



## CSP<sup>id</sup> 2.1 User's Guide

# Simplifying the Management of Cryptographic Credentials

Version 2.1.0  
Apr 29, 2010

**Abstract:** This document describes a platform-agnostic software product that acts as a universal key store as well as a cryptographic service provider. Accessed via Microsoft CAPI, or via its industry standard PKCS#11 interface, CSP<sup>id</sup> ensures that authorized security-enabled applications have instant access to the user's latest certificates and private keys without the need to synchronize or replicate credentials among those applications. Using platform-independent PKCS #15 key stores, CSP<sup>id</sup> simplifies the migration of credentials between workstations with different operating systems. After presenting a technical overview of the product, this document covers the installation, administration, and use of CSP<sup>id</sup>.

## **CSP<sup>id</sup> User's Guide**

Information in this document is subject to change without notice and does not represent a commitment on the part of Information Security Corporation. The software described in this document is furnished under a license agreement or nondisclosure agreement. The software may be used or copied only in accordance with the terms of the agreement. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose other than the purchaser's personal use without the prior written permission of Information Security Corp.

CSP<sup>id</sup> software is commercial computer software and, together with any related documentation, is subject to the restrictions on U.S. Government use as set forth below.

**RESTRICTED RIGHTS LEGEND:** Use, duplication, or disclosure by the United States Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software Clause at DFARS 52.227-7013. "Contractor/manufacturer" is Information Security Corporation, 1141 Lake Cook Road, Suite D, Deerfield, IL 60015-9461.

The U.S. International Traffic in Arms Regulations (ITARs) (22 CFR 125.03) prohibits the dissemination of certain types of technical data to foreign nationals.

Protected by U.S. Patents No. 5,274,707, 5,373,560, and 5,699,431.

CSP<sup>id</sup> is a trademark of Information Security Corp. Other product and company names mentioned in this document may be the trademarks of their respective owners.

The cryptographic functionality of CSPid is provided by CDK 7.0, ISC's FIPS 140-2 compliant cryptographic module. In addition, CSPid uses the following open source software packages redistributable under the terms of the LGPL or other licenses:

FLTK, Version 1.1.7: Copyright© 1998-2005 Bill Spitzak and others.  
CSPid is based in part on the work of the FLTK project (<http://www.fltk.org>).  
<http://www.fltk.org/>

PCSC, Version 1.01: Copyright© 2001 Dmitry Basko.  
See the readme.txt file located in the pcsc subfolder of the installation for complete license information.  
[http://www.bds.dogma.net/pc\\_sc.htm](http://www.bds.dogma.net/pc_sc.htm)

CSP<sup>id</sup> 2.1 User's Guide, First Edition (Apr 2010)  
©2007-10 Information Security Corporation. All rights reserved.

### **Information Security Corporation**

1011 Lake Street, Suite 425  
Oak Park, IL 60301

Phone: +1 847 405-0500  
Fax: +1 708 445-9705

Website: <http://www.infoseccorp.com>  
E-mail: [sales@infoseccorp.com](mailto:sales@infoseccorp.com)

## Table of Contents

---

<b>1. Introduction .....</b>	<b>8</b>
1.1. Overview.....	8
1.2. CSP <sup>id</sup> Architecture.....	8
1.3. CSP <sup>id</sup> Events.....	9
1.4. CSP <sup>id</sup> DAS Support.....	11
<b>2. Installation .....</b>	<b>12</b>
2.1. Overview.....	12
2.2. System Requirements.....	13
2.3. Windows .....	14
2.4. UNIX.....	14
<b>3. Configuration.....</b>	<b>15</b>
3.1. Using CSP <sup>id</sup> with a Web Browser .....	15
3.1.1. CAPI-based Browsers.....	15
3.1.2. Netscape-based Browsers.....	15
3.2. Using CSP <sup>id</sup> with Java Applications.....	15
3.3. Using CSP <sup>id</sup> with Apache Tomcat .....	16
3.4. The CSP <sup>id</sup> Configuration File .....	17
3.4.1. Windows .....	17
3.4.2. UNIX.....	17
3.4.3. CSP <sup>id</sup> Events.....	17
3.4.4. Configuration Options .....	18
<b>4. Using CSP<sup>id</sup> .....</b>	<b>25</b>
4.1. The CSP <sup>id</sup> Management Tool .....	25
4.1.1. Two Views of the Key Store.....	26
4.1.2. Importing Credentials.....	27
4.1.3. Exporting Credentials.....	28
4.1.4. Deleting Credentials.....	28
4.1.5. Changing Your Password .....	29
4.1.6. Initiating Password Reset.....	30
4.1.7. Handling a User's Password Reset Request (PRA only).....	30
4.1.8. Completing Password Reset.....	31
4.1.9. Viewing Client Applications .....	31
4.1.10. Registering With Applications .....	32
4.1.11. Renewing Your Certificates.....	32
4.1.12. The About Dialog .....	32
4.1.13. Terminating the Management Tool .....	33
4.1.14. The Management Tool Command Line.....	33
4.2. The CSP <sup>id</sup> Command Line .....	35
4.3. The CSP <sup>id</sup> Audit Trail .....	38

## CSP<sup>id</sup> User's Guide

4.3.1.	Windows .....	38
4.3.2.	UNIX.....	38
<b>5.</b>	<b>CSP<sup>id</sup> Password and Key Management.....</b>	<b>39</b>
5.1.	Password Management .....	39
5.1.1.	Password History.....	39
5.1.2.	Forced Password Change.....	39
5.1.3.	Administrative Password Reset .....	39
5.1.4.	Password Timeout.....	40
5.2.	Key Management.....	40
5.2.1.	Encrypted Memory.....	41
5.2.1.1.	Encrypted Memory on Windows 2000 and Windows XP.....	42
5.2.1.2.	Encrypted Memory on Windows Vista and Windows 7.....	42
5.2.1.3.	Encrypted Memory on UNIX-based Systems .....	43
5.2.2.	Password Timeout Implementation Details .....	43
5.2.3.	Key Management in the CSP <sup>id</sup> Management Tool.....	43
<b>6.</b>	<b>CSP<sup>id</sup> Libraries.....</b>	<b>44</b>
6.1.	The PKCS #11 Library .....	44
6.2.	The Microsoft Windows Library .....	44
<b>7.</b>	<b>Quality Assurance Issues.....</b>	<b>45</b>
7.1.	Testing Methodology.....	45
7.2.	Known Issues .....	46
7.2.1.	Windows.....	46
7.2.2.	Netscape 4.x or higher w/o PSM.....	46
7.2.3.	Netscape 4.75 or higher w/PSM 1.4.....	46
7.2.4.	Internet Explorer .....	46
7.2.5.	Microsoft Outlook.....	46
<b>8.</b>	<b>Net-Centric Applications.....</b>	<b>47</b>
8.1.	Role-Based Authentication.....	47
8.2.	Role-Based Encryption .....	48
8.3.	Confidentiality within a Community of Interest.....	48
8.4.	Expanded Storage and Enhanced Security for Private Keys.....	49
8.5.	Brokered Authentication - "Need-to-Know" Control Over Sensitive Resources.....	50
<b>9.</b>	<b>Future Development.....</b>	<b>52</b>
9.1.	Future Releases .....	52
<b>10.</b>	<b>References.....</b>	<b>53</b>
<b>11.</b>	<b>Appendix A: Default Configuration File.....</b>	<b>55</b>
<b>12.</b>	<b>Appendix B: The PKCS #15 Key Storage Format.....</b>	<b>62</b>
12.1.	PKCS #15 .....	62

12.2.	Object Syntax.....	62
12.3.	Confidentiality.....	62
12.4.	Integrity .....	63
12.5.	Initialization .....	63
<b>13.</b>	<b>Appendix C: A Sample PKCS #15 Key Store.....</b>	<b>64</b>

## List of Figures

---

Figure 1: CSP <sup>id</sup> 2.0 Architecture Diagram.....	9
Figure 2: Modifying the <code>java.security</code> File .....	16
Figure 3: Modifying Your Java Code .....	16
Figure 4: Modifying Java SSL/TLS Properties .....	16
Figure 5: Modifying the Tomcat <code>server.xml</code> File .....	16
Figure 6: The CSP <sup>id</sup> System Tray Menu .....	25
Figure 7: The CSP <sup>id</sup> Manager's Main Window (Standard View).....	26
Figure 8: The CSP <sup>id</sup> Manager's Main Window (Advanced View) .....	26
Figure 9: Importing a File .....	27
Figure 10: Exporting a File .....	28
Figure 11: Confirming Certificate Deletion.....	28
Figure 12: Confirming Private Key Deletion.....	29
Figure 13: Changing Your CSP <sup>id</sup> Password.....	29
Figure 14: Viewing Registered Applications .....	31
Figure 15: Viewing the About Dialog .....	32
Figure 16: The Command Line Interface.....	35
Figure 17: Watch Officers Sign Messages with Shared Role Key on DAS Server .....	47
Figure 18: Commander Decrypts Documents Using Role Key on DAS Server.....	48
Figure 19: Col Members Decrypt Documents Using Col Key on DAS Server .....	49
Figure 20: Officer Retains Access to His Entire Key History.....	50
Figure 21: A DAS Server Used to Impose Strong "Need-to-Know" Controls over Sensitive Resources.....	50
Figure 22: A Sample CSP <sup>id</sup> Configuration File With Default Settings.....	61

**List of Tables**

---

Table 1: CSP <sup>id</sup> Events .....	11
Table 2: System Requirements .....	13
Table 3: Installation Options .....	18
Table 4: CSP Options .....	18
Table 5: Log Options.....	19
Table 6: Password Options .....	20
Table 7: Netscape/Mozilla Options.....	20
Table 8: Import/Export Options .....	21
Table 9: Event Handling Options.....	23
Table 10: DAS Options .....	24
Table 11: Management Tool Command Line Options .....	34
Table 12: Command Line Options.....	38
Table 13: Test Configurations .....	45

# CSP<sup>id</sup> User's Guide

## 1. Introduction

### 1.1. Overview

This document describes a software package that manages a user's X.509 credentials (*i.e.*, certificates and private keys), securely protecting them while making them available upon demand (and appropriate user confirmation) to any security-enabled application and allowing them to be easily migrated between workstations in an operating system-independent manner.

CSP<sup>id</sup> is a virtual smartcard that maintains a central repository for private keys and X.509 certificates on behalf of its owner. It provides a secure environment for cryptographic operations that applications can access via Java, PKCS#11, or Microsoft CAPI.

The CSP<sup>id</sup> system:

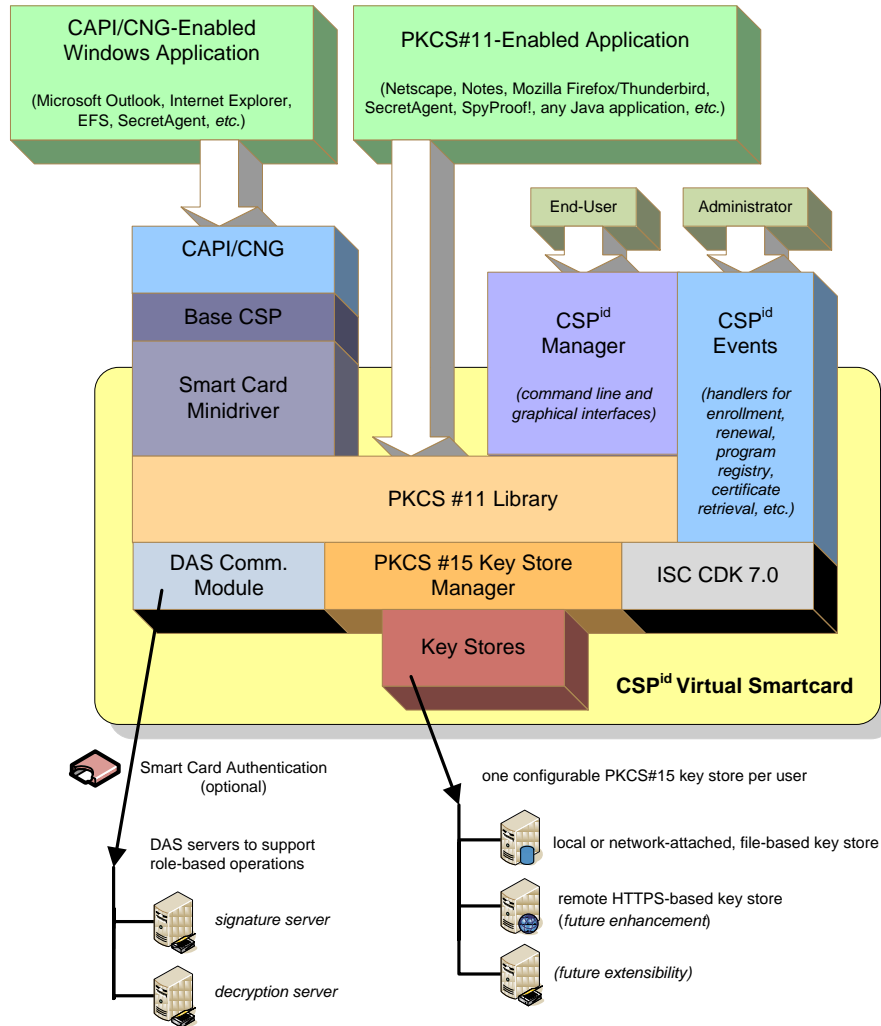
1. provides a single OS-independent credential store that may be shared by all security-enabled applications on the user's system
2. provides superior protection for private keys and overcomes password change/reset issues with Windows, Internet Explorer, and Mozilla
3. simplifies enterprise-wide credential management: users need not replicate keys among applications, and may effortlessly migrate credentials between workstations
4. provides administrative control over security policy settings and can be augmented to support key escrow, key recovery, and/or password reset functions
5. provides event hooking enabling the execution of external programs based on events
6. reduces help desk costs and PKI training requirements

A major step forward in credential management, CSP<sup>id</sup>'s programming interfaces allows administrators to build custom, nearly user-transparent PKI enrollment, key rollover, credential backup, and other management tools that dramatically simplify the PKI experience for an organization's end-users.

### 1.2. CSP<sup>id</sup> Architecture

The CSP<sup>id</sup> system consists of four components:

- a credential store: an encrypted file stored by the system on a local fixed disk, on removable media, or on a remote network drive
- a PKCS #11 library providing access to the credential store and certain private key operations
- a smart card minidriver that is used by the Microsoft Windows Base Smart Card Cryptographic Service Provider (CSP) to provide access to the integrated PKCS #11 library to Microsoft CryptoAPI-enabled applications (such as Internet Explorer, Microsoft Outlook, and Microsoft Outlook Express)
- a set of tools that permit the end-user to easily configure and maintain the CSP<sup>id</sup> system



**Figure 1: CSP<sup>id</sup> 2.0 Architecture Diagram**

As illustrated in the above diagram, a PKCS #11 library (based on ISC's FIPS-validated CDK 7.0), a PKCS #15 key store manager, a smart card minidriver, and the CSP<sup>id</sup> management tools represent the core components of the CSP<sup>id</sup> system. These tools and the provided APIs allow users to manage their key stores and easily integrate CSP<sup>id</sup> into existing applications. As of version 2.0, the DAS communication module allows CSP<sup>id</sup> to provide all applications (including Outlook and Thunderbird S/MIME) high-assurance "role-based" signature and decryption operations that rely on remote private keys, possibly stored on an HSM (requires DAS 1.8 or above).

### 1.3. CSP<sup>id</sup> Events

CSP<sup>id</sup> exposes a portion of its programmable "event model interface" through its configuration file. This interface, fully described in Chapter 3, allows an administrator to manage an end-user's system and credentials by means of custom scripts that are automatically executed in response to certain trigger events. Among the events for which such custom "handlers" can be installed are:

- initial PKI enrollment or key import

## CSP<sup>id</sup> User's Guide

- certificate renewal/key rollover
- certificate import
- manual selection of the “Register with Applications” command

Event Name	Triggers	Configuration Options
<i>Start up</i>	The CSP <sup>id</sup> Manager starts and the user successfully authenticates.	The CSP <sup>id</sup> Manager will execute these commands allowing the import/removal of DAS or issuer certificates or other operations that need to occur daily.
<i>Enrollment</i>	The CSP <sup>id</sup> Manager starts and no credentials are found.	<p>CSP<sup>id</sup> provides three choices for handling this event:</p> <ul style="list-style-type: none"> <li>• transfer credentials from the CAPI “personal” store</li> <li>• run one or more commands</li> <li>• open a URL</li> </ul> <p>The options are tried in the order listed above. If there are still no private key objects present after running the option the next option is attempted. If there are still no private key objects present after running all the options the Renewal event is run.</p>
<i>Renewal</i>	<p>The CSP<sup>id</sup> Manager starts and all of the existing signing credentials, or all of the existing encrypting credentials, will be invalid within a specified number of days,</p> <p>The user selects the <i>Renew My Certificates</i> menu command.</p>	<p>CSP<sup>id</sup> supports a single trigger setting:</p> <ul style="list-style-type: none"> <li>• lead time prior to expiration</li> </ul> <p>and provides two event handling options:</p> <ul style="list-style-type: none"> <li>• open a URL</li> <li>• run one or more commands</li> </ul>
<i>New Signing Credential</i>	A signing certificate is imported and a matching private key is found in CSP <sup>id</sup> .	CSP <sup>id</sup> creates necessary entries in the user's CAPI “personal” store; if the certificate was imported from a third-party application, CSP <sup>id</sup> can run one or more external commands.
<i>New Encryption Credential</i>	An encrypting certificate is imported and a matching private key is found in CSP <sup>id</sup> .	CSP <sup>id</sup> creates necessary entries in the user's CAPI “personal” store; if the certificate is imported from a third-party application, CSP <sup>id</sup> can run one or more external commands.
<i>New DAS Signing Credential</i>	A DAS-enabled signing certificate is imported and DAS support is enabled.	CSP <sup>id</sup> creates necessary entries in the user's CAPI “personal” store; if the certificate was imported from a third-party application, CSP <sup>id</sup> can run one or more external commands.
<i>New DAS Encryption Credential</i>	A DAS-enabled encrypting certificate is imported and DAS support is enabled.	CSP <sup>id</sup> creates necessary entries in the user's CAPI “personal” store; if the certificate is imported from a third-party application, CSP <sup>id</sup> can run one or more external commands.
<i>Register with Applications</i>	<p>The user selects the <i>Register with Applications</i> menu command.</p> <p>The user imports a recently issued PKCS #12 file and P12IMPORTREGAPPS is non-zero.</p>	In addition to configuring the CAPI store, Netscape-based applications, and SecretAgent 5, CSP <sup>id</sup> can run one or more external commands.

