SafeNet Authentication API for Java 1.3.0

DEVELOPER GUIDE

Document Information

Product Version	1.3.0
Document Part Number	007-013959-001, Rev. C
Release Date	September 2021

Trademarks, Copyrights, and Third-Party Software

Disclaimer

All information herein is either public information or is the property of and owned solely by Thales DIS France S.A. and/or its subsidiaries or affiliates who shall have and keep the sole right to file patent applications or any other kind of intellectual property protection in connection with such information.

Nothing herein shall be construed as implying or granting to you any rights, by license, grant or otherwise, under any intellectual and/or industrial property rights of or concerning any of Thales DIS France S.A. and any of its subsidiaries and affiliates (collectively referred to herein after as “Thales”) information.

This document can be used for informational, non-commercial, internal and personal use only provided that:

> The copyright notice below, the confidentiality and proprietary legend and this full warning notice appear in all copies.

> This document shall not be posted on any network computer or broadcast in any media and no modification of any part of this document shall be made.

Use for any other purpose is expressly prohibited and may result in severe civil and criminal liabilities.

The information contained in this document is provided "AS IS" without any warranty of any kind. Unless otherwise expressly agreed in writing, Thales makes no warranty as to the value or accuracy of information contained herein.

The document could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Furthermore, Thales reserves the right to make **any change or** improvement in the specifications data, information, and the like described herein, at any time.

Thales hereby disclaims all warranties and conditions with regard to the information contained herein, including all implied warranties of merchantability, fitness for a particular purpose, title and non-infringement. In no event shall Thales be liable, whether in contract, tort or otherwise, for any indirect, special or consequential damages or any damages whatsoever including but not limited to damages resulting from loss of use, data, profits, revenues, or customers, arising out of or in connection with the use or performance of information contained in this document.

Thales does not and shall not warrant that this product will be resistant to all possible attacks and shall not incur, and disclaims, any liability in this respect. Even if each product is compliant with current security standards in force on the date of their design, security mechanisms' resistance necessarily evolves according to the state of the art in security and notably under the emergence of new attacks. Under no circumstances, shall Thales be held liable for any third party actions and in particular in case of any successful attack against systems or equipment incorporating Thales products. Thales disclaims any liability with respect to security for direct, indirect, incidental or consequential damages that result from any use of its products. It is further stressed that independent testing and verification by the person using the product is particularly encouraged, especially in any application in which defective, incorrect or insecure functioning could result in damage to persons or property, denial of service or loss of privacy.

CONTENTS

PREFACE

The SafeNet Authentication API for Java enable agents to support all functions required to interact with the SafeNet authentication server.

SafeNet agents are third-party applications with embedded plug-in code, enabling the collection of user names and One-Time Passwords (OTPs) to be passed to the SafeNet server for verification.

The SafeNet Authentication API is represented by a single Java class **`CRYPTOCardAPI`**. The Java class is a singleton class with no public constructor. Class instance can be acquired using the **`getInstance`** public method.

Release Notes

The Customer Release Notes (CRN) document provides important information about this release that is not included in other customer documentation. It is strongly recommended that you read the CRN to fully understand the capabilities, limitations, and known issues for this release. You can view the following CRN for this release:

- *SafeNet Authentication API for Java 1.3.0: Customer Release Notes*

Audience

This document is intended for personnel responsible for maintaining your organization's security infrastructure.

All products manufactured and distributed by Thales are designed to be installed, operated, and maintained by personnel who have the knowledge, training, and qualifications required to safely perform the tasks assigned to them. The information, processes, and procedures contained in this document are intended for use by trained and qualified personnel only.

Document Conventions

This section describes the conventions used in this document.

Command Syntax and Typeface Conventions

This document uses the following conventions for command syntax descriptions, and to highlight elements of the user interface.

Convention	Description
bold	The bold attribute is used to indicate the following: > Command-line commands and options (Type **dir /p**.)

	> Button names (Click **Save As**.) > Check box and radio button names (Select the **Print Duplex** check box.) > Window titles (On the **Protect Document** window, click **Yes**.) > Field names (**User Name:** Enter the name of the user.) > Menu names (On the **File** menu, click **Save**.) (Click **Menu** > **Go To** > **Folders**.) > User input (In the **Date** box, type **April 1**.)
italic	The italic attribute is used for emphasis or to indicate a related document. (See the *Installation Guide* for more information.)
Double quote marks	Double quote marks enclose references to other sections within the document. For example: Refer to "**Error! Reference source not found.**" on page **Error! Bookmark not defined.**.
<variable>	In command descriptions, angle brackets represent variables. You must substitute a value for command line arguments that are enclosed in angle brackets.
[optional] [<optional>] [a | b | c] [<a> | <b> | <c>]	Square brackets enclose optional keywords or <variables> in a command line description. Optionally enter the keyword or <variable> that is enclosed in square brackets, if it is necessary or desirable to complete the task. Square brackets enclose optional alternate keywords or variables in a command line description. Choose one command line argument enclosed within the braces, if desired. Choices are separated by vertical (OR) bars.

