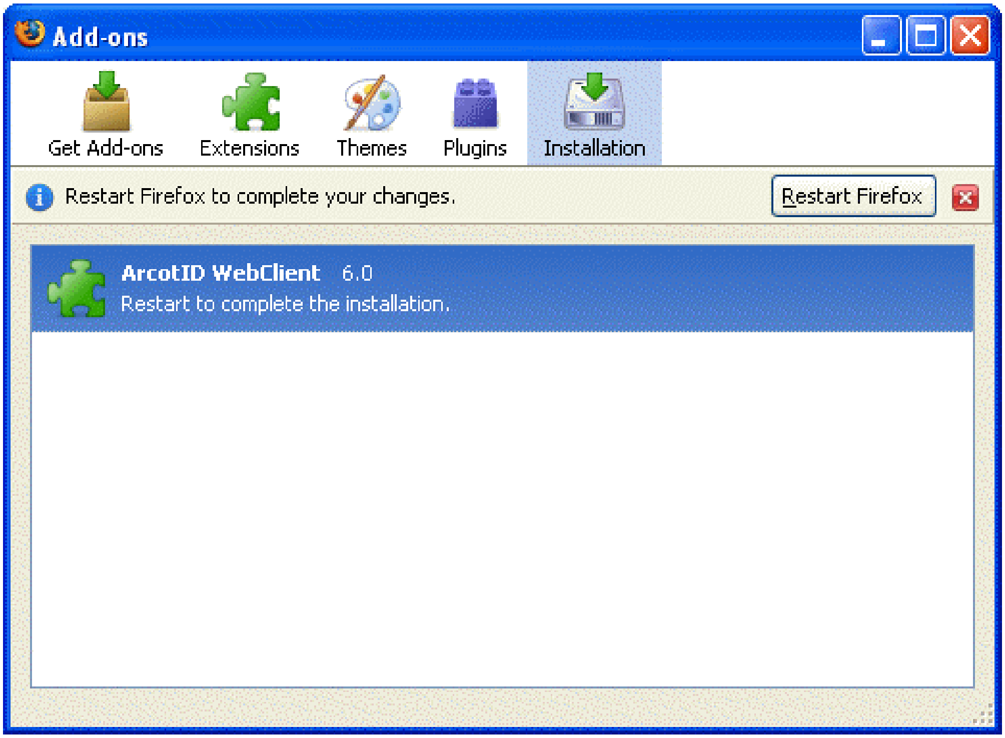
You have asked to install the following item:

arcotplugin.xpi *(Arcot Systems)*
http://10.150.1.205:8080/webfort-5.4.1-sample-application/client/arcotplugin.xpi

3. Click **Install Now** in the Software Installation dialog.

   After successful installation, the **Add-ons** screen (Figure 7-1) appears and lists the Arcot WebFort Client 6.0.

Figure 7-3 ArcotID Client Installation Successful



Restart Firefox to complete your changes.

ArcotID WebClient 6.0

Restart to complete the installation.

4. Click the **Restart Firefox** button to restart the Web browser.

Deploying on Mac OS X

Web-Based Installation of Native Plugin

The `arcotplugin.dmg` file contains the ArcotID native client. This file must be stored on a Web server or application server.

Place the `arcotplugin.dmg` file in a directory anywhere on the Web server or application server. The URL of this directory must be added as the `clientBaseURL` attribute in the JavaScript that instantiates the client on the Web page.