Table 1: CSP<sup>id</sup> Events

#### 1.4. CSP<sup>id</sup> DAS Support

CSP<sup>id</sup> versions 1.2 and above includes optional DAS support. DAS is a server-side component that provides role-based capabilities enabling users to either sign for a role or decrypt anything encrypted for a particular role. When C\_Decrypt is called CSP<sup>id</sup> communicates with a DAS server in order to unwrap the symmetric key which is then rewrapped for the specific user. CSP<sup>id</sup> then uses the user's private key to unwrap the response from the server. When C\_Sign is called CSP<sup>id</sup> communicates with a DAS server which performs the signature operation. Support for DAS in CSP<sup>id</sup> is accomplished by

- enabling DAS operations via the configuration file
- importing one or more DAS CoI certificates into CSP<sup>id</sup>

With DAS enabled, CSP<sup>id</sup> will assert to applications that the private key for each DAS CoI certificate resides locally and can be used via CSP<sup>id</sup>. To do this, CSP<sup>id</sup> creates a temporary private key object for each DAS CoI certificate when C\_Initialize or C\_CreateObject is called. Each temporary private key has a vendor defined attribute enabling CSP<sup>id</sup> to differentiate real private keys from DAS CoI private keys when C\_Sign or C\_Decrypt are called. These objects are not permanent and will disappear if DAS is not enabled in the configuration file.

When an application asks CSP<sup>id</sup> to decrypt or sign using one of these temporary private key objects, CSP<sup>id</sup> interacts with a DAS server to perform the operation only if the user is a member of the community. For decrypt operations membership is checked using the encryption certificate provided to the DAS server. DAS operations require SSL/TLS client authentication and, for unwrap requests, an encryption certificate (see the DAS documentation for a complete description of a DAS transaction). CSP<sup>id</sup> automatically selects the best certificates for both authentication and rewrap purposes using the notBefore date and, if specified in the configuration, preferred issuer information. **The private keys for the selected certificates may reside either within CSP<sup>id</sup> itself (as software keys) or on a hardware token (e.g., CAC) specified in the configuration file. In the latter scenario CSP<sup>id</sup> can store only the DAS CoI certificates and no private keys are required to reside in CSP<sup>id</sup>.**

Inclusion of DAS support in CSP<sup>id</sup> allows any application to be DAS enabled including Microsoft Outlook and Mozilla Thunderbird. This allows users to send S/MIME encrypt and signed e-mail in a role based fashion rather than a user based fashion. Only members of a particular role will be able to decrypt the e-mail messages and only members of the role will be able to sign e-mail messages as the role. Chapter 8 Net-Centric Applications provides more information on this capability and its uses.

## **CSP<sup>id</sup> User's Guide**

### **2. Installation**

#### **2.1. Overview**

CSP<sup>id</sup> is available on a variety of platforms and the installation process is platform-dependent. The following files are typically installed:

- the CSP<sup>id</sup> management tool (`cspid_ui.exe` or `cspid_ui`) and, if available, its translation files (`*.qm`)
- the CSP<sup>id</sup> command line tool (`cspid_cli.exe` or `cspid_cli`)
- the CSP<sup>id</sup> PKCS #11 Library (`cspid.dll` or `libcspid.so`)
- the `modutil` component of the Mozilla NSS toolset (`nss3.9`)
- a CSP<sup>id</sup> configuration file (`cspid.cfg`) - optional
- the CSP<sup>id</sup> User's Guide (`cspid.pdf`) – optional

On Windows systems the following additional items are installed:

- the CSP<sup>id</sup> Smart Card Minidriver (`cspid.dll`)
- the CSP<sup>id</sup> Virtual Smart Card Reader Driver (`cspidrdr.sys`)
- shortcuts in the Windows Start Menu for viewing this User's Guide and starting the CSP<sup>id</sup> management tool
- a Java PKCS #11 configuration file (`java_pkcs11.cfg`)
- the Microsoft Smart Card Base Cryptographic Service Provider (if not already installed)

## 2.2. System Requirements

The following table lists the minimum version numbers for several of the most common applications that one might wish to use with CSP<sup>id</sup>:

<b>Supported Browsers</b>	Internet Explorer 4 or higher Netscape 4.75 or higher Mozilla 1.0 or higher Firefox 1.0 or higher.
<b>Java</b>	Java 1.5 and higher w/PKCS #11 support (see Java PKCS#11)
<b>Operating Systems</b>	Windows 2000 SP 4 or higher Windows XP SP 1 or higher Windows Server 2003 Windows Vista Solaris 8 or higher for SPARC RedHat Enterprise Linux 4 or higher for x86

**Table 2: System Requirements**

### 2.3. Windows

CSP<sup>id</sup> for Windows is distributed as standard MSI installation package that provides a wizard for individual installation as well as supporting automated installation mechanisms. This setup program first installs the core CSP<sup>id</sup> components; it will then install the translation files, configuration file, and user's guide if they are present in the same folder as the MSI file (you can specify an alternate location: `msiexec.exe /I cspid.msi CONFIGFILELOC=<path to cspid.cfg>`). To associate PKCS#12 (.p12 and .pfx) files with CSP<sup>id</sup> rather than Internet Explorer specify `ASSOCP12=1` on the `msiexe` command line.

To customize the CSP<sup>id</sup> installation process, open the `cspid.cfg` file in an editor prior to installation, and modify the options settings as desired. (See section 3.4 for more information.) Be sure to save the modified configuration file into the same folder as the `cspid.msi` file. Then install the CSP<sup>id</sup> system by executing `cspid.msi` and following the prompts.

After installation on a Windows system, the CSP<sup>id</sup> management tool (see section 4.1) will automatically start up and run in the system tray each time the user logs in. The first time the tool is executed it will ask the user to set a master password.

Under Windows, a registry entry is created so that as each (new or existing) user logs in after installation, CSP<sup>id</sup> will be automatically registered with all Netscape-based profiles located in that user's `Application Data` folder. (The `cspid.cfg` file may be modified to search for Netscape-based application profiles in alternate or additional locations. See section 3.4 for details.)

### 2.4. UNIX

CSP<sup>id</sup> for UNIX is distributed as a single compressed tar ball (`cspid.x.y.z.os.processor.tar.gz`) that must be decompressed and `untar`'d. The resulting `cspid` folder contains an `INSTALL` file that provides platform-specific installation tips. Typically all that is required is to move the `cspid` folder to `/opt/cspid` and to edit the `cspid.cfg` file.

To customize the way CSP<sup>id</sup> behaves, modify the options settings in `cspid.cfg` as desired. (See section 3.4 for more information.) `libcspid.so` and all associated utilities read their program configuration data from `cpsid.cfg` location in one of the following folders (in the order listed): `/etc`, `/usr/etc`, `/opt/cspid` or the folder specified in the `CSPidInstDir` environment variable, if one is set.

After installation, all users should execute the following command to register CSP<sup>id</sup> with Netscape-based PKCS#11 applications on their system:

```
/opt/cspid/cspid_cli -r
```

In the event that the *Renew My Certificates* feature (see section 4.1.11) is not functioning, users may need to set a `BROWSER` environment variable that points to the web browser to use for enrollment.

Ideally `/opt/cspid/cspid_ui` will be configured to automatically start when each user logs into their desktop.

## 3. Configuration

### 3.1. Using CSP<sup>id</sup> with a Web Browser

#### 3.1.1. CAPI-based Browsers

When CSP<sup>id</sup> is installed, the certificates in its key store are copied into the appropriate CAPI stores (and linked to the private keys stored in CSP<sup>id</sup>) and are therefore available to any Microsoft CryptoAPI-based application (such as Internet Explorer and Outlook). However, to ensure that CSP<sup>id</sup> generates and stores users' private keys during PKI enrollment, they should explicitly tell the enrollment application to use the *Microsoft Base Smart Card Crypto Provider* as its cryptographic service provider (CSP). When Internet Explorer, for example, is configured in this way and used for PKI enrollment, it will use CSP<sup>id</sup> to generate and store private keys and to construct certificate requests. Alternatively, the certificate authority may be capable of controlling the CSP that is used and could be configured to only use the *Microsoft Base Smart Card Crypto Provider*.

#### 3.1.2. Netscape-based Browsers

CSP<sup>id</sup> must be registered with Netscape-based programs before it can be used by them. Users may use the CSP<sup>id</sup> management tool's *Registering with Applications* command (see section 4.1.10; or use the appropriate *CSP<sup>id</sup> Command Line* option as described in section 4.2) to configure newly created Netscape-based profiles.

To manually register CSP<sup>id</sup> with a Netscape-based product, just add `cspid.dll` or `libcspid.so` to its list of supported PKCS #11 tokens by opening the product's *Preferences* dialog and modifying the "Security Devices" or "Advanced" properties. Once this is done, *ISC CSPid* should appear in its list of available cryptographic tokens.

To ensure that CSP<sup>id</sup> generates and stores users' private keys during PKI enrollment, users must select *ISC CSPid* as the cryptographic token to use when prompted by the browser.

### 3.2. Using CSP<sup>id</sup> with Java Applications

Some manual configuration of a system is required to use CSP<sup>id</sup> with Java applications.<sup>1</sup> Since complete instructions are available from Sun<sup>2</sup>, only an outline of the required procedures is provided in this section.

To make CSP<sup>id</sup> available to all Java programs on a system, add a line to the `java.security` file that references the Java PKCS #11 configuration file, `java_pkcs11.cfg`, placed in the CSP<sup>id</sup> folder during installation. A sample is provided below:

---

<sup>1</sup> Java 2 version 1.5 or above is required for native PKCS #11 support through the JDK; earlier versions of Java might be able to access CSP<sup>id</sup> on certain platforms, but this use is not supported by ISC.

<sup>2</sup> See <http://java.sun.com/j2se/1.5.0/docs/guide/security/p11guide.html>

```
security.provider.7=sun.security.pkcs11.SunPKCS11 c:/Program Files/CSPid/java_pkcs11.cfg
```

**Figure 2: Modifying the `java.security` File**

To make CSP<sup>id</sup> available to a Java program programmatically, insert the following lines:

```
String configName = "C:/Program Files/CSPid/java_pkcs11.cfg";
Provider p = new sun.security.pkcs11.SunPKCS11(configName);
Security.addProvider(p);
KeyStore ks = KeyStore.getInstance("PKCS11");
// PIN is the CSPid login password.
ks.load(null, PIN.toCharArray());
```

**Figure 3: Modifying Your Java Code**

To use CSP<sup>id</sup> for client authentication in SSL/TLS connections in existing programs, users may have to set the following system properties with the `-D` option. See *Java SSL/TLS* for more information.

```
javax.net.ssl.keyStoreType=PKCS11
javax.net.ssl.keyStorePassword=<password>
javax.net.ssl.keyStore=NONE
javax.net.ssl.keyStoreProvider=SunPKCS11-CSPid
```

**Figure 4: Modifying Java SSL/TLS Properties**

A sample Java program, `cspid.java`, that illustrates how CSP<sup>id</sup> may be used programmatically from within Java code is included in the installation.

### 3.3. Using CSP<sup>id</sup> with Apache Tomcat

To use CSP<sup>id</sup> as the TLS key container for Apache Tomcat, the `java.security` file must be modified as indicated in Figure 2 above, and the `server.xml` file located in `<Tomcat Home>/conf/server.xml` must be changed so that it looks roughly as follows:

```
<Connector port="443" scheme="https" secure="true" clientAuth="false" sslProtocol="TLS"
keystoreType="PKCS11" keystorePass="PASSWORD" />
```

**Figure 5: Modifying the Tomcat `server.xml` File**

With these settings CSP<sup>id</sup> will be used as the server's keystore and trust keystore.

If TLS client authentication is enabled by setting `clientAuth="true"` in the `server.xml` file, only client certificates subordinate to a root certificate stored in CSP<sup>id</sup> will be accepted.

### 3.4. The CSP<sup>id</sup> Configuration File

CSP<sup>id</sup>'s operating characteristics can be controlled by modifying the `cspid.cfg` file before or after installation. If no `cspid.cfg` file is found, CSP<sup>id</sup> will operate with its default internal settings reproduced in Appendix A: Default Configuration File.

#### 3.4.1. Windows

On Windows, the active configuration file is the one `cspid.dll` finds in its own folder upon startup. Additionally, since Windows users often have write access to the configuration file, administrators have the option of making the configuration file tamper evident by setting `SECURE=1` in the configuration file **prior to installation**. When the installation utility detects `SECURE=1` in the configuration file it places a cryptographic hash of the configuration file being installed into the `HKEY_LOCAL_MACHINE` portion of the Windows registry. When the CSP<sup>id</sup> library loads it compares the hash of the current configuration file to that in the registry and refuses to load if they are not the same.

On Windows systems, the CSPid will replace both `JAVA_HOME` and `CSPID_HOME` with appropriate values when used in a configuration file's event handler (`REGAPPCMD`, `ENROLLCMD`, *etc.*). The special `CSPID_UI_AUTH` value will be replaced with a onetime use value that allows the CSPid Management Tool to perform the requested operation. See section 4.1.14 for more information.

#### 3.4.2. UNIX

On UNIX, `libcspid.so` searches the following directories, in order, for `cspid.cfg`: `/etc`, `/usr/etc`, `/opt/cspid`, the directory specified by the `CSPidInstDir` environment variable. By searching for the configuration file in this manner, it is possible to ensure that only the configuration file supplied by the administrator is used.

#### 3.4.3. CSP<sup>id</sup> Events

Before delving into the details of CSP<sup>id</sup> Events (see Table 9: Event Handling Options), consider for a moment the following reasonable scenario: suppose that an administrator wants to facilitate PKI enrollment and application configuration for end users by performing the following, fairly typical, tasks:

1. have a user who has no existing credentials obtain a signing certificate from the enterprise CA using his default web browser,
2. use the signing certificate just obtained to download via client-authenticated SSL a PKCS#12 PDU containing (a possibly escrowed) encryption key,
3. import the PKCS#12 PDU into CSP<sup>id</sup> and then delete it,
4. publish the user's new credentials to the Exchange Global Address List (GAL), and
5. configure Microsoft Outlook to use the new credentials for S/MIME.

Employing CSP<sup>id</sup> event handlers (along with ISC's Credential Management Utility, `cmu`) the administrator can accomplish these five steps by:

## CSP<sup>id</sup> User's Guide

- i. adding the following line to the system's `cspid.cfg` configuration file to cause each occurrence of an *Enrollment* event (an event triggered by the user starting CSP<sup>id</sup> with no credentials in their active key store) to be handled by opening a specified webpage in the user's default browser:

ENROLLURL=<http://ca1.yourorganization.com/enroll>

- ii. adding the following five lines to the configuration file to handle occurrences of the *New Signing Credential* event:

NEWSIGCERTCMD= "C:\CMU\cmu.exe" d "%TEMP%\temp.p12"

NEWSIGCERTCMD= "C:\Program Files\CSPid\cspid\_ui.exe" -i -f "%TEMP%\temp.p12"

NEWSIGCERTCMD= del "%TEMP%\temp.p12"

NEWSIGCERTCMD= "C:\CMU\cmu.exe" p

NEWSIGCERTCMD= "C:\CMU\cmu.exe" m

This handler, whose execution is triggered by the retrieval of a new signing certificate (possibly caused by the completion of the enrollment process handled in step i), invokes `cmu` several times to download the user's escrowed encrypting PKCS #12 file, install it into their key store, publish the new credentials to GAL and configure their default MAPI provider.

### 3.4.4. Configuration Options

Configuration options are detailed in the following tables:

Option	Values	Notes
<i>SECURE</i> (Windows Only)	<ul style="list-style-type: none"> <li>• 0 (no)</li> <li>• 1 (yes)</li> </ul>	If set to 1, the installation program will store a cryptographic hash of the installed configuration file in the Windows Registry. Upon startup, the CSP <sup>id</sup> library checks this hash against the hash of the current configuration file and will refuse to function if the configuration file has been modified.

**Table 3: Installation Options**

Option	Values	Notes
<i>P15URL</i>	The path and filename to use for the PKCS#15 PDU.	On Windows use UNC paths (not mapped network drive letters) to specify the location of the PDU.
<i>USE_OLD_CHANGE_NOTIFY_METHOD</i> (Windows Only)	Set to 1 to use the original method for detecting changes in the PDU file on Windows.	Versions 2.1 and above use Windows folder change notifications to detect changes to the PDU file. This improves performance and reduces network traffic. If set to 1 CSP <sup>id</sup> will load the PDU and compare to its cached copy.