{ a | b | c }
{ <a> | <b> | <c> }
	Braces enclose required alternate keywords or <variables> in a command line description. You must choose one command line argument enclosed within the braces. Choices are separated by vertical (OR) bars.

Notifications and Alerts

Notifications and alerts are used to highlight important information or alert you to the potential for data loss or personal injury.

Tips

Tips are used to highlight information that helps to complete a task more efficiently.

> **TIP:** This is some information that will allow you to complete your task more efficiently.

Notes

Notes are used to highlight important or helpful information.

> **NOTE:** Take note. Contains important or helpful information.

Cautions

Cautions are used to alert you to important information that may help prevent unexpected results or data loss.

> CAUTION! Exercise caution. Contains important information that may help prevent unexpected results or data loss.

Warnings

Warnings are used to alert you to the potential for catastrophic data loss or personal injury.

> **WARNING** **Be extremely careful and obey all safety and security measures. In this situation you might do something that could result in catastrophic data loss or personal injury.**

Related Documents

The following document contains related information:

- *SafeNet Authentication API for Java 1.3.0: Customer Release Notes*

Sample Code Usage

Sample codes are provided for demonstration (and test) purposes and must never be used in the production environment. For any damage arising out of such use, Thales Group shall not be liable, whether in contract, tort or otherwise. We do not claim the copyright to certain sample codes, as they may belong to (related / unrelated) third-parties.

Support Contacts

If you encounter a problem while installing, registering, or operating this product, please refer to the documentation before contacting support. If you cannot resolve the issue, contact your supplier or Thales Customer Support.

Thales Customer Support operates 24 hours a day, 7 days a week. Your level of access to this service is governed by the support plan arrangements made between Thales and your organization. Please consult this support plan for further information about your entitlements, including the hours when telephone support is available to you.

Customer Support Portal

The Customer Support Portal, at https://supportportal.thalesgroup.com, is where you can find solutions for most common problems. The Customer Support Portal is a comprehensive, fully searchable database of support resources, including software and firmware downloads, release notes listing known problems and workarounds, a knowledge base, FAQs, product documentation, technical notes, and more. You can also use the portal to create and manage support cases.

> **NOTE:** You require an account to access the Customer Support Portal. To create a new account, go to the portal and click the **REGISTER** link.

Telephone Support

The support portal also lists telephone numbers for voice contact (Contact Us).

Email Support

You can also contact technical support by email at technical.support.DIS@thalesgroup.com.

Third Party Licenses

[ini4j] (http://ini4j.sourceforge.net/index.html)
Copyright 2021 Thales Group

The Legion of the Bouncy Castle

Copyright (c) 2000 - 2021 The Legion of the Bouncy Castle Inc. (https://www.bouncycastle.org/about.html)

CHAPTER 1: Introduction

The SafeNet Authentication API for Java enable agents to support all functions required to interact with the SafeNet authentication server.

SafeNet agents are third-party applications with embedded plug-in code, enabling the collection of user names and One-Time Passwords (OTPs) to be passed to the SafeNet server for verification.

Applicability

The information in this document applies to the following:

- **SafeNet Authentication Service - Service Provider Edition (SAS SPE)** — The on-premises, server version targeted at service providers interested in hosting SAS in their data center(s).
- **SafeNet Authentication Service - Private Cloud Edition (SAS PCE)** — The on-premises, server version targeted at organizations interested in hosting SAS in their private cloud environment.
- **SafeNet Trusted Access (earlier, SAS Cloud)** — The SafeNet’s cloud-based authentication service.

Prerequisites

> **IMPORTANT:** For FIPS support, it is necessary to download the updated BSID key from the SafeNet server.

System Requirements

Java

- Java 8 (version 161 and above)
- Java 11

Supported Platforms

Windows/ Linux/ IBM AIX version 7.1 TL5

CHAPTER 2: Installing and Upgrading SafeNet Authentication API for Java

IMPORTANT: Always work in **Run as administrator** mode when installing, configuring, or uninstalling the SafeNet Authentication API for Java.