Installing the Native Plugin

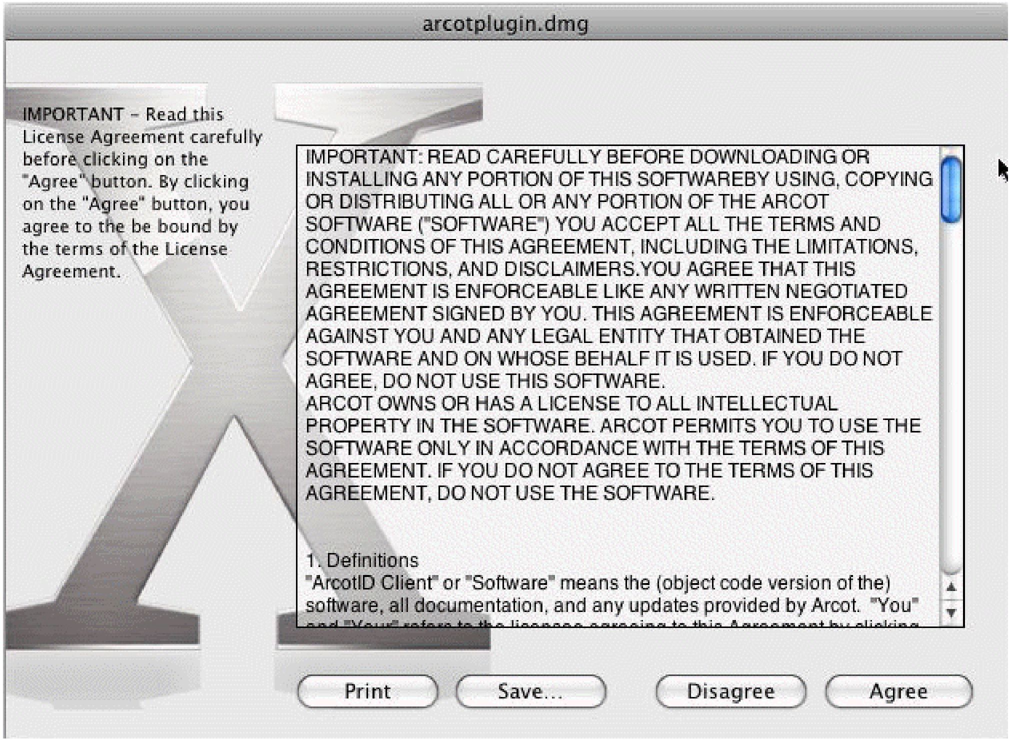
When the user browses to the application pages, the `arcotiplugin.dmg` is automatically downloaded by the browser and opened for installation on the user machine.

Perform the following steps to install the ArcotID Native Client plug-in on your system:

1. Open the `arcotplugin.dmg` file to start the installation.

   The License Agreement screen appears as shown in the following figure.

   Figure 7-4 License Agreement



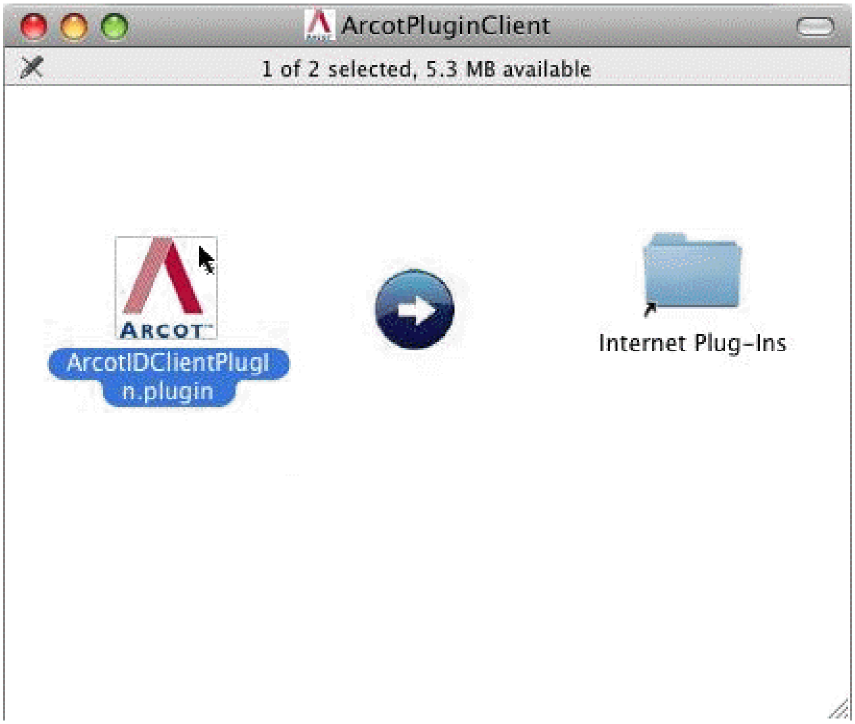
IMPORTANT - Read this License Agreement carefully before clicking on the "Agree" button. By clicking on the "Agree" button, you agree to the be bound by the terms of the License Agreement.