**Table 4: CSP Options**

Option	Values	Notes
<i>LOGLEVEL</i>	<ul style="list-style-type: none"> <li>0 (critical information)</li> <li>1 (information)</li> <li>2 (debug)</li> </ul>	Critical information is always logged to the system log in addition to any specified log file. Debug and information log entries are only logged to the log file if specified.
<i>LOGFILE</i>	The path and filename to use for the log file.	None.
<i>APPLOGFILE</i>	The path and filename to use for the "registered" applications list.	Every application that calls or causes to be called the C_Initialize() function in cspid.dll or libcspid.so is stored in this file and displayed when the user selects "Show Registered Applications".

**Table 5: Log Options**

Option	Values	Notes
<i>PWREQNUM</i>	<ul style="list-style-type: none"> <li>0 (no)</li> <li>1 (yes)</li> </ul>	If set to 1, user created passwords must contain at least one number: 0-9.
<i>PWREQALPHA</i>	<ul style="list-style-type: none"> <li>0 (no)</li> <li>1 (yes)</li> </ul>	If set to 1, user created passwords must contain at least one alphabetic character: a-z or A-Z.
<i>PWREQMIXEDCASE</i>	<ul style="list-style-type: none"> <li>0 (no)</li> <li>1 (yes)</li> </ul>	If set to 1, user created passwords must contain both lower and upper case characters.
<i>PWREQPUNC</i>	<ul style="list-style-type: none"> <li>0 (no)</li> <li>1 (yes)</li> </ul>	If set to 1, user created passwords must contain at least one punctuation mark: .,?!;:'".
<i>PWREQSPEC</i>	<ul style="list-style-type: none"> <li>0 (no)</li> <li>1 (yes)</li> </ul>	If set to 1, user created passwords must contain at least one special character: @#\$\$%^&*()+=-_/>< []{}~`
<i>PWMINLEN</i>	The minimum length of user created passwords.	None.
<i>PWCHANGE</i>	The number of days between required user password changes.	None.
<i>PWHISTORY</i>	The number of unique passwords required before the user can re-use a password.	None.
<i>PWTIMEOUT</i>	<ul style="list-style-type: none"> <li>-1 (never)</li> <li>nonzero (seconds)</li> </ul>	If set to -1, password timeout is disabled. If set to a positive value the library will require the user to enter their password if the library has been inactive for the number of seconds indicated. The minimum value is 2 seconds.

<i>RSTPWAGENTS</i>	A properly formatted base64-encoded PKCS #7 PDU containing one or more password reset agent certificates.	The command line option "p7tob64" will output a properly formatted string given PKCS #7 input file. A properly formatted base64-encoded PKCS#7 PDU, in this case means one whose lines do not exceed 100 characters. When specifying this file as RSTPWAGENTS you must precede each line with an equal sign. <u>This option is only used if SECURE=1 was specified during installation.</u>
<i>RSTPWWMINLEN</i>	The minimum length of the reset password.	This value is the number of random bytes to use for the reset password. The value the user must type in will be 1/3 longer than this value. Any values less than 8 will be ignored.

**Table 6: Password Options**

<b>Option</b>	<b>Values</b>	<b>Notes</b>
<i>NSSEARCHPATH</i>	One or more semi-colon delimited paths that should be searched for Netscape-based profiles in addition to CSP <sup>id</sup> 's default paths.	A common addition for Netscape 4.x users is: %ProgramFiles%\Netscape
<i>NSCREATETRUST</i>	<ul style="list-style-type: none"> <li>• 0 (no)</li> <li>• 1 (yes)</li> </ul>	If set to 1, ephemeral CKO_NETSCAPE_TRUST objects are created and made accessible to applications using CSP <sup>id</sup> 's PKCS #11 interface. These objects tell the calling application to trust the certificates in CSP <sup>id</sup> .

**Table 7: Netscape/Mozilla Options**

Option	Values	Notes
<i>CAIMPORT</i>	<ul style="list-style-type: none"> <li>• 0 (never)</li> <li>• 1 (prompt user for trust)</li> <li>• 2 (always)</li> </ul>	<p>If set to 0, CSP<sup>id</sup> will not store certificates for which it does not store a matching private key.</p> <p>If set to 1, CSP<sup>id</sup> will store certificates without private keys, but it will prompt the user for permission to import (and thus trust) root certificates.</p> <p>If set to 2, as if set to 1, but on UNIX the user will not be asked to trust root certificates.</p>
<i>P12EXPORT</i>	<ul style="list-style-type: none"> <li>• 0 (no)</li> <li>• 1 (yes)</li> </ul>	<p>If set to 1, user's will be allowed to export their credentials as PKCS #12 files.</p>
<i>P12IMPORTCA</i>	<ul style="list-style-type: none"> <li>• 0 (no)</li> <li>• 1 (yes)</li> </ul>	<p>If set to 1, CSP<sup>id</sup> will attempt to import any issuer certificates it finds when importing PKCS #12 files adhering to the CAIMPORT setting. When set to 1, CSP<sup>id</sup> will attempt to include issuer certificates when exporting PKCS #12 files.</p>
<i>P12IMPORTREGAPPS</i>	<ul style="list-style-type: none"> <li>• 0 (disabled)</li> <li>• max hours since issuance</li> </ul>	<p>Causes the CSP<sup>id</sup> UI to trigger the register with applications event when a user imports a PKCS #12 file via the CSP<sup>id</sup> UI command line (<i>cspid_ui.exe</i>) if the certificate was issued within the number of hours specified.</p>
<i>P12CLIIMPDELONMATCH</i>	Keyword	<p>When present the CSP<sup>id</sup> UI will examine the path and filename of PKCS#12 files imported via its command line and, if the keyword is present, delete the file after import. This is typically used to delete PKCS#12 files "opened" from within the user's web browser by setting the value to Temp (Firefox) or Temporary (Internet Explorer).</p>

**Table 8: Import/Export Options**

Option	Values	Notes
<i>STARTUPCMD</i>	One or more commands to execute when the CSP <sup>id</sup> Manager application starts.	None.
<i>ENROLLURL</i>	The URL to open when the enrollment event occurs.	This option is ignored if ENROLLCMD or ENROLLCAPI is set.
<i>ENROLLURL_MESSAGE</i>	Text to display to the user in a message box in place of the default text notifying the user that a browser will open to start enrollment.	Leave blank to use the default text.
<i>ENROLLCMD</i>	One or more commands to execute when the enrollment event occurs.	To specify multiple commands, include multiple ENROLLCMD= lines in the configuration file. This option is ignored if ENROLLCAPI is specified.
<i>ENROLLCAPI</i>	<ul style="list-style-type: none"> <li>• 0 (disabled)</li> <li>• 1 (enabled)</li> </ul>	If set to 1, the CSP <sup>id</sup> Manager will move credentials from the CAPI "personal" store to CSP <sup>id</sup> . <sup>3</sup>
<i>ENROLLCAPI_MESSAGE</i>	Text to display to the user in a message box in place of the default text notifying the user that they may be prompted for their password when moving their credentials from CAPI.	Leave blank to use the default text.
<i>NEWSIGCERTCMD</i>	One or more commands to execute when new signing certificate event occurs.	To specify multiple commands, include multiple NEWSIGCERTCMD= lines in the configuration file.
<i>NEWENCCERTCMD</i>	One or more commands to execute when a new encryption certificate event occurs.	To specify multiple commands, include multiple NEWENCCERTCMD= lines in the configuration file.
<i>NEWDASSIGNCERTCMD</i>	One or more commands to execute when new DAS-enabled signing certificate event occurs.	To specify multiple commands, include multiple NEWSIGCERTCMD= lines in the configuration file.
<i>NEWDASENCCERTCMD</i>	One or more commands to execute when a new DAS-enabled encryption certificate event occurs.	To specify multiple commands, include multiple NEWENCCERTCMD= lines in the configuration file.
<i>RENEWURL</i>	A URL to open with the default web browser when the renewal	This option is ignored if RENEWCMD is specified. If set,

---

<sup>3</sup> Credentials that cannot be exported (they are stored in a smart card or are marked non-exportable) will not be moved. Credentials are moved by exporting the private key, importing it into CSP<sup>id</sup> and then deleting it from the current Microsoft CAPI provider. The current Microsoft EFS credential, however, will be copied to CSP<sup>id</sup>. EFS in Windows 2000 and XP only works with credentials stored in Microsoft CSPs. On Windows Vista, EFS can work with smart card credentials, but it independently caches the CSP of the private key. For these reasons, CSP<sup>id</sup> does not modify the EFS certificate in CAPI's personal store.

	event occurs.	the Renew My Certificates item appears in the CSP <sup>id</sup> Manager.
<i>RENEWURL_MESSAGE</i>	Text to display to the user in a message box in place of the default text notifying the user that a browser will open to start enrollment.	Leave blank to use the default text.
<i>RENEWCMD</i>	One or more commands to execute when the renewal event occurs.	To specify multiple commands, include multiple <i>RENEWCMD=</i> lines in the configuration file. If set, the Renew My Certificates item appears in the CSP <sup>id</sup> Manager.
<i>RENEWDAYS</i>	The number of days before certificate expiration that should be considered when deciding whether the renewal event should occur.	None.
<i>REGAPPSCMD</i>	One or more additional commands to execute when the user invokes Register with Applications.	To specify multiple commands, include multiple <i>REGAPPSCMD=</i> lines in the configuration file.
<i>ALLCERTSISSUERS</i>	Indicates how register with applications should treat certificates without matching private keys. <ul style="list-style-type: none"> <li>• 0 (follow RFC 5280)</li> <li>• 1 (treat as issuers)</li> </ul>	If set to 0, certificates without matching private keys will be treated as issuer certificates only if they contain a basic constraints extension asserting the CA flag. If set to 1, all certificates without matching private keys will be treated as issuer certificates.

**Table 9: Event Handling Options**

Option	Values	Notes
<i>DASENABLE</i>	<ul style="list-style-type: none"> <li>• 0 (no)</li> <li>• Serial Number (yes)</li> </ul>	If set to a CSP <sup>id</sup> DAS Enablement serial number, DAS will be enabled.
<i>DASSERVER</i>	One or more DAS rewrap server URIs.	If provided CSP <sup>id</sup> will use this list of DAS servers rather than expect the full DAS URL to reside in the CoI certificates.
<i>DASSIGSERVER</i>	One or more DAS signing server URIs.	If provided CSP <sup>id</sup> will use this list of DAS servers rather than expect the full DAS URL to reside in the CoI certificates.
<i>DASTIMEOUT</i>	The number of seconds to wait for a DAS server to connect.	If provided CSP <sup>id</sup> will use this value instead of the default value.

<i>EXT_P11_LIB</i>	The PKCS#11 library to use for cryptographic operations when connecting to a DAS server.	If provided CSP <sup>id</sup> will use the first available smart card when authenticating with a DAS server or decrypting a DAS server's response.
<i>EXT_P11_LABEL</i>	The label of the token to use for PKCS#11 operations.	If provided CSP <sup>id</sup> will find and use the slot containing a token with this label. If not present then it will use the first available smart card. This option is most useful when using a hardware security module with multiple slots for password reset operations.
<i>EXT_P11_NAME</i>	The name to display to the user when prompting for the password to the external smart card.	If provided CSP <sup>id</sup> will display this name rather than the card's label returned from the PKCS#11 library.
<i>EXT_P11_SIGN_ISSUER_ID</i>	Either an issuer DN or a AKI value to use when selecting a signing certificate.	If provided CSP <sup>id</sup> will use this information when choosing between multiple signing certificates.
<i>EXT_P11_ENC_ISSUER_ID</i>	Either an issuer DN or a AKI value to use when selecting a encryption certificate.	If provided CSP <sup>id</sup> will use this information when choosing between multiple encryption certificates.
<i>EXT_P11_ENC_LABEL_ID</i>	A "keyword" that indicates that a particular certificate is "better" for encryption than other certificates on the device.	For CAC it should be Encryption.
<i>EXT_P11_SIGN_LABEL_ID</i>	A "keyword" that indicates that a particular certificate is "better" for signing than other certificates on the device.	For CAC is should be Signature.

**Table 10: DAS Options**

## 4. Using CSP<sup>id</sup>

CSP<sup>id</sup> integrates into browsers, e-mail clients, and other system applications in the same manner as a smart card. Once you enter your password, the private keys in the CSP<sup>id</sup> key store are immediately available for use in decryption and/or signing operations in any Microsoft CAPI- or (registered) Netscape-based application.

NOTE: During browser-based PKI enrollment operations, you (or your certificate authority's webpage) must explicitly specify the CSP<sup>id</sup> token. For Internet Explorer select *Microsoft Base Smart Card Crypto Provider*, and for Netscape-based browsers select *ISC CSPid*.

### 4.1. The CSP<sup>id</sup> Management Tool

The CSP<sup>id</sup> management tool is a system tray application (on systems that support the system tray concept) that allows you to perform common management tasks and to view the objects managed by CSP<sup>id</sup>. It normally starts automatically each time you log in. (On UNIX this behavior must be configured by an administrator.)

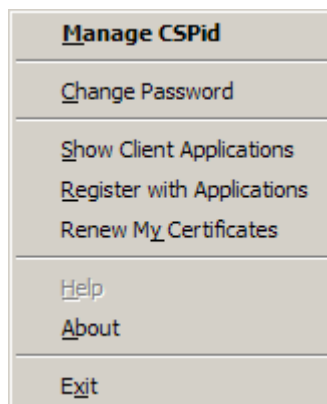


Figure 6: The CSP<sup>id</sup> System Tray Menu

The *Manage CSPid* menu item opens the *CSP<sup>id</sup> Manager*'s main dialog, illustrated below; the remaining menu items are simply shortcuts to identically named items in the main dialog's menu bar.

This section explains how to use the *CSP<sup>id</sup> Manager* (or equivalent system tray menu items) to view the certificates and other objects stored by CSP<sup>id</sup>, and to import into, export from, and delete objects from, your active key store. The *CSP<sup>id</sup> Manager* supports the import and export of PKCS #12 and PKCS #7 PDUs, as well as that of (raw binary ASN.1 DER-encoded) X.509 certificate files.

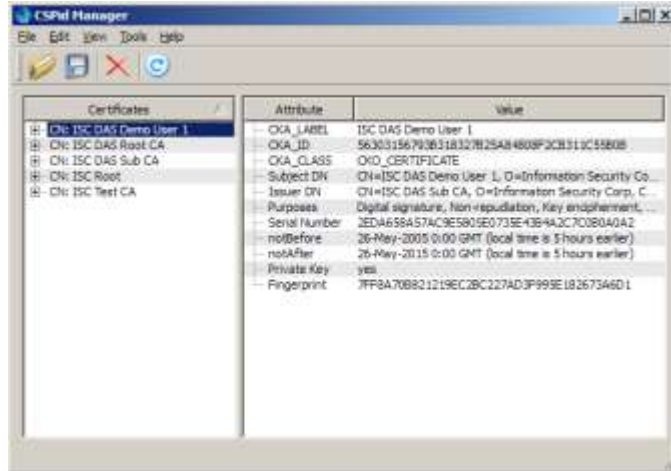


Figure 7: The CSP<sup>id</sup> Manager's Main Window (Standard View)

#### 4.1.1. Two Views of the Key Store

The *CSPid Manager*'s main window initially lists all certificates managed by CSP<sup>id</sup>; this is the default “view” of the CSP<sup>id</sup> key store. When in this view, for example, you can determine if a private key is associated with a particular certificate by selecting that certificate in the left-hand pane and inspecting its *Private Key* property in the right-hand pane.

Selecting *Advanced* from the *View* menu causes the main window to switch from showing only certificates to displaying all objects managed by CSP<sup>id</sup>. *Advanced* view allows you to manage individual certificates, public keys, and private keys.

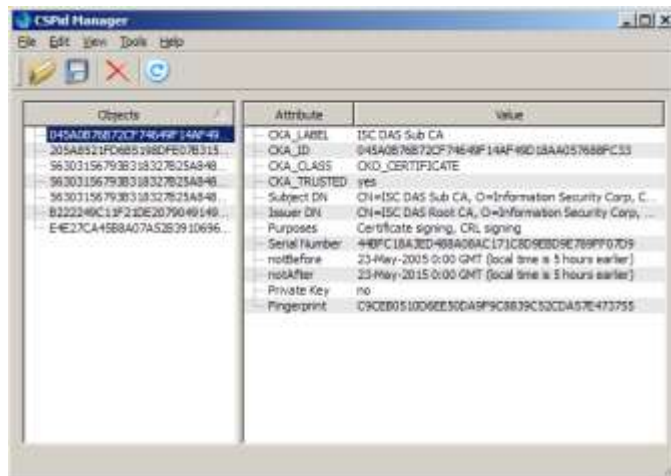


Figure 8: The CSP<sup>id</sup> Manager's Main Window (Advanced View)

#### 4.1.2. Importing Credentials

Choosing *Import* from the *CSPid Manager's File* menu opens the following file selection dialog. Change the type of file to be imported (PKCS #12, PKCS #7, or X.509), if necessary, using the *Files of type* drop down list, select one or more files, and click *Open*. (As usual in such dialogs, several files of the specified type can be selected by using the control and/or shift keys in conjunction with the left mouse button.)

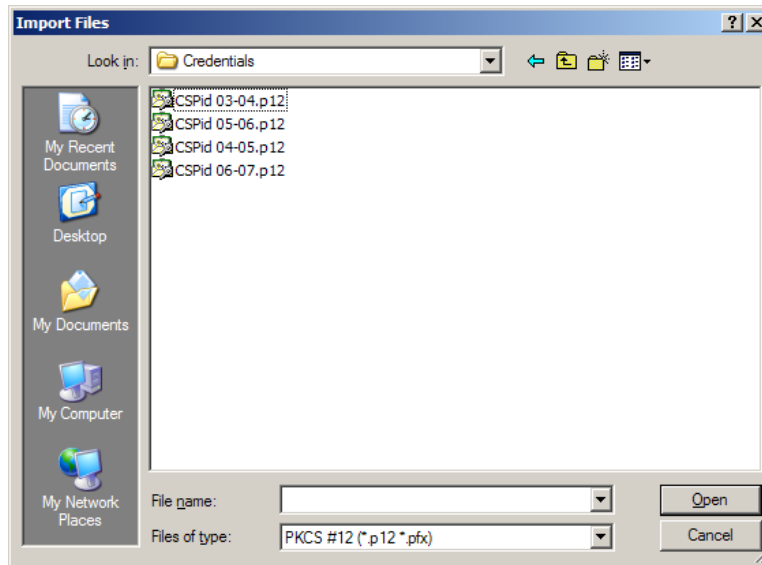


Figure 9: Importing a File

## 4.1.3. Exporting Credentials

Selecting a certificate, then choosing *Export* from the *CSPid Manager's File* menu opens a standard *Save as* dialog. Enter a filename for the file to be created, change its type (PKCS #12, PKCS #7, or X.509), if necessary, using the *Save as type* drop down list, then click *Save*.

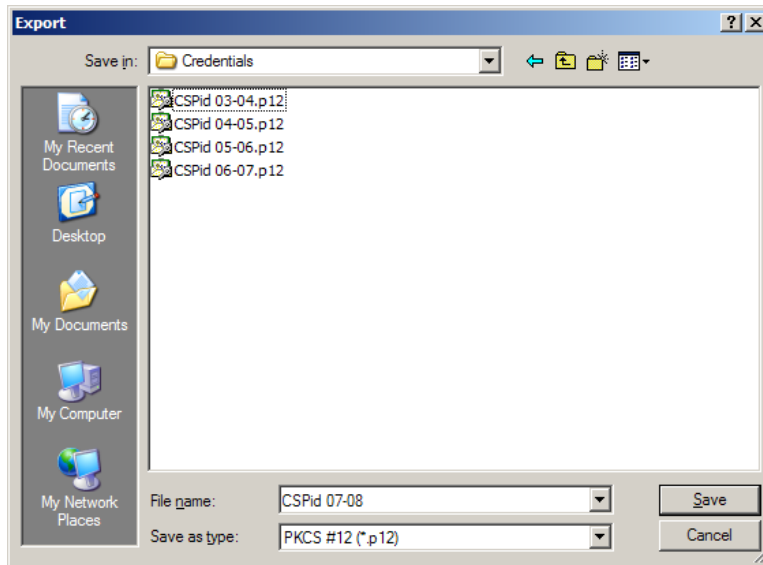


Figure 10: Exporting a File

## 4.1.4. Deleting Credentials

Selecting a certificate (or key in *Advanced* view), and choosing *Delete* from the *CSPid Manager's Edit* menu starts the process of removing a certificate and any associated keys from the CSP<sup>id</sup> key store. You will be prompted to confirm that you really want to delete the selected key or certificate.

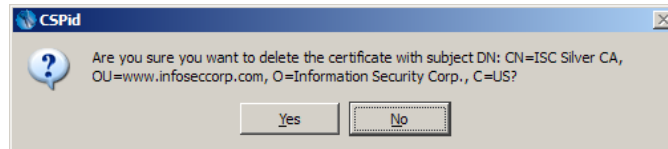
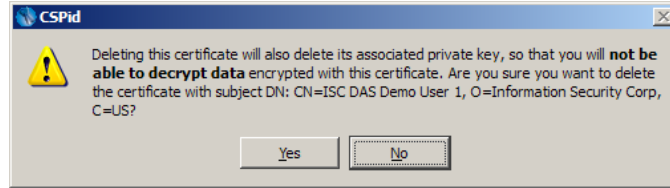


Figure 11: Confirming Certificate Deletion

If the certificate being removed has an associated private key (which, in the standard view, will also be deleted), you will be prompted for confirmation a second time to ensure that you fully understand the consequences of your decision.

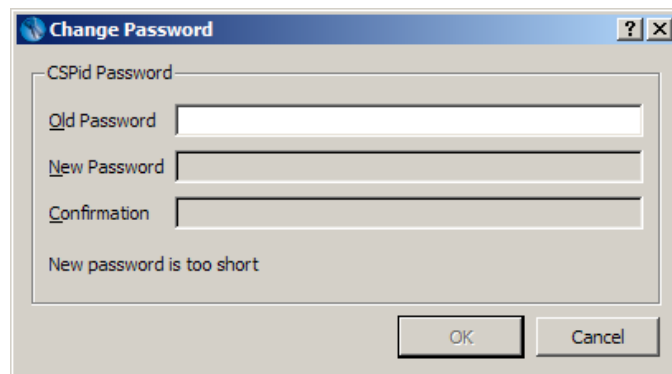


**Figure 12: Confirming Private Key Deletion**

If you select an object in *Advanced* view, and choose *Delete* from the *Edit* menu, only the selected object will be removed from the CSP<sup>id</sup> key store. (This is different from the behavior of *Delete* in the standard view where only certificates are displayed and all associated keys are deleted.)

#### 4.1.5. Changing Your Password

To change the password used to protect private items in your CSP<sup>id</sup> key store, select *Change Password* from the *Tools* menu (or from the system tray menu).



**Figure 13: Changing Your CSP<sup>id</sup> Password**

The *OK* button in this dialog will become enabled once you have entered at least one character in all password fields and the new password satisfies any quality requirements that may be in effect pursuant to an active security policy. (The status line keeps you apprised of any requirements that remain to be satisfied.)

**NOTE:** You will be forced to change your password whenever you upgrade from a previous release of CSP<sup>id</sup> as well as when a security policy specifying one or more Password Recovery Agents is installed on your system by an administrator.

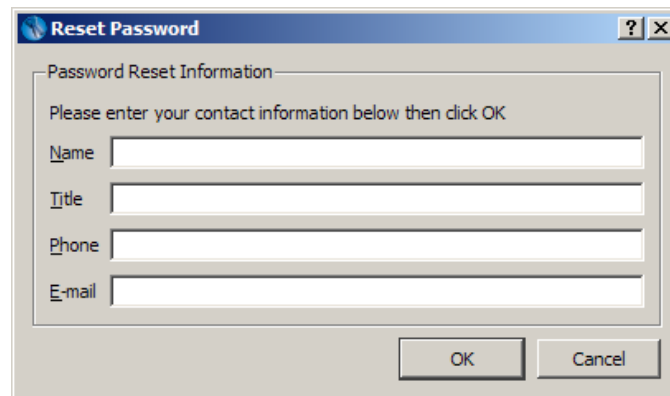
## CSP<sup>id</sup> User's Guide

### 4.1.6. Initiating Password Reset

If you forget your password (and password reset is enabled<sup>4</sup>), you may request that a PRA assist you in resetting it. If password reset is enabled on your system, you'll see a "**Forgot Password?**" link on CSP<sup>id</sup>'s login dialog:



To begin the password reset process, click this link and enter your contact information into the following form:



Once you click **OK**, the supplied information will be e-mailed with an attached password reset (.cpr) file to all designated PRAs using the e-mail addresses in their certificates.

NOTE: The .cpr file contains only a reset password for your system encrypted for each of the PRAs; it does not contain any of your private keys. One of the PRAs should decrypt this file for you and provide you with the reset password which can then be used at the CSP<sup>id</sup> login prompt in place of your forgotten password.

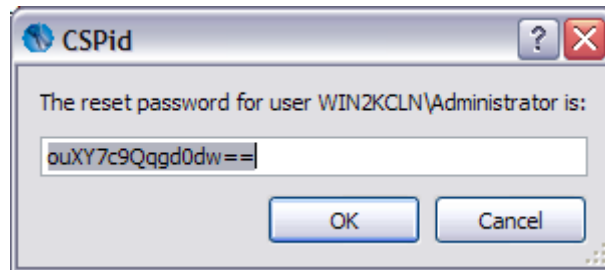
### 4.1.7. Handling a User's Password Reset Request (PRA only)

When a PRA receives a password reset request via e-mail from an end-user, they should first ascertain that the user is who they claim to be -- insist on appropriate identification! The reset password dialog will

---

<sup>4</sup> Password reset is enabled by one or more PRA certificates being specified in the active security policy. See Table 6: Password Options for details.

contain the user name (and, on Windows, the domain name) of the user that was logged in when the password was set. This information allows an administrator to know which account owns the reset password. Administrators use the CSP<sup>id</sup> Manager to open the .cpr attachment (by “opening” the file or by selecting *Open Password Reset File* from the *Tools->Administration* menu). The (case-sensitive) reset password displayed to the administrator can then be returned to the end-user.



#### 4.1.8. Completing Password Reset

Once you have received a (case-sensitive) reset password from a PRA, you may enter it at the CSP<sup>id</sup> Manager login prompt in place of your forgotten password. You will be forced to change your password to complete the password reset process.

NOTE: When you change your password, a new reset password is also generated and the old reset password known to, and supplied by, your PRA will no longer be valid.

#### 4.1.9. Viewing Client Applications

Selecting *Show Client Applications* from the *Tools* menu (or from the system tray menu) will display a dialog listing all applications that have accessed the CSP<sup>id</sup> library since it was installed. Note that registering with applications will not necessarily make them show up in this list. They will appear only after the application uses a key stored in CSP<sup>id</sup>.

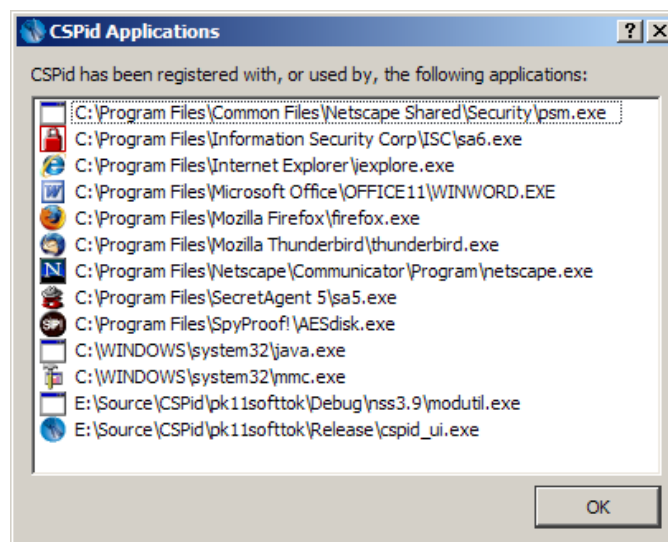


Figure 14: Viewing Registered Applications

## CSP<sup>id</sup> User's Guide

This dialog provides a quick way of determining which applications have potentially accessed your private keys without you having to open and search through your system's event log.

If a Netscape- or other PKCS#11-based application does not appear in this list, it may not be able to use CSP<sup>id</sup> until you have invoked the *Register with Applications* command described next.

### 4.1.10. Registering With Applications

Selecting *Register with Applications* from the *Tools* menu (or the system tray menu) causes CSP<sup>id</sup> to:

- search for Netscape-based profiles along the paths specified in CSP<sup>id</sup>'s configuration file; CSP<sup>id</sup> then registers itself as a PKCS #11 library in each discovered profile using the `NSS_modutil` program
- (*Windows only*) copy and install all certificates in the CSP<sup>id</sup> key store into the following Microsoft CAPI stores: `My`, `addressBook`, `CA`, and `Root`

*Register with Applications* is intended to help users easily migrate their credentials to a new system or to use CSP<sup>id</sup> on multiple systems (with possibly a network-accessible PKCS #15 PDU as their key store). Once this command is run, your system will be aware of all your certificates and private keys, as well as all CA certificates stored by CSP<sup>id</sup>.

### 4.1.11. Renewing Your Certificates

If available, choosing *Renew My Certificates* from the *Tools* menu (or from the system tray menu) will initiate the credential renewal process for your organization as specified in the active security policy. When enabled by an administrator, this command will execute custom scripts that should help simplify the PKI enrollment/ certificate renewal process for you.

### 4.1.12. The About Dialog

Selecting *About* from the *Help* menu (or from the system tray menu) opens an informative dialog that, among other things, refers users to ISC for support and provides the necessary contact information.



Figure 15: Viewing the About Dialog

#### 4.1.13. Terminating the Management Tool

Selecting *Exit* from the *File* menu (or from the system tray menu) will terminate the CSP<sup>id</sup> management tool. (Closing the tool's main dialog minimizes it to your system tray, but leaves it running.)

#### 4.1.14. The Management Tool Command Line

The CSP<sup>id</sup> management tool has limited command line functionality designed for integration with the operating system and to provide functionality for command based events (REGAPPSCMD, ENROLLCMD, RENEWCMD, NEWSIGCERTCMD, etc.).

Switch	Command	Required Options	Notes
-R	reset user pin	-f <file>	Opens the administrative password reset file specified.
-f <file>	none	none	Specifies a filename to open or import when used with the -R and -i options.
-i	import file	-f <file>	Imports the specified file (.p7b, .cer, .der, .p12, .pfx). If used with --cspid-auth %CSPID_UI_AUTH%, CSP <sup>id</sup> will not prompt the user when importing root certificates (MS CAPI may still prompt the user). If -f is a wild card specification a recursive search will be performed to find and install all matching items. Use --p12-pass to specify the import password. Use --showsucces to display a message of your choice on success.
--p12-pass <password>	specifies the PKCS #12 import password with the -i command	none	Allows the PKCS#12 import password to be provided via script rather than input by the user. If the password is incorrect they user will be prompted to enter the password.
--showsucces <message>	specifies a message to display upon successful import	none	Causes the application to display a dialog containing the provided message upon successful import of a file.
--export-all- keys <folder>	export all credentials as PKCS #12 files	--cspid-auth %CSPID_UI_AUTH%	If PKCS #12 export is allowed, this command will prompt the user to create a password to protect the exported files and then save all of the user's credentials as PKCS #12 files in the specified location.
--backup-p15 <folder>	create a copy of the CSPid PKCS #15 file	--cspid-auth %CSPID_UI_AUTH%	This command will create a copy of the user's PKCS #15 PDU in the specified location for

			backup purposes.
<code>--cspid-auth %CSPID_UI_AUTH%</code>	none	none	This option authorizes the user interface to execute sensitive commands. When found in the configuration file the management tool will replace %CSPID_UI_AUTH% with a nonce that is checked when the command is executed.
<code>--move-capi- keys</code>	migrate CAPI credentials	none	This command moves any credentials stored in CAPI/CNG into CSP <sup>id</sup> .
<code>--showmessage &lt;message&gt;</code>	display message box	none	This command displays a dialog with the message provided and an OK button. It can be used to alert the user of an impending event or the completion of an event.
<code>--prog-display &lt;0 1&gt;</code>	display or hide the progress status window	<code>--prog-status &lt;message&gt;</code>	<code>--prog-display 1</code> will cause a window to appear and display the message provided by the <code>-prog-status</code> option.  <code>--prog-display 0</code> will hide the window.
<code>--start-hidden (UNIX-only)</code>	on systems without a system tray start in the background	none	This command starts the GUI without showing the main window (it will prompt for the password, run any events, and then disappear into the background). If the user runs <code>cspid_ui</code> again the background instance will display the manager window. If the user closes the manager window the application returns to the background state. If the user exits the manager window the application quits. This simulates the behavior of the application on systems with a system tray.

Table 11: Management Tool Command Line Options

## 4.2. The CSP<sup>id</sup> Command Line

A command line utility allowing the scripting of certain actions is provided for bulk operations and is used by the installation program.

```

CSPid(tm) Command Line Interface, Version 2.1.0
(C) 2007-2010 Information Security Corp. All rights reserved.

Usage: cspid_cli {-b | -c | -C | -e | -h | -i | -r | -u} [-f <file>] [-I]
        [-p <pwd>] [-P <path>] [-t <pin>] [--pause] [-y]

Functions (specify exactly one):
  --admin-pwdr    decrypt and display a user's reset password
  -b, --p7tob64  base64-encode a PKCS#7 PDU (for RSTPWAGENTS)
  -c, --chgpin   change the CSPid password
  -C, --clear    erase all CSPid objects
  -e, --export   export all CSPid objects; if P12EXPORT=1, certificates
                with private keys are saved as .p12 files
  -h, --help     print this usage information, then exit
  -i, --import   import a PKCS#12, PKCS#7, or ASN.1 DER-encoded file
                --resetpin create a password reset request
  -r, --register  install CSPid into Netscape-based applications and
                SecretAgent 5, and register certificates in CAPI
  -u, --uninstall remove CSPid from Netscape-based applications
                --post url HTTP(S) POST contents of -f argument to specified <url>
                --query opt Query Active Directory for user info spec'd by <opt>
                --make-inf cfg hash configuration file <cfg> and save a registry import
                file <inf> in the file specified by -f

Options:
  --content-type ct specify the content type to use with --post

  -f file
  --filename file  read (-i, -b) the specified file

  -I
  --installonly   skip CAPI certificate registration (with -r)

  -p pwd
  --password pwd  use pwd as the CSPid password

  -P path
  --path path     with -e, export files to path
                with -i, import all PKCS#12 (.p12/.pfx) files in path
                with -r, append path to NSSEARCHPATH

  -t pin
  --p12pin pin    use pin as the PKCS#12 password

  --pause         prompt for a key press before exiting

  -y
  --yestoall      suppress prompts for object deletion and file overwrite

See accompanying documentation for additional information.