IMPORTANT: READ CAREFULLY BEFORE DOWNLOADING OR INSTALLING ANY PORTION OF THIS SOFTWAREBY USING, COPYING OR DISTRIBUTING ALL OR ANY PORTION OF THE ARCOT SOFTWARE ("SOFTWARE") YOU ACCEPT ALL THE TERMS AND CONDITIONS OF THIS AGREEMENT, INCLUDING THE LIMITATIONS, RESTRICTIONS, AND DISCLAIMERS.YOU AGREE THAT THIS AGREEMENT IS ENFORCEABLE LIKE ANY WRITTEN NEGOTIATED AGREEMENT SIGNED BY YOU. THIS AGREEMENT IS ENFORCEABLE AGAINST YOU AND ANY LEGAL ENTITY THAT OBTAINED THE SOFTWARE AND ON WHOSE BEHALF IT IS USED. IF YOU DO NOT AGREE, DO NOT USE THIS SOFTWARE.
ARCOT OWNS OR HAS A LICENSE TO ALL INTELLECTUAL PROPERTY IN THE SOFTWARE. ARCOT PERMITS YOU TO USE THE SOFTWARE ONLY IN ACCORDANCE WITH THE TERMS OF THIS AGREEMENT. IF YOU DO NOT AGREE TO THE TERMS OF THIS AGREEMENT, DO NOT USE THE SOFTWARE.

1. Definitions
"ArcotID Client" or "Software" means the (object code version of the) software, all documentation, and any updates provided by Arcot. "You" and "Your" refers to the licensee agreeing to this Agreement by clicking

2. Click the **Agree** button to accept the terms and proceed further.
3. The **ArcotIDPluginClient** drag-and-drop screen appears.

Figure 7-5 ArcotID Client Plug-In



4. Drag the `ArcotIDClientPlugin.plugin` file and drop it in the Internet Plug-Ins directory by following the on-screen instructions.

   The ArcotID Client plug-in is installed successfully.

Upgrading ArcotID Client

Upgrading is not necessary on a user's computer for the Flash client or Java applet since the latest Flash client or Java applet is downloaded to a user's computer each time they are invoked.

To upgrade the ArcotID native client, install the newer native client. The installer for the new client will automatically remove the old client and install the new client.

Uninstalling the Native Plug-In

This section lists the steps to uninstall ArcotID Client:

- Uninstalling on Windows
- Uninstalling on Mac OS X

Note: The uninstallation process does not delete the ArcotIDs from the local machine. The user has to manually delete the `.aid` files.

Uninstalling on Windows

To uninstall the ArcotID Client:

1. Navigate to the appropriate directory and double-click the `arcotplugin_win32.exe` file.

   The ArcotID Client Application Maintenance screen appears.

2. Click **Next** on the this screen to proceed with the uninstallation.

   The Program Maintenance screen appears.

3. Select the **Remove** option and click **Next**.

   The Remove the Program screen appears.

4. Click **Back** to change any of the settings or **Remove** to uninstall the software.

   After the software is uninstalled, the Installation Complete screen appears with a success message.

5. Click **Finish** to exit the wizard.

Uninstalling on Mac OS X

To uninstall the ArcotID Client:

1. Navigate to the `/Library/Internet Plugins` folder.
2. Drag the `ArcotIDPluginClient` folder and drop it in the `Trash` folder.

Installation Directory

The ArcotID Client software creates and modifies files on a user's computer. This section lists the files for Windows and Mac operating systems.

Windows Installation Directory

The following table lists the files for Windows.

Table 7-1. ArcotID Client Directory Structure

Destination	Client Type	Files
<*%SYSTEMDRIVE%*>:\Program Files\Arcot Systems\ArcotID Client	Native Client	• ArcotClient.dll DLL containing both the Arcot plug-in and Arcot CSP. • LICENSE.txt License information for the ArcotID Client. • ArcotCSPInstall.exe Program that is used to register the Arcot CSP. • ArcotClient.sig Signature file that is used when the Arcot CSP is registered.
<*%SYSTEMDRIVE%*>:\Program Files\Common Files\Arcot Shared	Native Client	Folders, • conf Default configuration information for the Arcot client. • images Image files used by ArcotID Client.

Table 7-1. ArcotID Client Directory Structure

Destination	Client Type	Files
`Windows\System32`	Native Client	• `ArcotProgressGUI.dll` DLL used by the ArcotID Client. • `ArcotCM.dll` DLL used by the ArcotID Client. • `ArcotPK11_5_0.dll` Arcot PKCS#11 module • `ArcotCardMgrGUI.dll` DLL used by the ArcotID Client. • `ArcotATMGUI.dll` DLL used by the ArcotID Client.
<%SYSTEMDRIVE%>`:\Documents and Settings\<userid>\My Documents\ArcotIDs\`	Native Client and Signed Java Applet	In each user's "`%APPDATA%`" directory `arcot` directory is created. It contains: • `conf` Directory where user preferences are stored. • `ids` Directory where ArcotID files (`.aid` files) are stored.