```

**Figure 16: The Command Line Interface**

In general, commands cannot be combined on the same command line and certain options apply only to particular commands (see table below). The *register* and *uninstall* options access the `cspid.cfg` file to determine which paths should be searched or use the default values if no paths are specified in the configuration file. If the *path* option is supplied to *register* or *uninstall* as a semi-colon delimited list of directories, those directories are searched in addition to paths specified in the configuration file.

Command	Required Options	Optional Switches	Additional Information
<i>chgpin</i>	none	--pause	Changes the user's CSPid password interactively from the command line.
<i>clear</i>	none	--password, --pause, --yestoall	Erases all objects (certificates, public keys, and private keys) from the user's CSPid store.
<i>export</i>	--path	--p12pin, --password, --pause, --yestoall	Exports all objects from the user's CSPid store.
<i>help</i>	none	none	Displays the information shown in the figure above.
<i>import</i>	--filename or --path	--p12pin, --password, --pause	With --filename, imports a single certificate, PKCS#7 or PKCS#12 file.  With --path, imports all PKCS#12 files found in any of the paths specified (on Windows multiple paths can be specified by separating them with semi-colons).
<i>register</i>	none	--installonly, --password, --path, --pause	With --installonly, registers the CSPid library with Netscape-based applications. The CSPid password is not required.  Without --installonly, it registers with Netscape-based applications and also registers with Microsoft CAPI/CNG. This option requires the CSPid password.
<i>uninstall</i>	none	--path, --pause	Removes the CSPid library integration from Netscape-based applications.
<i>p7tob64</i>	--filename	--pause	Converts the PKCS#7 file specified by --filename into the

			format required by the configuration file for password reset agents.
<code>admin-pwdr</code>	<code>--filename</code>	None	Attempts to decrypt and display the reset password for the password reset file specified by <code>--filename</code> .
<code>resetpin</code>	<code>--filename</code>	none	Creates a password reset file to initiate password reset.
<code>post &lt;url&gt;</code>	<code>--filename</code>	<code>--content-type</code>	Uploads the contents specified in <code>--filename</code> to the URL provided (using either HTTP or HTTPS). The <code>--content-type</code> option is used to specified the content-type HTTP header value when necessary. The default value is <b>application/x-www-form-urlencoded</b> which is used for posting simple forms. When posting multi-part forms the content-type must be specified along with the boundary value.
<code>query &lt;opt&gt;</code>	none	none	There is currently only one option:  1. <b>(windows only)</b> query Active Directory for the user's first name, last name, middle initial, and user name. Output a lowercase string of the following format <b>CN=lastname firstname middleinitial username</b>
<code>make-inf &lt;cfg&gt;</code>	none	none	<b>(Windows only)</b> Create a <code>cspid.inf</code> file for the specified configuration file <code>&lt;cfg&gt;</code> . The <code>.inf</code> can be used in conjunction with the configuration file to update end user systems with a new, authenticated,

			<p>configuration file. The configuration file must be installed in the same location as the original configuration file. The inf file must be installed in the registry (requires administrative access).</p>
--	--	--	---

**Table 12: Command Line Options**

**4.3. The CSP<sup>id</sup> Audit Trail**

CSP<sup>id</sup> includes a configurable audit trail facility that supports logging events to a specified text file as well as to the system event log.

All user actions that modify the key store, export private data, or are related to authentication are written to the system event log regardless of CSP<sup>id</sup> audit trail settings. This includes object creation, object modification, use of the credential export functions in the management tools, password entry, password failure, and password change.

If **debug** level logging is enabled and a log file is specified, CSP<sup>id</sup> will append information to assist in troubleshooting to the log file location. Logging is configured via the CSP<sup>id</sup> configuration file; see section 3.4 for more information.

**4.3.1. Windows**

CSP<sup>id</sup> creates entries in its own CSP<sup>id</sup> section of the Windows Event Log. To view the log, use the Event Viewer accessory provided by Microsoft.

**4.3.2. UNIX**

On UNIX, CSP<sup>id</sup> uses `syslog` to create entries with an identifier of `CSPid` in the system event log. You may need to configure `syslog` to properly capture and store these messages.

## 5. CSP<sup>id</sup> Password and Key Management

### 5.1. Password Management

Password change is accomplished using the standard PKCS #11 `C_SetPin()` call. Users can use the supplied programs, Netscape-based applications, or other PKCS #11 compliant tools to change their passwords. It should be noted, however, that once the user changes their password all applications that currently have the library open will begin receiving `CKR_DEVICE_REMOVED` errors until restarted.

#### 5.1.1. Password History

The configuration file supports the prevention of password re-use. The administrator can specify the number of unique passwords a user must use before they can re-use a past password. The password history is stored by CSP<sup>id</sup> in the PKCS #15 PDU as a string. The first 8 bytes of the string are the random salt chosen the first time the password was set. The remaining values are the 20-byte SHA-1 hash values for each password that the user has used. The string is truncated at each password change to contain only the last *X* password hash values where *X* is the number specified in the configuration file. An iteration count of 512 is used along with the salt to help ensure that it is not possible to do easy table lookups of SHA-1 hash to password by attackers.

#### 5.1.2. Forced Password Change

If the configuration file has specified password change requirements based on time (i.e. that users must change their passwords every *X* days) the library will enforce this requirement by returning a vendor defined error code when the `C_Initialize()` function is called (`0x82050112` is returned in this case). In fact, the `C_Initialize()` function has succeeded but well behaved applications will detect the error code and fail. A message box is displayed by the library alerting the user. If the management tool detects this error it displays the password change dialog to allow the user to easily change their password. For the 10 days preceding a required password change users are asked if they would like to change it when the management tool starts. The date and time of the last password change is stored in the PKCS #15 PDU.

#### 5.1.3. Administrative Password Reset

When the administrative password reset feature is enabled and the user sets, resets, or changes their password, CSP<sup>id</sup> will:

1. generate (in a FIPS 140 compliant manner) a pseudorandom string of at least 8 bytes (see Table 6: Password Options)
2. base64-encode the random string generated in 1 to create a new reset password
3. encrypt as a CMS PDU the reset password under the user password, the reset password itself, and each PRA certificate, and store the result in the user's active key store (a PKCS #15 PDU)
4. use the reset password as a "PBE-recipient" key during re-encryption of all private objects
5. use the reset password as a PBE-recipient key when generating an HMAC-based CMS authentication wrapper for the PKCS#15 PDU

Because of 4, the reset password and user password are on an equal footing with respect to their ability to unwrap private objects, but unlike the user password, the reset password can be recovered (*i.e.*, decrypted)

## CSP<sup>id</sup> User's Guide

using an administrator private key. (Note that administrators cannot recover the actual user password which protects the user in case they have used that password for other purposes.)

So that the reset password is recoverable by any PRA as well as available at runtime to encrypt new or changed private objects, it is stored in the PKCS#15 PDU encrypted under itself, the user password, and all PRA certificates. When the user logs in with either their own password or the reset password provided by a PRA, the system can, in either case, easily recover the reset password and determine which of the two passwords was entered. If the user entered the reset password, they are forced to change their password (thereby generating a new reset password unknown to any PRA).

### 5.1.4. Password Timeout

When the password timeout feature is enabled CSP<sup>id</sup> will require the user to reenter their password after the specified period of inactivity has passed. The method used to cause the calling application to prompt for the password varies depending on how the calling application accesses the library:

- Via Microsoft Windows CAPI/CNG the library's CardAuthenticatePin function will return SCARD\_W\_WRONG\_CHV when the KEK is encrypted. This causes the Microsoft Base Smartcard Provider to prompt the user for their password.
- Via PKCS#11, the information returned from the library's C\_GetSessionInfo function will indicate a state of CKS\_RW\_PUBLIC\_SESSION which tells applications that the card is in the logged out state when timeout has occurred. The calling application can then call C\_Login to authenticate to the library. If the calling application fails to call C\_Login the library will prompt the user for their password if a sensitive value is required.
- If an application attempts an operation that requires the KEK but fails to test and re-authenticate to the library the library itself will display a password prompt. This is the method that the CSP<sup>id</sup> Management Tool uses.

NOTE: When used with a web browser, the server's SSL/TLS server side timeout value may be greater than the CSP<sup>id</sup> timeout value in which case no prompting will occur because the server does not require the user to authenticate again. Sensitive web sites should have a timeout value that is less than or equal to the timeout value set in CSP<sup>id</sup>.

## 5.2. Key Management

CSP<sup>id</sup> stores certificates, public keys, private keys, and PKCS#11 data objects (CKO\_DATA). Following PKCS #11 specifications, objects that are marked CKA\_PRIVATE, TRUE have strong restrictions on which of their attributes are exposed and which can be changed. Similarly, objects marked CKA\_EXTRACTABLE, FALSE cannot be extracted from the token. Finally, certain object attributes can be changed in only one direction. For example, the attribute CKA\_SENSITIVE can be changed from FALSE to TRUE, but not from TRUE to FALSE.

NOTE: The correct handling of such restrictions by CSP<sup>id</sup> might result in unexpected behavior. For example, since most applications generate keys with CKA\_EXTRACTABLE set to FALSE, users may not be able to export such private keys from the device using a PKCS #11-enabled application.

However, for backup and recovery purposes, users can extract their private keys from the CSP<sup>id</sup> system using the integrated management tools; both a graphical user interface and a command line program are supplied. (These tools do not use standard PKCS #11 API calls but operate directly on the active key store as a PKCS #15 PDU.)

Private objects are stored in the active key store as encrypted objects. When in memory, all data is stored in ISC CDK strings or Key objects, both of which meet the FIPS 140-1/2 level 1 requirements for in-memory storage of keying material. When C\_Login is called successfully the encrypted objects in the PKCS#15 PDU are decrypted into encrypted memory. Memory related to specific objects is decrypted temporarily when required to perform operations.

### 5.2.1. Encrypted Memory

When a user logs into CSP<sup>id</sup> all sensitive objects in the PKCS#15 PDU are decrypted and then stored in an object in memory. Prior to version 2.1 this information was stored in memory in plaintext. Beginning with version 2.1, the in memory storage object encrypts sensitive items.

For RSA private keys the following PKCS#11 values are encrypted while in memory:

- CKA\_PRIVATE\_EXPONENT
- CKA\_PRIME\_1
- CKA\_PRIME\_2
- CKA\_EXPONENT\_1
- CKA\_EXPONENT\_2
- CKA\_COEFFICIENT

For ECC private key objects the following values are encrypted while in memory:

- CKA\_VALUE

For secret key objects (resulting from ECDH computations for example) the following values are encrypted while in memory:

- CKA\_VALUE

For data objects that are marked private (resulting from Lotus Notes storing the user's password) the following values are encrypted while in memory:

- CKA\_APPLICATION
- CKA\_OBJECT\_ID
- CKA\_VALUE

Other sensitive information that is encrypted while in memory:

- All copies of the PKCS#15 PDU password
- All copies of the (optional) password reset password
- Temporary values computed when calling C\_Decrypt with a NULL buffer in order to obtain the size of the result.

The encryption scheme varies by platform as discussed below, but always uses a 32-byte key encryption key (KEK) generated at random and stored in memory that cannot be written to the Windows page file, UNIX swap partitions or accessed by other processes. The KEK is generated using the getrand3() method in ISC's FIPS-validated CDK 7 after which getrand3() is called again to transition the PRNG's state to ensure that the KEK does not remain in the PRNG's memory space.

### 5.2.1.1. Encrypted Memory on Windows 2000 and Windows XP

On Windows 2000 and Windows XP, CSP<sup>id</sup> uses CryptProtectData and CryptUnprotectData functions to encrypt sensitive information in memory. The KEK is stored in memory allocated with VirtualAlloc, marked with VirtualLock (to prevent paging), and protected with VirtualProtect (to prevent access by other programs). The KEK is passed into the CryptProtectData and CryptUnprotectData data functions via the optional entropy parameter in order to protect against other processes obtaining the encrypted memory and using CryptUnprotectData.

Using this scheme the PKCS#15 PDU passwords and sensitive keying material are protected against the following:

1. Other programs running on the system cannot access the process' memory and decrypt the in memory information because the KEK is protected and is inaccessible to other applications.
2. Someone in possession of the Windows hibernation file cannot decrypt the in memory information because when Windows hibernates or sleeps it clears information required by the CryptUnprotectData function to decrypt.

NOTE: When Windows is run inside a virtual machine and that virtual machine is "suspended" the KEK will be accessible (as will sensitive operating system protected information) in a file on the host machine that contains the system's memory. In such instances, ISC recommends using the password timeout option with a short timeout period in order to clear the KEK from memory as soon as possible.

### 5.2.1.2. Encrypted Memory on Windows Vista and Windows 7

On Windows Vista/7 CSP<sup>id</sup> uses the CryptProtectMemory and CryptUnprotectMemory functions to encrypt sensitive information in memory. These functions require the data to be a multiple of CRYPTPROTECTMEMORY\_BLOCK\_SIZE (currently 16) and PKCS#5 padding is used to expand plaintext as necessary. CryptProtectMemory is called with the CRYPTPROTECTMEMORY\_SAME\_PROCESS flag to prevent other processes from decrypting the memory. The KEK is stored in protected memory as described for Windows 2000/XP. The KEK is used to super encrypt the output of CryptProtectMemory using the AES Key Wrap algorithm specified in RFC 3394 in order to link the user's password with the encrypted memory to properly suppose password timeout. In this way, it is more difficult to recover the encrypted memory when the password has timed out.

Using this scheme the user's PKCS#15 PDU password and sensitive keying material is protected against the following:

1. Other programs running on the system cannot access the process' memory and decrypt the in memory information because the CRYPTPROTECTMEMORY\_SAME\_PROCESS flag restricts access to the original process.
2. Someone in possession of the Windows hibernation file cannot decrypt the in memory information because when Windows hibernates or sleeps it clears information required by the CryptUnprotectMemory function to decrypt.

NOTE: When Windows is run inside a virtual machine and that virtual machine is "suspended" the KEK will be accessible (as will sensitive operating system protected information) in a file on the host machine that contains the system's memory. In such instances, ISC recommends using the password timeout

option with a short timeout period in order to clear from memory the KEK which is required remove the super encryption prior to calling CryptUnprotectMemory.

### **5.2.1.3. Encrypted Memory on UNIX-based Systems**

On Linux, OSX, and Solaris CSP<sup>id</sup> uses the AES Key Wrap algorithm specified in RFC 3394 to encrypt sensitive information in memory which is padded according to PKCS#5. As on Windows, the KEK is generated and stored in protected memory (mlock is used to prevent it from being swapped; mprotect is used to prevent access by other processes).

Using this scheme the user's PKCS#15 PDU password and sensitive keying material is protected against an attacker accessing the process' memory and decrypting the in memory information. However, not all platforms behave in a consistent manner with regards to the mlock and mprotect functions. Notably, Solaris 8 requires root access to use the mlock function. Thus, it is possible that the KEK could be written to the swap partition. In general, on UNIX-based systems ISC recommends using the password timeout option with a value appropriate to the sensitivity of the keys.

### **5.2.2. Password Timeout Implementation Details**

When the C\_Login function is first called the library spawns a thread that waits for the specified time of inactivity to pass and then encrypts the KEK value with the PKCS#15's password. The encryption algorithm used is the same algorithm used when encrypting private information for storage in the PKCS#15 (see section 12.3). When the user reenters their password, the KEK is decrypted and placed back into protected memory. Thus, when password timeout is enabled all sensitive information is encrypted and the PKCS#15 PDU password is required to decrypt.

### **5.2.3. Key Management in the CSP<sup>id</sup> Management Tool**

Prior to version 2.1 the CSP<sup>id</sup> Management Tool would keep all keys in memory in plaintext form when running. Beginning with version 2.1, in addition to storing the keys in encrypted memory, the CSP<sup>id</sup> management tool will encrypt the KEK, using the same method as described in 5.2.2, when minimized. If the KEK is required for an operation the tool will prompt the user for the password.

## 6. CSP<sup>id</sup> Libraries

### 6.1. The PKCS #11 Library

CSP<sup>id</sup> includes a PKCS #11 library module named `CSPid.dll` (Windows) or `libcspid.so` (UNIX) for use by security-enabled applications that can access PKCS #11 compliant tokens.

On Windows, this library is installed in the CSP<sup>id</sup> program folder, by default `C:\Program Files\CSPid`, and is automatically registered in any Netscape-based profiles the next time the user logs in.

On UNIX, the default installation directory is `/opt/cspid`.

Manual configuration is normally required for non-Netscape-based applications and most Java applications. See Chapter 3 for more detailed information.

At this time the PKCS #11 library implements only RSA private key operations and publishes its capabilities via the `C_GetMechanismInfo()` function.

See “*CSP<sup>id</sup> API Developer's Guide, Version 1.1.0*” for complete details.