Mac OS X Installation Directory

The following table lists the files for Mac OS X.

Table 7-2. ArcotID Client Directory Structure

Destination	Client Type	Files
`/library/Internet Plug-Ins/ArcotIDPluginClient/contents`	Native Client	• `MACOS/ARcotIDClientPlugin` File containing the Arcot plug-in. • `Resources` • `info` This file is read by the Web browser to collect the display information.

Registry Changes

The Arcot Native client software modifies the register on user's computer. The following new registry keys are created.

- `HKEY_CURRENT_USER\Software\Arcot Systems`, which is used to store the user-specific settings.
- `HKEY_LOCAL_MACHINE\SOFTWARE\Arcot Systems`, which is used to store the settings that apply to all users.

Chapter 8
ArcotID Client Tools

This chapter describes the following tools used with ArcotID Client:

- ArcotID CSP Registration Tool
- ArcotID Tool

ArcotID CSP Registration Tool

The `ArcotCSPInstall.exe` file provides the Arcot CSP registration tool. It is stored on the user's machine when the Native client is installed. This is a command-line tool, when invoked, it registers the `ArcotClient.dll` as a CSP within Microsoft Windows CAPI (Cryptographic API). It includes the `ArcotClient.dll` in its directory and registers it as a CSP using the signature that is stored in the `ArcotClient.sig` file.

Usage

Register CSP

```
ArcotCSPInstall.exe install
```

Unregister CSP

```
ArcotCSPInstall.exe remove
```

Alternatively click on the **Start** -> **Programs** -> **Arcot Systems** -> **Register Arcot Client as a CSP** to register the ArcotID Client.

ArcotID Tool

ArcotIDTool.exe is a command line tool that is not installed as part of the Arcot client, but available separately. This tool is used for testing and troubleshooting. It tests the functionality of importing keys and certificates into the *key vault* of an ArcotID. The keys and certificates to be imported must be stored in PKCS#12 or PFX format. The tool is invoked as follows:

Usage

```
arcotidtool ImportP12sWallet arcotid_filename arcotid_PIN
number_of_P12files_toimport P12filename_1 P12password1, P12filename_2
P12password2,…
```

Where,

- arcotid_filename is the file name of the ArcotID.
- arcotid_PIN is the password of the ArcotID.
- number_of_P12files_toimport is the number of PKCS#12 or PFX files that will be imported into the ArcotID.
- P12filename_1 is the file name of the first PKCS#12 or PFX file.
- P12password1 is the password of the first PKCS#12 file or PFX file to be imported.
- P12filename_2 and P12password2 are the filename and password of the PKCS#12 or PFX file.

Example

```
arcotidtool ImportP12sWallet tooltest1_10_0_21_125.aid 123456 1 MyCerts.p12
123456
```

This command imports the P12 file MyCerts.p12 into the ArcotID tooltest1_10_0_21_125.aid.

Chapter 9
ArcotID Client Compatibility

The ArcotID Client is compatible with previous versions of clients.

Server Support

The ArcotID client will continue to support the necessary set of features to be compatible with any standard WebFort 5.x and 6.x server installations.

ArcotID Support

The client will work with ArcotIDs created with previous versions of WebFort servers.

New ArcotID format attributes are designed so that an older client can still use the ArcotID for basic purposes such as authentication. The existing 4.x client will not support the new 5.0 card attributes, it would just ignore them without crashing or producing error messages.

New ArcotID's that are protected with device locking (double camouflage; user PIN and machine PIN) will be ignored by previous clients. This will be done by changing wallet version field in the `ASN.1` of the device-locked ArcotID.

Since the ArcotID file extension is changed, new ArcotID files downloaded by the ArcotID Client 5.0 client will not be recognized by any old version of client installed on the same machine.

Chapter 10
Integrating with Third-Party Applications

This chapter describes the process of configuring Adobe Reader and Adobe Acrobat to use Arcot PKCS#11 module for digital signing.