### 6.2. The Microsoft Windows Library

CSP<sup>id</sup> can integrate with any application that uses the native cryptographic framework on a supported Windows platform (*e.g.*, Internet Explorer, Outlook, Outlook Express) through their CryptoAPI (CAPI) or “Cryptography API: Next Generation” (CNG) interfaces. This is accomplished by installing a virtual smart card reader, the Microsoft Base Smart Card Provider Cryptographic Service Provider, and a Smart Card Minidriver adhering to Microsoft specifications, and then registering CSP<sup>id</sup> as the system's default CSP.

When configuring Windows applications to use CSP<sup>id</sup> for PKI enrollment purposes, users and administrators must select the *Microsoft Base Smart Card Crypto Provider* to ensure that all keys are generated and stored by CSP<sup>id</sup>.

## 7. Quality Assurance Issues

### 7.1. Testing Methodology

On Windows, CSP<sup>id</sup> was tested against the following system and applications software, to ensure that it correctly performed the indicated tasks with all applications:

Software	Tasks Executed
Windows XP SP 2	Certificate enrollment
Internet Explorer 6	Client-authenticated SSL/TLS
Microsoft Outlook 2003	S/MIME sign/decrypt
Firefox 2.0	Memory scanning for sensitive material
Thunderbird 1.5	
JRE 1.5.0_09	
Netscape 4.79 w/PSM 1.4	

**Table 13: Test Configurations**

See the *CSP<sup>id</sup> Test Plan* document for detailed information on the range of tests that were performed.

## **CSP<sup>id</sup> User's Guide**

### **7.2. Known Issues**

#### **7.2.1. Windows**

- Uninstalling CSP<sup>id</sup> may leave links in CAPI between certificates and the removed CSP<sup>id</sup>.dll. If the user attempts to use a private key associated with one of these certificates, they will either be shown an error message or be prompted to insert a smart card containing the associated private key. Reinstalling CSP<sup>id</sup> and “clearing” it using the command line management utility will resolve this issue. Alternatively, the user can just delete the affected certificates from the CAPI store using Internet Explorer.

#### **7.2.2. Netscape 4.x or higher w/o PSM**

- Currently none.

#### **7.2.3. Netscape 4.75 or higher w/PSM 1.4**

- Importing a certificate (as part of a PKI enrollment process) and then immediately using the personal security manager (PSM) to view available certificates causes it to crash, requiring the user to restart Netscape.

#### **7.2.4. Internet Explorer**

- Currently none.

#### **7.2.5. Microsoft Outlook**

- Currently none.

## 8. Net-Centric Applications

CSP<sup>id</sup> can be deployed in conjunction with DAS and your company's existing security-enabled applications (*e.g.*, Microsoft Outlook S/MIME) to support enhanced security protocols, such as role-based signing and decryption that were previously impossible to implement with conventional PKI-based tools. In fact, the combination of CSP<sup>id</sup> and DAS allows you to leverage security tools based on today's standards to provide functionality that some vendors would have you believe can only be obtained using more recent schemes such as identity-based encryption (IBE) for which standards have not yet been established.

The following scenarios illustrate four of the possible applications of CSP<sup>id</sup>/DAS in a 'net-centric' environment:

### 8.1. Role-Based Authentication

**PROBLEM:** A group of *watch officers* are to sign messages that recipients can validate as having been issued by some authorized group member. (In this scenario, recipients don't care which individual signed a given message, they only need assurance that an authorized member of the group did so.)

**SOLUTION SETUP:** An asymmetric key pair is generated for the watch officer role, a special role certificate is issued on the public component, and the role private key is put under the control of a DAS server that is configured to perform signing operations with that key only for individuals on the active duty roster for this role. The role certificate is loaded into the CSP<sup>id</sup> clients on all watch officer systems and CSP<sup>id</sup> is registered with all security-enabled client applications on those systems. (In fact, all watch officer logins may be hosted on a single system.)

**OPERATION:** When a watch officer attempts to sign an outgoing message using the role credentials with any security-enabled client application on his system, his CSP<sup>id</sup> client automatically establishes a TLS-secured connection (with client authentication) to the appropriate DAS server where the signature computation is performed using its protected role private key. As usual, recipients use the watch officer role certificate to validate all signed messages.

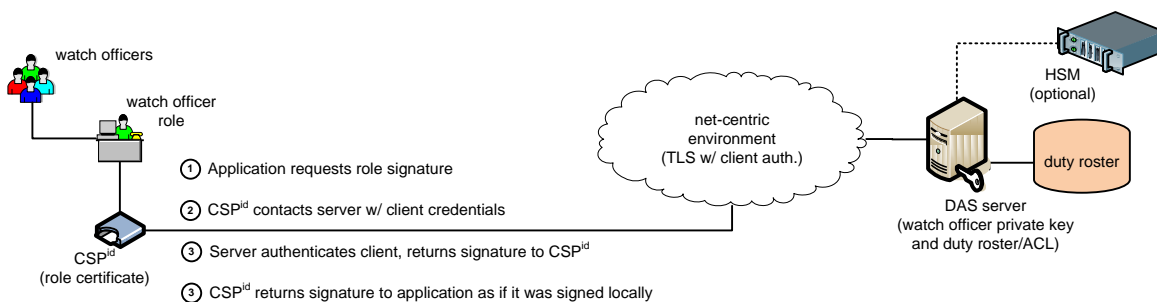


Figure 17: Watch Officers Sign Messages with Shared Role Key on DAS Server

**RESULT:** Recipients can be assured that an authorized watch officer issued each validly-signed message while the DAS server's audit trail keeps track of the identities of the individuals who actually performed

## CSP<sup>id</sup> User's Guide

each signing operation. Since the watch officer private key is securely protected by the DAS server (possibly on an independent HSM), at no time is it available for compromise even by authorized signers.

### 8.2. Role-Based Encryption

**PROBLEM:** Documents and e-mail messages encrypted for a particular commander (*e.g.*, CMDR CENTCOM) must be available (without re-keying) to his successor in that role.

**SOLUTION SETUP:** An asymmetric key pair is generated for the commander, a special role certificate is issued on the public component, and the private component is put under the control of a DAS server that is configured to perform decrypt operations with that key only for the active commander. The certificate is loaded into the CSP<sup>id</sup> client on the commander's system and CSP<sup>id</sup> is registered with all security-enabled client applications. All documents and e-mail messages are encrypted for the commander using the role certificate.

**OPERATION:** When the commander attempts to decrypt a document or e-mail message encrypted with the role certificate, the CSP<sup>id</sup> client on his system automatically establishes a TLS-secured connection (with client authentication) to the appropriate DAS server where the required asymmetric key unwrapping operation is performed using its protected private key. When the commander is replaced, an administrator need only update the access control list (ACL) for the corresponding DAS account.

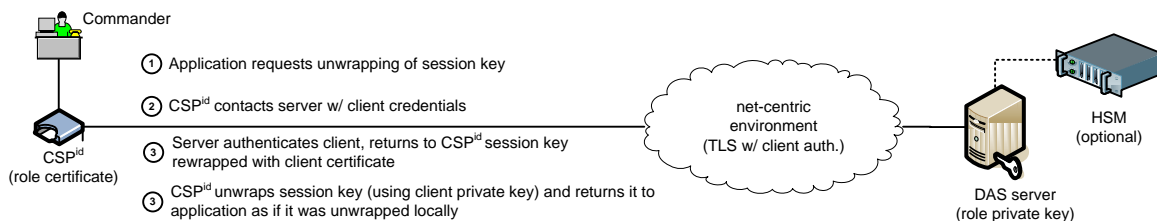


Figure 18: Commander Decrypts Documents Using Role Key on DAS Server

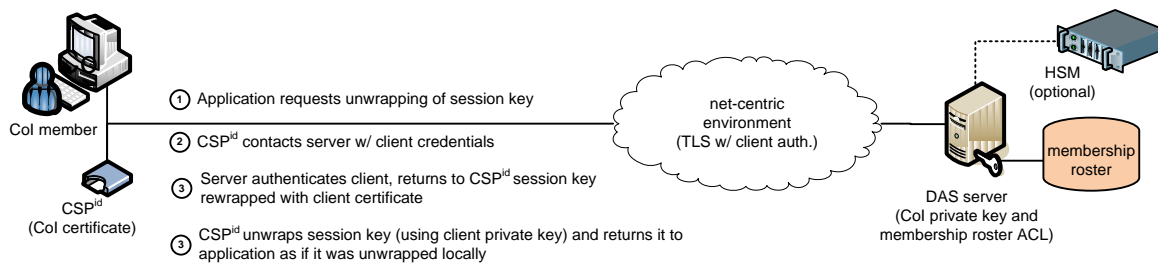
**RESULT:** All sensitive documents and e-mail messages remain encrypted under a single certificate and can only be decrypted by the current commander after strong authentication using his current individual credentials (*e.g.*, **CAC card**). No re-keying is necessary as different individuals inherit the commander role and the critical private key is securely stored on a DAS server or, even more securely, on an attached HSM. The DAS server's audit trail provides a centralized record of all decrypt operations and, optionally, of the true identity of the individual who performed them.

### 8.3. Confidentiality within a Community of Interest

**PROBLEM:** Sensitive documents must be security shared among the members of a community of interest (CoI) with a dynamic membership roster.

**SOLUTION SETUP:** An asymmetric key pair is generated for the CoI, a special group certificate is issued on the public component, and the private component is put under the control of a DAS server that is configured to perform decrypt operations with that key only for active CoI members. The certificate is loaded into the CSP<sup>id</sup> client on the systems of all CoI members, and CSP<sup>id</sup> is registered with all security-enabled client applications on those systems. All sensitive documents and e-mail messages intended for the CoI are encrypted under the group certificate.

**OPERATION:** When a CoI member attempts to decrypt a document or e-mail message encrypted with the group certificate, the CSP<sup>id</sup> client on his system automatically establishes a TLS-secured connection (with client authentication) to the appropriate DAS server where the required asymmetric key unwrapping operation is performed using its protected private key. When the CoI membership roster changes, an administrator need only update the access control list (ACL) for the corresponding DAS account.



**Figure 19: CoI Members Decrypt Documents Using CoI Key on DAS Server**

**RESULT:** All sensitive documents and e-mail messages remain encrypted under a single certificate and can only be decrypted by active CoI members after strong authentication using their current individual credentials. No re-keying is necessary as changes are made to the CoI membership roster. The critical private key is securely stored on a DAS server or, even more securely, on an attached HSM. The DAS server's audit trail provides a centralized record of all decrypt operations and, optionally, of the identities of the individual CoI members who performed them.

#### **8.4. Expanded Storage and Enhanced Security for Private Keys**

**PROBLEM:** Key rollover results in a new certificate and private key being loaded onto a high-ranking officer's smartcard where limited storage causes an older certificate and private key to be displaced. Suddenly the officer has lost the ability to decrypt documents and e-mail messages encrypted with the older credentials.

**SOLUTION SETUP:** Load the officer's entire key history into an account on a DAS server and install his certificate history into a CSP<sup>id</sup> client on his own system.

**OPERATION:** Once CSP<sup>id</sup> has been registered with all security-enabled applications on the officer's system, they will use his smartcard for cryptographic operations requiring his latest private key but transparently establish a TLS-secured CSP<sup>id</sup>/DAS connection when access to an older private key is required. Going forward, key rollover simply involves moving the officer's retiring certificate to CSP<sup>id</sup> and the corresponding key pair to his account on the DAS server.

## CSP<sup>id</sup> User's Guide

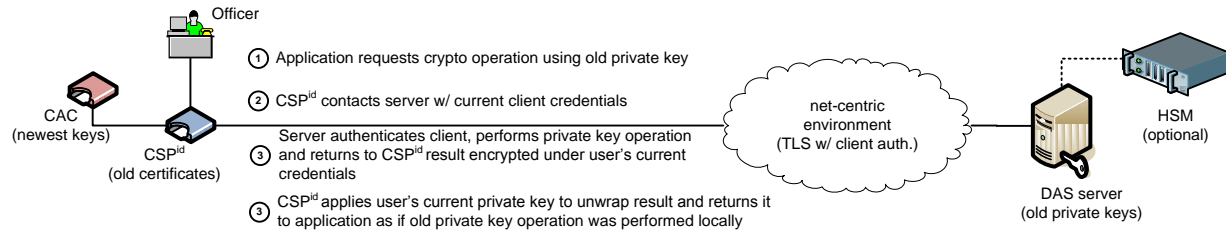


Figure 20: Officer Retains Access to His Entire Key History

**RESULT:** By relying on the virtually unlimited storage of the DAS server (and the potentially greater security of its key store which may be located on an HSM), an (on-line) user can maintain use of his entire key history. He thereby retains access to all encrypted documents and e-mail messages ever sent to him in a completely seamless manner.

### 8.5. Brokered Authentication - "Need-to-Know" Control Over Sensitive Resources

**PROBLEM:** Access to a sensitive resource needs to be controlled with a set of finely-tuned access control rules that might vary over time.

**SOLUTION SETUP:** Generate a "resource key pair," obtain a "resource certificate" on the public key, and distribute it to all users who might require access to that resource; ensure that the certificate is loaded into the CSP<sup>id</sup> client on each user's system (this step may be automated in several ways). Create a DAS account for the resource and install its certificate and private key along with a custom "authenticator" that implements the access control rules. (An authenticator is a simple Java function that implements the access control predicate: it takes as input the user's credentials and returns a Boolean response indicating whether the user should be allowed or denied access to the requested resource. Sample predicates based on LDAP group membership, certificate extension and/or attribute values, etc., can be provided.)

**OPERATION:** When an application attempts to access the sensitive resource, CSP<sup>id</sup> will be asked to provide authentication (which normally takes the form of a "proof of possession" of the resource private key — typically a digital signature). CSP<sup>id</sup> will then attempt to obtain this signature from the appropriate DAS server as illustrated in the following diagram:

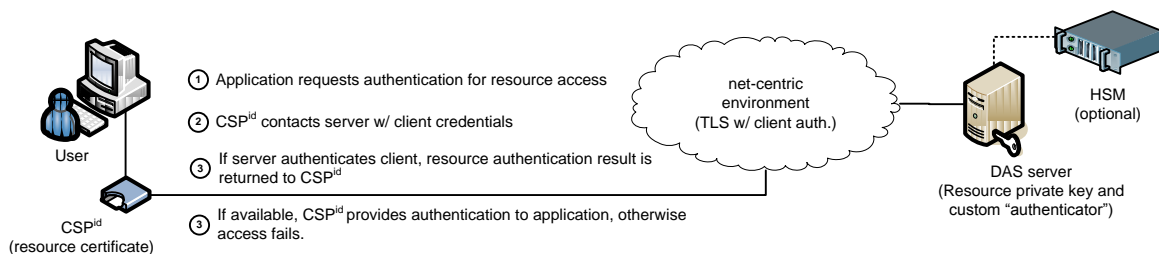


Figure 21: A DAS Server Used to Impose Strong "Need-to-Know" Controls over Sensitive Resources

Only if the user's credentials pass the custom authenticator's test will the DAS server provide the required signature for resource access.

RESULT: Together, CSP<sup>id</sup> and DAS provide assurance that the sensitive resource is securely protected. Maintenance of the entire system is simplified by reliance on a custom "authenticator" that may be easily modified whenever access control policies change. A single DAS server can be used to protect an unlimited number of sensitive resources in this way.

## **9. Future Development**

CSP<sup>id</sup> is a rather new application and ISC is committed to improving it based on customer feedback.

### **9.1. Future Releases**

Some of the improvements and additional development tasks that ISC is currently considering include:

- porting CSP<sup>id</sup> to Mac OS X (the CSP<sup>id</sup> 2.x series is available for OS X on Intel-based systems but lacks integration with the native “keychain”).
- support for single sign-on: once the user starts and logs in to the CSP<sup>id</sup> management tool they will not need to enter their CSP<sup>id</sup> password in other applications (Microsoft has announced that a future version of their Base Smart Card CSP will permit us to provide this feature for Windows Vista and above).
- ECC support (the CSP<sup>id</sup> 2.x series includes experimental ECC support).

## 10. References

- CSP<sup>id</sup> API      *CSP<sup>id</sup> API Developer's Guide*, v 2.1.0, Information Security Corp., April 2010.
- CSP<sup>id</sup> Test Plan      *CSP<sup>id</sup> Test Plan*, v 2.1.0, Information Security Corp., April 2010.
- FIPS 46-3      *FIPS 46-3: Data Encryption Standard (DES)*, NIST, October 25, 1999.  
<http://csrc.nist.gov/publications/fips/index.html>
- FIPS 180-2      *FIPS 180-2: Secure Hash Standard*, NIST., August 1, 2002.  
<http://csrc.nist.gov/publications/fips/index.html>
- FIPS 197      *FIPS 197: Advanced Encryption Standard (AES)*, NIST, November 26, 2001.  
<http://csrc.nist.gov/publications/fips/index.html>
- Java PKCS#11      *Java™ PKCS#11 Reference Guide*, Sun Microsystems, May 2004.  
<http://java.sun.com/j2se/1.5.0/docs/guide/security/p11guide.html>
- Java SSL/TLS      *Java™ Secure Socket Extension (JSSE) Reference Guide*, Sun Microsystems, 2004.  
<http://java.sun.com/j2se/1.5.0/docs/guide/security/jsse/JSSERefGuide.html>
- MODUTIL      *Using the Security Module Database Tool (modutil)*, Mozilla.org  
<http://www.mozilla.org/projects/security/pki/nss/tools/modutil.html>
- MS SCMD      *Smart Card Minidriver Specification for Windows Base Cryptographic Service Provider (Base CSP) and Smart Card Key Storage Provider (KDP) v 5.05*, Microsoft Corp., October 2006.  
<http://www.microsoft.com/whdc/device/input/smartcard/sc-minidriver.msp>
- NIST AES Key Wrap      *AES Key Wrap Specification*, NIST, November 2001.  
<http://csrc.nist.gov/encryption/kms/key-wrap.pdf>
- PCSC      Basko, Dmitry, *Development of PC/SC compatible driver for WINDOWS (MS VC++.NET)*, 2003. [http://www.bds.dogma.net/pc\\_sc.htm](http://www.bds.dogma.net/pc_sc.htm)
- PKCS #5      *Password-Based Encryption Standard*. v2.0, RSA Laboratories, March 25, 1999.  
<http://www.rsasecurity.com/rsalabs/node.asp?id=2127>
- PKCS #8      *Private-Key Information Syntax Standard*. v1.2, RSA Laboratories, November 1993. <http://www.rsasecurity.com/rsalabs/node.asp?id=2130>
- PKCS #11      *Cryptographic Token Interface Standard*. v2.20, RSA Laboratories, June 2004.  
<http://www.rsasecurity.com/rsalabs/node.asp?id=2133>
- PKCS #12      *Personal Information Exchange Syntax Standard*. v1.0, RSA Laboratories, June 1999. <http://www.rsasecurity.com/rsalabs/node.asp?id=2138>
- PKCS #15      *Cryptographic Token Information Syntax Standard* v1.1, RSA Laboratories, June 2000. <http://www.rsasecurity.com/rsalabs/node.asp?id=2141>
- RFC 3211      Gutmann, P., *RFC 3211: Password-based Encryption for CMS*, University of Auckland, December 2001. <http://tools.ietf.org/html/rfc3211>

## CSP<sup>id</sup> User's Guide

- RFC 3394      Schaad, J., and R. Housley, *RFC 3394: Advanced Encryption Standard (AES) Key Wrap Algorithm*, Soaring Hawk Consulting and RSA Laboratories, September 2002. <http://tools.ietf.org/html/rfc3394>
- RFC 3565      Schaad, J., *RFC 3565: Use of the Advanced Encryption Standard (AES) Encryption Algorithm in Cryptographic Message Syntax (CMS)*, Soaring Hawk Consulting, July 2003. <http://tools.ietf.org/html/rfc3565>
- RFC 3852      Housley, R., *RFC 3852: Cryptographic Message Syntax (CMS)*, Vigil Security, July 2004. <http://tools.ietf.org/html/rfc3852>
- RFC 4231      Nystrom, M., *RFC 4231: Identifiers and Test Vectors for HMAC-SHA-224, HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512*, RSA Security, December 2005. <http://tools.ietf.org/html/rfc4231>
- Tomcat        *The Apache Tomcat 5.5 Servlet/JSP Container SSL Configuration HOW-TO*, The Apache Software Foundation, 1999-2006. <http://tomcat.apache.org/tomcat-5.5-doc/ssl-howto.html>
- X.680         *Information Technology – Abstract Syntax Notation One (ASN.1): Specification of Basic Notation*, ITU-T, July 2002.
- X.690         *Information Technology – ASN.1 Encoding Rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER), and Distinguished Encoding Rules (DER)*, ITU-T, July 2002.

## 11. Appendix A: Default Configuration File

The default configuration file as provided to administrators for modification prior to installation.

```
#####
# CSPID SAMPLE CONFIGURATION FILE
# Version 2.1.0
# Copyright (c) 2007-10 Information Security Corp. All rights reserved.
#
#####
#
#   Edit this file to configure how the CSPid token functions.
#
#   Windows:
#       This file must be located in the same folder as the CSPid
#       library (CSPid.dll or libcspid.so).
#
#   UNIX:
#       The CSPid library searches for this file in this order
#       a. /etc/cspid.cfg
#       b. /usr/etc/cspid.cfg
#       c. /opt/cspid/cspid.cfg
#       d. $CSPidInstDir/cspid where CSPidInstDir is an environment
#          variable specifying the location of the configuration file.
#
#   File Format:
#       a. Any line beginning with # or ; is considered a comment.
#       b. Blank lines are ignored.
#       c. All other lines should be Option=Value pair.
#       d. Where noted, you may continue an option on the next line by
#          using nothing for the Option (i.e. simply "=Value")
#
#####

#####
# CSP SETTINGS
#####
#
#   P15URL specifies the path and filename to use for the PKCS#15 PDU.
#;P15URL=%APPDATA%\CSPid\cspid.pl5
#####

#####
# LOG SETTINGS
#####
#
#   LOGLEVEL specifies an integer representing the logging level to use.
#       0 (critical information only)
#       1 (information)
#       2 (debug)
#   Debug and information entries appear only in the file specified by LOGFILE.
#   Critical information entries are always logged to the system event log, and
#   optionally to the file specified by LOGFILE.
#;LOGLEVEL=0
#
#   LOGFILE specifies path and filename in which to log information in
#   addition to the system event log.
#;LOGFILE=
#
#   APPLOGFILE is the location where CSPid stores the list of
#   applications that have called the C_initialize function.
#;APPLOGFILE=%APPDATA%\CSPid\CSPidApps.log
#####
```

```
#####
# PASSWORD SETTINGS
#####
#
# PWREQNUM specifies whether or not user created passwords must include
# a number.
# 0 (no)
# 1 (yes, one of [0-9] must be included)
;PWREQNUM=0
#
# PWREQALPHA specifies whether or not user created passwords must
# include an alpha character.
# 0 (no)
# 1 (yes, one of [a-z][A-Z] must be included)
;PWREQALPHA=0
#
# PWREQPUNC specifies whether or not user created passwords must
# included a punctation mark.
# 0 (no)
# 1 (yes, one of [.,?!;:'"] must be included)
;PWREQPUNC=0
#
# PWREQSPEC specifies whether or not user created passwords must
# include a special character.
# 0 (no)
# 1 (yes, one of [!@#$%^&*()+=~/><|\|[]{}~`] must be included)
;PWREQSPEC=0
#
# PWREQMIXEDCASE specifies whether or not user created passwords
# must include both lower and upper case characters
# 0 (no)
# 1 (yes)
;PWREQMIXEDCASE=0
#
# PWMINLEN specifies the minimum length of user created passwords.
;PWMINLEN=1
#
# PWCHANGE specifies the time between required user password changes.
# During the 10 days prior to mandatory password change the user will
# be offered the chance to changed their password when the CSPid
# Manager starts.
;PWCHANGE=0
#
# PWHISTORY specifies the number of unique passwords required before
# a user can re-use a previous password.
;PWHISTORY=0
#
# PWTIMEOUT specifies the password timeout value in seconds. After
# X seconds of inactivity the user will be required to reenter their
# CSPid password. Please see the CSPid user's guide for complete
# implementation details and guidance.
# -1 = never
# nonzero = seconds of inactivity afterwhich PIN reentry is required
;PWTIMEOUT=-1
#
# RSTPWAGENTS specifies one or more password reset agent certificates
# as a base64-encoded PKCS#7 PDU. The cspid_cli utility's --p7tob64
# option will convert a PKCS#7 generated by another application into
# the form expected by this configuration file. The basic rules are:
# a. No line can exceed 500 characters in length
# b. Continuation lines must begin with a = character
#
;RSTPWAGENTS=First line of PDU
;=Second line of PDU
;=Third line of PDU
#
# RSTPWMINLEN specifies the number of random bytes to generate when
# creating the reset password. The reset password the user must enter
```

```

# will be 1/3 bigger than this value. The default value is 10, the
# minimum value is 8. This value should be at least as big as
# the PWINLEN value.
;RSTPWINLEN=10
#####

#####
# NETSCAPE/MOZILLA SETTINGS
#####
#
# NSSEARCHPATH specifies one or more semi-colon delimited paths
# that CSPid should search when finding Netscape-based application
# profiles during installation or when the user selects Register with
# Applications. A common addition is %ProgramFiles%\Netscape
#
;NSSEARCHPATH=%APPDATA%\Mozilla;%APPDATA%\Thunderbird
#
# NSCREATE TRUST specifies whether or not to CKO_NETSCAPE_TRUST
# objects are created for certificates stored by CSPid.
# 0 (no)
# 1 (yes)
# Setting this value to 1 will cause most Netscape-based applications
# to trust any certificate stored by CSPid (including roots).
;NSCREATE TRUST=1
#####

#####
# IMPORT/EXPORT SETTINGS
#####
#
# CAIMPORT specifies whether or not CSPid should store issuer
# certificates.
# 0 (Never import CA certificates via the CSPid User Interface.
# Effectively disables the import of anything but PKCS#12 files.)
# 1 (Allow users to import CA certificates via the CSPid user
# interface, but prompt for approval of root certificates.)
# 2 (Allow users to import CA certificates via the CSPid user
# interface without any prompting (CAPI prompts *may* appear).)
# Note, this setting affects PKCS#12 import and export with regard to
# prompting only.
;CAIMPORT=1
#
# P12EXPORT specifies whether or not users can export their
# private keys from CSPid as PKCS#12 formatted files.
# 0 (no)
# 1 (yes)
;P12EXPORT=1
#
# P12IMPORTCA specifies whether or not CSPid should import issuer
# certificates it finds when importing PKCS#12 files.
# 0 (no)
# 1 (yes)
# Note, setting this value to 1 causes the CSPid user interface to
# import any issuer certificates it finds in the PKCS#12 during import
# if allowed by the CA Import Settings. During export, if this value is
# set to 1, issuing certificates will be included in the exported
# PKCS#12 if available.
;P12IMPORTCA=1
#
# P12IMPORTREGAPPS specifies whether or not the CSPid UI should run
# the register with applications event when a PKCS#12 file is imported
# from the CSPid UI's command line.
# 0 (no)
# non-zero = how recently the certificate must have been issued in hours
# (i.e. 24 means if the imported certificate was issued within
# the last 24 hours run the event, otherwise do not).

```

## CSPid User's Guide

```
;P12IMPORTREGAPPS=0
#
# P12CLIIMPDELOMATCH specifies the string to search for when importing
# a PKCS#12 file using the CSPid UI's command line. If the string is
# found in the path or name of the imported file it will be deleted
# after it has been successfully imported. Common values are Temp (for
# Firefox) and Temporary (for Internet Explorer).
;P12CLIIMPDELOMATCH=
#####

#####
# EVENT HANDLING
#####
#
# ENROLLURL specifies a URL that should be opened by the CSPid Manager
# when it starts and no certificates with private keys can be found.
# This value is superceded by the ENROLLCMD option if present.
;ENROLLURL=http://www.infoseccorp.com/ca/silver/contents.htm
#
# ENROLLURL_MESSAGE specifies the text to display to the user prior to
# opening the browser to the enrollment URL specified in ENROLLURL.
;ENROLLURL_MESSAGE=
#
# ENROLLCMD specifies one or more commands (see note below) that should
# be executed when the CSPid Manager starts and no certificates with
# private keys can be found. This value supercedes the ENROLLURL
# option.
;ENROLLCMD=
#
# ENROLLCAPI causes CSPid to move all credentials that can be exported
# from the user's Microsoft CAPI "Personal" store to CSPid when the
# CSPid manager starts and no credentials are found. This value
# supercedes both the ENROLLURL and ENROLLCMD values. The currently
# selected EFS certificate and any non-exportable certificates are not
# moved.
# 0 (disabled)
# 1 (enabled)
;ENROLLCAPI=0
#
# ENROLLCAPI_MESSAGE specifies the text to display to the user prior to
# migrating their keys from CAPI/CNG on initial startup.
;ENROLLCAPI_MESSAGE=
#
# NEWSIGCERTCMD specifies one or more commands (see note below) that
# should be executed when a new signing certificate is stored in CSPid.
# The following must be true for these commands to be executed:
# a. The certificate must have a matching private key already stored
# by CSPid.
# b. The program creating the loading the certificate object into
# CSPid cannot be the CSPid Manager or the CSPid Command Line.
# Therefore, these are most useful when you wish to execute one or more
# commands during web enrollment. For example, you can use these
# options with ISC's Credential Management Utility to download and
# install a PKCS#12 file using a newly issued (and thus installed)
# signing certificate. For complete details please see the CSPid User's
# Guide.
;NEWSIGCERTCMD=
#
# NEWENCCERTCMD specifies one or more commands (see note below) that
# should be executed when a new encryption certificate is stored in
# CSPid. Please see NEWSIGCERTCMD for details.
;NEWENCCERTCMD=
#
# NEWDASSIGNCERTCMD specifies one or more commands (see note below) that
# should be executed when a new DAS-enabled signing certificate is stored
# in CSPid. The following must be true for these commands to be executed:
# a. DAS must be enabled (see below)
# b. The program creating the loading the certificate object into
```

```

#          CSPid cannot be the CSPid Manager or the CSPid Command Line.
;NEWDASSIGNCERTCMD=
#
#          NEWDASENCCERTCMD specifies one or more commands (see note below) that
#          should be executed when a new DAS-enabled encryption certificate is
#          stored in CSPid. Please see NEWDASSIGNCERTCMD for details.
;NEWDASENCCERTCMD=
#
#          RENEWURL specifies the URL to open in the default web browser
#          when the user selects Renew My Certificate in the CSPid Manager or
#          when the CSPid Manager determines that renewal is required based on
#          the RENEWDAYS option. This value is superceded by the RENEWCMD
#          option. The CSPid determines whether renewal is required as follows:
#          a. If there is at least one signing certificate and no signing
#             certificate will be valid in RENEWDAYS.
#          b. or if there is a least one encrypting certificate and no
#             encrypting certificate will be valid in RENEWDAYS.
;RENEWURL=http://www.infosecorp.com/ca/silver/contents.htm
#
#          RENEWURL_MESSAGE specifies the text to display to the user prior to
#          opening the browser to the enrollment URL specified in RENEWURL.
;RENEWURL_MESSAGE=
#
#          RENEWCMD specifies one or more commands (see note below) that should
#          be executed when the CSPid Manager starts and finds that either no
#          signing or no encrypting certificates will be valid in RENEWDAYS.
#          This value supercedes the RENEWURL option. See RENEWURL for more
#          information.
;RENEWCMD=
#
#          RENEWDAYS specifies the number of days before certificate expiration
#          that should be considered when deciding whether or not certificate
#          renewal should execute.
;RENEWDAYS=
#
#          REGAPPSCMD specifies one or more additional commands (see note below)
#          that should be executed when the user selects Register with
#          Applications in the CSPid Manager.
;REGAPPSCMD=
#
#          ALLCERTSISSUERS modifies the treatment of certificates installed that
#          are not associated with private keys. If enabled (1) all certificates
#          installed without an associated private key will be treated as issuer
#          certificates and installed in the Windows "CA" (Intermediate Certification
#          Authorities) store. If disabled (0) any certificate without a private
#          key that does not include a basic constraints extension identifying the
#          type will be installed in the Windows "Other" (other people) store.
;ALLCERTSISSUERS=0
#
#          STARTUPCMD specifies one or more commands that should be executed
#          when the user starts the CSPid Manager.
;STARTUPCMD=
#####

#####
# DAS OPTIONS
#####
#
#          DASENABLE turns on DAS support. If set to a CSPid DAS Enablement
#          serial number CSPid will assert to applications that it holds the
#          private key matching any DAS-enabled certificate imported into CSPid.
;DASENABLE=0
#
#          DASSERVER specifies one or more DAS server URIs that should be used
#          when performing DAS unwrap operations. Servers are tried in order
#          listed.
;DASSERVER=https://192.168.0.85:6444/das/cois/
#

```

## CSP<sup>id</sup> User's Guide

```
# DASSIGSERVER specifies one or more DAS server URLs that should be
# used when performing DAS signature operations. Servers are tried
# in order listed.
;DASSIGSERVER=https://192.168.0.85:6444/das/cois/
#
# DASTIMEOUT specifies the number of seconds to wait for a connection
# to a DAS server for a signature or unwrap request before aborting the
# attempting and trying the next server.
;DASTIMEOUT=5
#
# EXT_P11_LIB specifies the PKCS#11 library path and name to use when
# authenticating to a DAS server and unwrapping the server's response.
;EXT_P11_LIB=C:\windows\system32\acpkcs211.dll
#
# EXT_P11_LABEL specifies the PKCS#11 label to use when CSPid communicates
# with a PKCS#11 device. If present, CSPid will only use a slot if the
# token present in that slot matches this value. If blank, CSPid will use
# the first slot that contains a token.
;EXT_P11_LABEL=
#
# EXT_P11_NAME specifies the name to display in the password dialog
# when CSPid prompts the user for the PIN to their smart card. If
# blank it will use the card's label.
;EXT_P11_NAME=DOD CAC
#
# EXT_P11_SIGN_ISSUER_ID specifies either the Issuer DN or the AKI
# hash value that is preferred when auto-selecting a signing
# certificate.
;EXT_P11_SIGN_ISSUER_ID=
#
# EXT_P11_ENC_ISSUER_ID specifies either the Issuer DN or the AKI
# hash value that is preferred when auto-selecting an encryption
# certificate.
;EXT_P11_ENC_ISSUER_ID=
#
# EXT_P11_ENC_LABEL_ID specifies a keyword that the label of the
# preferred encryption certificate will contain. For CACs it's Encryption
;EXT_P11_ENC_LABEL_ID=
#
# EXT_P11_SIGN_LABEL_ID specifies a keyword that the label of the
# preferred signature certificate will contain. For CACs it's Signature
;EXT_P11_SIGN_LABEL_ID=
#####

#####
# INSTALLATION OPTIONS
#####
#
# SECURE specifies whether or not the installation program should
# install this configuration file in a tamper evident manner.
# 0 (no)
# 1 (yes)
# Note, if this is selected and the user tampers with the configuration
# file an error dialog will be displayed and the user's keys will be
# unavailable under the configuration file is restored to its original
# state.
;SECURE=0
#####

#####
# A NOTE ABOUT COMMANDS
#####
#
# Options ending in CMD, such as REGAPPSCMD, are handled as follows:
# a. If one or more options with the same value are provided in this
# configuration file each command will be executed when the
```

```

#           particular event occurs.
#   b. If multiple commands are present they are execute in the order
#       they appear in this file and each command must finish before
#       the next will be started.
#   c. On Windows the commands are launched in hidden windows.
#
# For example, to cause 2 additional commands to run when the user
# selects Register with Applications you would add 2 REGAPPSCMD= lines
# to this file like so:
#   REGAPPSCMD="C:\My Program\mp1.exe" -L "%TEMP%\logfile.txt"
#   REGAPPSCMD="C:\My Program\mp2.exe" -L "%TEMP%\logfile.txt"
#   When the event occurs mp1.exe will be executed followed by mp2.exe
#
#   Please see the User's Guide for complete details
#####

```

**Figure 22: A Sample CSP<sup>id</sup> Configuration File With Default Settings**

NOTE: On Windows, environment variables may be referenced in the configuration file as illustrated in the above example. On UNIX, replace %VAR% with the conventional \$VAR to dereference an environment variable named VAR. These variables are expanded prior to use.