- Integrating with Adobe Reader 7.0
- Integrating with Adobe Acrobat 7.0

Integrating with Adobe Reader 7.0

Arcot PKCS#11 module can be configured with Adobe Reader 7.0 and higher, the following steps describes the configuration steps for 7.0:

1. Open Adobe Reader.
2. Click the **Document** menu and choose **Security Settings**.

   The *Security Settings* page appears.
3. Expand the **Digital IDs** menu and select **PKCS#11 Modules and Tokens**.
4. In the upper-right pane, click **Add Module**.

   The *Locate a PKCS#11 Module* dialog box appears.
5. Browse to the `\Windows\`**`System32`** directory. Select the `ArcotPK11_5_0.dll` file and click the **Open** button.
6. The Arcot PKCS#11 module appears in the list of modules in the upper-right pane. The **Module Manufacturer ID** must have an entry **Arcot Systems, Inc.**, which ensures that the module has been successfully installed.

   Now, when a PDF is signed, Digital IDs stored in the Key Vault of an ArcotID will be displayed as option for signing. The password of the ArcotID has to be entered as the PKCS#11 PIN to access the Digital IDs stored in the ArcotID.

Integrating with Adobe Acrobat 7.0

Arcot PKCS#11 module can be configured with Adobe Acrobat 7.0 and higher, the following steps describes the configuration steps for 7.0:

1. Open Adobe Acrobat.
2. Click the **Advanced** menu and choose **Security Settings**.

   The *Security Settings* page is displayed.
3. Expand the **Digital IDs** menu and select **PKCS#11 Modules and Tokens**.
4. In the upper-right pane, click **Add Module**.

   The *Locate a PKCS#11 Module* dialog box appears.
5. Browse to the `\Windows\`**`System32`** directory. Select the file `ArcotPK11_5_0.dll` and click the **Open** button.
6. The Arcot PKCS#11 module will appear in the list of modules in the upper-right pane. The **Module Manufacturer ID** must have an entry **Arcot Systems, Inc.**, which ensures that the module has been successfully installed.

   Now, when a PDF is signed, Digital IDs stored in the Key Vault of an ArcotID will be displayed as option for signing. The password of the ArcotID has to be entered as the PKCS#11 PIN to access the Digital IDs stored in the ArcotID.

Appendix A
Source Code Samples

This appendix describes the deployment of ArcotID sample applications and the directory location to examine the files.

- Deploying Sample Applications
- Code Samples

Deploying Sample Applications

Perform the following steps to deploy the ArcotID Client:

1. Install `WebFort6ClientSample.war` on the application server (for example Apache Tomcat), in the following location:

   `<APP_SERVER_HOME>\webapps`

   Note: The deployment procedure depends on the application server that you are using. Refer to your application server vendor documentation for detailed instructions.

2. Restart the application server.

After the WAR file is deployed, the following sample applications are available:

- Login sample with applet:

  http://<Host_name>:<Port_name>/WebFort5ClientSample/loginApplet.html

- Login sample with native client:

  http://<Host_name>:<Port_name>/WebFort5ClientSample/loginActiveX.html

- Login sample with flash client:

  http://<Host_name>:<Port_name>/WebFort5ClientSample/loginFlash.html

- Login sample that detects multiple clients and attempts to use the best available.

  http://<Host_name>:<Port_name>/WebFort5ClientSample/login.html

The following URL is used to configure the URL of the WebFort Server:

http://<Host_name>:<Port_name>/WebFort5ClientSample/setup.html

The following URL is used to *upload* a test ArcotID for use with the samples. The ArcotID must be issued by the same WebFort installation that is used for the authentication.

http://<Host_name>:<Port_name>/WebFort5ClientSample/uploadArcotID.html

Code Samples

The following directories containing the sample source code.

- `/WebFort5ClientSample`

  Contains the home HTML file for each sample application.
- `/WebFort5ClientSample/client`

  Contains the Arcot applets and flash client used by the sample applications.
- `/WebFort5ClientSample/images`

  Contains image files used by the sample applications.
- `/WebFort5ClientSample/js`

  Contains JavaScript files used by the sample applications.
- `/WebFort5ClientSample/jsp`

  `Contains JSP's used by the sample applications.`
- `/WebFort5ClientSample/WEB-INF`

  Contains classes used for the servlets that enable the sample applications to communicate with the WebFort server (get challenge, verify signed challenge.)

Appendix B Glossary

AID file extension	Files with the `.AID` extension are ArcotID files stored by the Arcot Client software.
Application Preferences	Attributes set through JavaScript in a Web application that invokes the ArcotID Client affect the client's behavior.
ArcotID	Is a *secure software credential* used for strong authentication, protect digital IDs in the same fashion as a physical smart card or USB security token.
ArcotID Attribute	A name-value pair stored in an ArcotID that contains extra attributes associated with an ArcotID, these attributes affect the behavior or appearance of the ArcotID.
ArcotID Client	The general term to refer to the set of Arcot client types; Native Client, Flash Client, Signed Java Applet, Unsigned Java Applet.
ArcotID Extension Directory	Additional folder (directory) on the end user's computer where the ArcotID client software looks for additional ArcotIDs in addition to the default location or attached USB drives.
ArcotID Password	Each ArcotID has a password, which consist of letters, numbers, and other characters. This password is used when the ArcotID is used to provide first factor of two-factor authentication, something the user knows.
Cryptographic Service Provider (CSP)	Arcot provides a CSP that enables the ArcotID Client to be used by applications designed to work with Microsoft CAPI (Crypto API).
Device Locking	Device Locking feature is set when an ArcotID is issued, which makes that ArcotID unusable on other computers. This is done by using configurable computer characteristics set by the ArcotID issuer to append additional characters to the user entered ArcotID password.

Digital ID	The combination of a private key and an X.509v3 digital certificate containing a public key. Digital IDs have been referred to by many names such as PKCS#12, digital certificate, signing certificate, or email certificate. Arcot is following the lead set by Adobe and Microsoft to standardize on the term *Digital ID*.
DNS Domain restrictions	An ArcotID is restricted to be used only within the domain from which it was originally downloaded from.
Flash Client	The flash client is the version of the ArcotID Client implemented as a flash *movie* that runs in the Adobe Flash Player 9 or higher.
Issuer Preferences	Attributes in an ArcotID file by its issuer to customize the behavior of the ArcotID Client software when using the ArcotID.
Key Vault	An attribute in an ArcotID that can protect one or more Digital IDs. The Digital IDs are protected by a symmetric key that is stored on the Arcot Key Authority server. Each time a digital ID in Key Vault is accessed, the ArcotID is used to authenticate to the Key Authority server, retrieve the symmetric key, then decrypt the Digital ID for use.
Native Client	Is operating system-specific native code that provides the ability to download an ArcotID onto a user's system through ActiveX in IE. It strongly authenticates a user through the Web, and provides CSP and PKCS#11 interfaces so that PKI-enabled applications can use digital IDs protected within the ArcotID.
Partial Hash	A partial hash of the password can be optionally stored in the ArcotID. The presence of the partial hash enables a portion of the invalid passwords to be tested on the client-side. This prevents user lockouts due to typos. However, systematic attempts like brute-force attack on the PIN will still be prevented. This feature is being deprecated and will be dropped completely in the ArcotID 6.0 client releases.
PKCS#11	PKCS#11 is a set of standard cryptographic APIs that allow PKI-enabled applications to use a digital ID.
Preferences	Preferences allow changing the behavior the ArcotID Client. There are three types of preferences Issuer Preferences, Application Preferences, and User Preferences.

Psuedo-PIN	A Pseudo-Pin is a pin, when hashed contains a partial hash that matches the partial hash stored in the ArcotID. This feature is being deprecated and will be dropped completely in the ArcotID 6.0 client releases.
Roaming Digital ID	A digital ID that is stored on the Arcot SignFort server and is used for digital signing through the ArcotID Client embedded in Adobe Acrobat or Adobe Reader.
Signed Java Applet	The Signed Java Applet is the version of the ArcotID Client implemented in Java that runs as a Java Applet in the user's Web browser. It has the ability to permanently store an ArcotID on a user's hard disk or USB flash token.
Unsigned Java Applet	The Unsigned Java Applet is the version of the ArcotID Client implemented in Java that runs as a Java Applet in the user's Web browser. It can only store an ArcotID in memory for the browser session. The Unsigned Java Applet cannot store an ArcotID permanently to the hard disk or a USB flash token.
USB Security Token	A Universal Serial Bus (USB) security token is a device that contains specialized cryptographic processing and storage hardware to securely store digital IDs. A USB security token requires driver software that must be installed by the user.
User Preferences	Behavior of the ArcotID Client that is set by a user in the ArcotID Client GUI.

Index