## 12. Appendix B: The PKCS #15 Key Storage Format

CSP<sup>id</sup> maintains a key store consisting of a single PKCS #15 PDU containing certificates, private keys, public keys, and their associated attributes. This PDU is currently stored on the user's file system (local disk, removable media, or network drive). Future versions of CSP<sup>id</sup> may allow the key store PDU to be stored in alternate locations (such as in a network-accessible database).

### 12.1. PKCS #15

PKCS #15 specifies syntax for storage of objects, syntax for protecting the confidentiality of sensitive objects, and a method to ensure the integrity of these objects. A brief summary of those aspects of the standard used by CSP<sup>id</sup> is presented below. For more information we refer the reader to the PKCS #15 standards document.

### 12.2. Object Syntax

PKCS #15 specifies an ASN.1 encoding format for all objects. For software-tokens a single flat file containing an ASN.1 DER-encoded sequence of objects is generated. This object is defined as a `PKCS15Token`. It is made up of a list of key IDs and a list of objects. The key IDs uniquely label the encrypted objects. A sample PKCS #15 PDU is presented in *Appendix C: A Sample PKCS #15 Key Store*.

### 12.3. Confidentiality

PKCS #15 specifies the use of CMS as defined in RFC 3852 using passwords for confidentiality. RFC 3852 specifies that a key-encryption key (KEK) be derived from a password and used to encrypt the content-encryption key (CEK). The KEK is to be derived from a password according to PKCS #5 v2.0 using PBKDF2. CSP<sup>id</sup> uses a random salt and an iteration count of 2048 when performing PBKDF2.

CSP<sup>id</sup> deviates slightly from PKCS #15 and PKCS #5 by using AES-256 in CBC mode (rather than TDES) for encryption of private objects within the PKCS #15 PDU as specified by RFC 3565 and RFC 3394. RFC 3394 is based on the NIST AES Key Wrap specification. For compatibility with the only known third party implementation of PKCS #15, CSP<sup>id</sup> will also support the RFC 3211 mechanism for decryption only.

Private key objects are encrypted using the algorithms and formats described in the previous paragraph. The output produced is a CMS `EnvelopedData` structure that is included, with additional information, in the eventual `PKCS15Token` PDU that is saved in the active key store.

## 12.4. Integrity

PKCS #15 stipulates the use of CMS as defined in RFC 3852 for assuring the integrity of the PDU. Specifically, it requires the creation of a CMS enveloped `authenticatedData` PDU containing both the PDU and a computed MAC value over it. CSP<sup>id</sup> uses HMAC-SHA512 as defined in RFC 4231 with a random key for the MAC operation, and then wraps the random HMAC key using AES-256 in the same manner as for the CEK as described above (*i.e.*, with a key derived from the user's password according to PKCS #5 v2.0 and PBKDF2.)

## 12.5. Initialization

When first started, or when the active key store cannot be opened, the PKCS #11 library creates an empty PKCS #15 PDU and appropriate enveloped `authenticatedData` structure with a default password of "PASSWORD" unless otherwise specified in the configuration file (see section 3.4). When the CSP<sup>id</sup> management tool detects that the user has not previously set a password, it forces the user to do so. The user can subsequently use the included utilities to change their password.

### 13. Appendix C: A Sample PKCS #15 Key Store

This appendix contains a sample CSP<sup>id</sup> key store file. The original binary PKCS #15 PDU has been “pretty printed” using GUIDumpASN.1.

```

0000 30 96F: SEQUENCE {
0004 06 B:   OBJECT IDENTIFIER authData (1 2 840 113549 1 9 16 1 2)
0011 A0 95E:   [0] {
0015 30 95A:     SEQUENCE {
0019 02 1:       INTEGER 0
001C 31 5B:       SET {
001E A3 59:         [3] {
0020 02 1:           INTEGER 0
0023 A0 1B:           [0] {
0025 06 9:             OBJECT IDENTIFIER pkcs5PBKDF2 (1 2 840 113549 1 5 12)
0030 30 E:             SEQUENCE {
0032 04 8:               OCTET STRING
:                   FD D3 C8 2D E1 60 08 AC
003C 02 2:               INTEGER 2048
:                   }
:               }
:           }
0040 30 D:           SEQUENCE {
0042 06 9:             OBJECT IDENTIFIER '2 16 840 1 101 3 4 1 45'
004D 05 0:             NULL
:                   }
004F 04 28:           OCTET STRING
:                   1A 2A 61 A5 79 09 15 0E 72 14 D5 9A BB 31 1B 1D
:                   9D B0 C8 B2 3A 85 A4 84 70 60 FC B8 D4 EE 43 54
:                   ED 5A FC C9 56 F8 8E 91
:                   }
:           }
0079 30 C:           SEQUENCE {
007B 06 8:             OBJECT IDENTIFIER '1 2 840 113549 2 11'
0085 05 0:             NULL
:                   }
0087 A1 D:           [1] {
0089 06 9:             OBJECT IDENTIFIER sha-512 (2 16 840 1 101 3 4 2 3)
0094 05 0:             NULL
:                   }
0096 30 897:          SEQUENCE {
009A 06 A:             OBJECT IDENTIFIER pkcs15content (1 2 840 113549 1 15 3 1)
00A6 A0 887:          [0] {
00AA 30 883:            SEQUENCE {
00AE 06 A:              OBJECT IDENTIFIER
:                  pkcs15content (1 2 840 113549 1 15 3 1)
00BA A0 873:            [0] {
00BE 30 86F:              SEQUENCE {
00C2 02 1:                INTEGER 0
00C5 30 868:              SEQUENCE {
00C9 A0 36C:                [0] {
00CD A0 368:                [0] {
00D1 30 364:                SEQUENCE {
00D5 30 2E:                SEQUENCE {
00D7 0C 28:                UTF8String
:                  '90c72555f6a105c071ec670be04120b4350831a3'
0101 03 2:                BIT STRING 7 unused bits
:                  '1'B (bit 0)
:                }
0105 30 21:                SEQUENCE {

```

```

0107 04 14:          OCTET STRING
                   :
                   : 90 C7 25 55 F6 A1 05 C0 71 EC 67 0B E0 41 20 B4
                   : 35 08 31 A3
011D 03 2:          BIT STRING 5 unused bits
                   : '110'B
0121 01 1:          BOOLEAN FALSE
0124 03 2:          BIT STRING 4 unused bits
                   : '1001'B
                   :
                   : }
0128 A1 30D:        [1] {
012C 30 309:          SEQUENCE {
0130 A2 302:          [2] {
0134 02 1:            INTEGER 2
0137 31 5B:          SET {
0139 A3 59:          [3] {
013B 02 1:            INTEGER 0
013E A0 1B:          [0] {
0140 06 9:            OBJECT IDENTIFIER
                   : pkcs5PBKDF2 (1 2 840 113549
1 5 12)
014B 30 E:          SEQUENCE {
014D 04 8:            OCTET STRING
                   : 46 B7 36 0C F7 84 7B C7
0157 02 2:            INTEGER 2048
                   :
                   : }
                   :
015B 30 D:          SEQUENCE {
015D 06 9:            OBJECT IDENTIFIER '2 16 840 1
101 3 4 1 45'
0168 05 0:          NULL
                   :
                   : }
016A 04 28:          OCTET STRING
                   : EF CC 57 56 44 9A 51 AD FC 4F 6B E6 2D 34 69 0A
                   : A9 28 D6 F0 D8 70 CD 99 12 90 4A 61 43 D3 63 ED
                   : 08 37 91 26 00 40 E5 89
                   :
                   : }
                   :
0194 30 29E:        SEQUENCE {
0198 06 9:            OBJECT IDENTIFIER
                   : data (1 2 840 113549 1 7 1)
01A3 30 1D:          SEQUENCE {
01A5 06 9:            OBJECT IDENTIFIER
                   : aes256-CBC (2 16 840 1 101 3
4 1 42)
01B0 04 10:          OCTET STRING
                   : E3 A4 4E 34 FB 0F AF A1 43 92 FD D6 B6 FA 2E B9
                   :
                   : }
01C2 80 270:        [0]
                   : 0E E4 36 24 12 77 55 BE 0E 24 CA F1 B2 7C 6F AE
                   : 96 1B B6 F1 1B 6E 71 F1 CD 97 D9 F1 4B E1 D5 30
                   : E6 B7 98 9A C8 7F 4E 17 15 5C CF F3 C7 39 D3 2B
                   : 62 AE B0 BA 31 72 E4 FC 6C 3A 69 AF BB 9B 43 54
                   : 54 CB F1 01 F2 EA 4E 87 BB DB BD D7 E8 7F 38 AF
                   : F3 54 1D A4 6F 47 07 D1 D9 75 64 A5 EF D4 66 10
                   : 6A EC 51 41 E8 3D 02 3B 8E F6 B7 7D 08 28 03 72
                   : 09 65 30 B3 38 91 D2 4C 70 5E 57 3C 21 38 40 DC
                   : [ Another 496 bytes skipped ]
                   :
                   : }
                   :
0436 02 1:          INTEGER 0
                   :
                   : }
                   :
                   : }

```



```

:                               35 08 31 A3
:                               }
0592 A1 39B:                   [1] {
0596 30 397:                   SEQUENCE {
059A 30 393:                   SEQUENCE {
059E 30 27B:                   SEQUENCE {
05A2 A0 3:                     [0] {
05A4 02 1:                     INTEGER 2
:                               }
05A7 02 14:                   INTEGER
:                               08 16 D8 EA 0C 26 68 6B 7F 5F F6 9D D6 52 F0 B2
:                               58 A4 74 EB
05BD 30 D:                     SEQUENCE {
05BF 06 9:                     OBJECT IDENTIFIER
:                               sha1withRSAEncryption (1 2
840 113549 1 1 5)
05CA 05 0:                     NULL
:                               }
05CC 30 B2:                   SEQUENCE {
05CF 31 B:                     SET {
05D1 30 9:                     SEQUENCE {
05D3 06 3:                     OBJECT IDENTIFIER
:                               countryName (2 5 4 6)
05D8 13 2:                     PrintableString 'US'
:                               }
:                               }
05DC 31 B:                     SET {
05DE 30 9:                     SEQUENCE {
05E0 06 3:                     OBJECT IDENTIFIER
:                               stateOrProvinceName (2 5
4 8)
05E5 13 2:                     PrintableString 'IL'
:                               }
:                               }
05E9 31 23:                   SET {
05EB 30 21:                   SEQUENCE {
05ED 06 3:                     OBJECT IDENTIFIER
:                               organizationName (2 5 4
10)
05F2 13 1A:                   PrintableString
'Information Security Corp.'
:                               }
:                               }
060E 31 11:                   SET {
0610 30 F:                     SEQUENCE {
0612 06 3:                     OBJECT IDENTIFIER
:                               localityName (2 5 4 7)
0617 13 8:                     PrintableString 'Oak Park'
:                               }
:                               }
0621 31 21:                   SET {
0623 30 1F:                   SEQUENCE {
0625 06 3:                     OBJECT IDENTIFIER
:                               organizationalUnitName (2
5 4 11)
062A 13 18:                   PrintableString 'Research
and Development'
:                               }
:                               }
0644 31 14:                   SET {
0646 30 12:                   SEQUENCE {
0648 06 3:                     OBJECT IDENTIFIER
:                               commonName (2 5 4 3)

```

## CSP<sup>id</sup> User's Guide

```

064D 13   B: PrintableString 'ISC Test
CA'
      :
      :
065A 31 25:
065C 30 23:
065E 06   9:
      :
113549 1 9 1)
0669 16 16: IA5String
'testca@infoseccorp.com'
      :
      :
      :
0681 30 1E: SEQUENCE {
0683 17   D:   UTCTime '061109000000Z'
0692 17   D:   UTCTime '071109000000Z'
      :
06A1 30  B5: SEQUENCE {
06A4 31   B:   SET {
06A6 30   9:     SEQUENCE {
06A8 06   3:     OBJECT IDENTIFIER
      :           countryName (2 5 4 6)
06AD 0C   2:     UTF8String 'US'
      :           }
      :     }
06B1 31 29: SET {
06B3 30 27:   SEQUENCE {
06B5 06   3:     OBJECT IDENTIFIER
      :           organizationName (2 5 4
10)
06BA 0C 20:   UTF8String 'Information
Security Corporation'
      :
      :
06DC 31 1E: SET {
06DE 30 1C:   SEQUENCE {
06E0 06   3:     OBJECT IDENTIFIER
      :           organizationalUnitName (2
5 4 11)
06E5 0C 15:   UTF8String 'R&D (no
assurance CA)'
      :
      :
06FC 31 12: SET {
06FE 30 10:   SEQUENCE {
0700 06   3:     OBJECT IDENTIFIER
      :           organizationalUnitName (2
5 4 11)
0705 0C   9:     UTF8String 'Test CA 1'
      :           }
      :     }
0710 31 18: SET {
0712 30 16:   SEQUENCE {
0714 06   3:     OBJECT IDENTIFIER
      :           commonName (2 5 4 3)
0719 0C   F:     UTF8String 'Test P12
Import'
      :
      :
072A 31 2D: SET {
072C 30 2B:   SEQUENCE {
072E 06   9:     OBJECT IDENTIFIER

```

```

:                               emailAddress (1 2 840
113549 1 9 1)
0739 16 1E:                     IA5String
'tech@infoseccorp.com'
:                               }
:                               }
:                               }
0759 30 9F:                     SEQUENCE {
075C 30 D:                       SEQUENCE {
075E 06 9:                       OBJECT IDENTIFIER
:                               rsaEncryption (1 2 840
113549 1 1 1)
0769 05 0:                       NULL
:                               }
076B 03 8D:                     BIT STRING 0 unused bits,
encapsulates {
076F 30 89:                     SEQUENCE {
0772 02 81:                     INTEGER
:                               00 CB 62 73 50 C9 5D 79 71 64 E9 D1 87 07 D3 88
:                               AB D6 A1 57 4D CF 34 C2 B8 04 E2 F4 3E EA 71 B4
:                               1E 47 9F F3 3E 6C AD A6 9E 26 E4 F4 08 52 A1 E6
:                               37 A7 02 B9 24 3E A3 47 5B 95 EB 8A 5F A4 C7 8B
:                               79 DC BB D6 30 22 AB 10 B4 5E 05 1A 84 A7 4B 9F
:                               CF 8E 15 C8 6F 0B D4 75 63 A0 A6 C9 8C BD 96 6D
:                               0B 9B 63 5C A3 2B 20 6F 55 98 27 88 91 49 1A E3
:                               D5 11 30 9E 09 3A 46 91 00 4F 9F 45 10 B5 5C 44
:                               [ Another 1 bytes skipped ]
07F6 02 3:                     INTEGER 65537
:                               }
:                               }
:                               }
07FB A3 20:                     [3] {
07FD 30 1E:                     SEQUENCE {
07FF 30 E:                       SEQUENCE {
0801 06 3:                       OBJECT IDENTIFIER
:                               keyUsage (2 5 29 15)
0806 01 1:                       BOOLEAN TRUE
0809 04 4:                       OCTET STRING, encapsulates
{
080B 03 2:                       BIT STRING 3 unused
bits
:                               '10111'B
:                               }
:                               }
080F 30 C:                     SEQUENCE {
0811 06 3:                     OBJECT IDENTIFIER
:                               basicConstraints (2 5 29
19)
0816 01 1:                     BOOLEAN TRUE
0819 04 2:                     OCTET STRING, encapsulates
{
081B 30 0:                       SEQUENCE {}
:                               }
:                               }
:                               }
:                               }
081D 30 D:                     SEQUENCE {
081F 06 9:                     OBJECT IDENTIFIER
:                               sha1withRSAEncryption (1 2 840
113549 1 1 5)
082A 05 0:                     NULL
:                               }

```