Installing SafeNet Authentication API for Java – Windows

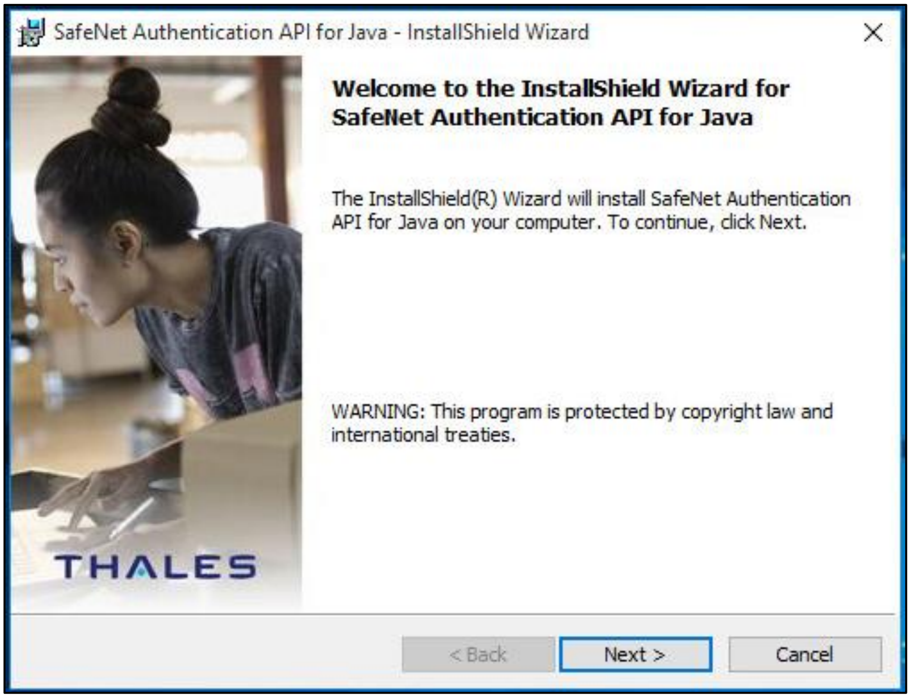
Perform the following steps to install the SafeNet Authentication API for Java on Windows:

1. Locate and execute one of the required installation file:

   ***32-bit**: SafeNet Authentication API for Java x86.exe*

   ***64-bit**: SafeNet Authentication API for Java x64.exe*

2. On the **Welcome to the InstallShield Wizard…** window, click **Next**.



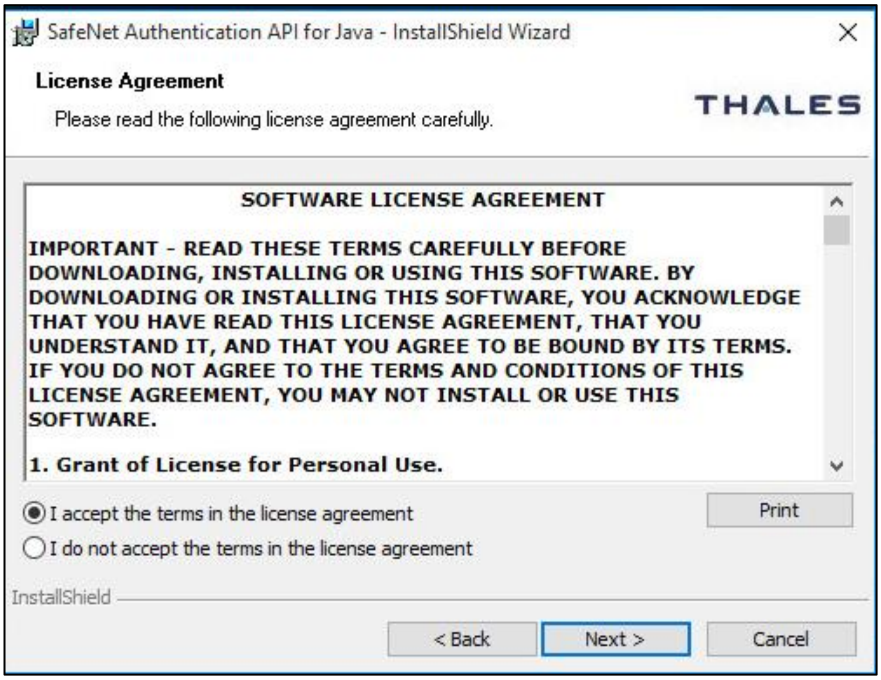
SafeNet Authentication API for Java - InstallShield Wizard

Welcome to the InstallShield Wizard for SafeNet Authentication API for Java

The InstallShield(R) Wizard will install SafeNet Authentication API for Java on your computer. To continue, click Next.

WARNING: This program is protected by copyright law and international treaties.

3. On the **License Agreement** window, read the software license agreement and to proceed, select **I accept the terms in the license agreement** option, and click **Next**.



License Agreement

Please read the following license agreement carefully.

SOFTWARE LICENSE AGREEMENT

IMPORTANT - READ THESE TERMS CAREFULLY BEFORE DOWNLOADING, INSTALLING OR USING THIS SOFTWARE. BY DOWNLOADING OR INSTALLING THIS SOFTWARE, YOU ACKNOWLEDGE THAT YOU HAVE READ THIS LICENSE AGREEMENT, THAT YOU UNDERSTAND IT, AND THAT YOU AGREE TO BE BOUND BY ITS TERMS. IF YOU DO NOT AGREE TO THE TERMS AND CONDITIONS OF THIS LICENSE AGREEMENT, YOU MAY NOT INSTALL OR USE THIS SOFTWARE.

1. Grant of License for Personal Use.

● I accept the terms in the license agreement

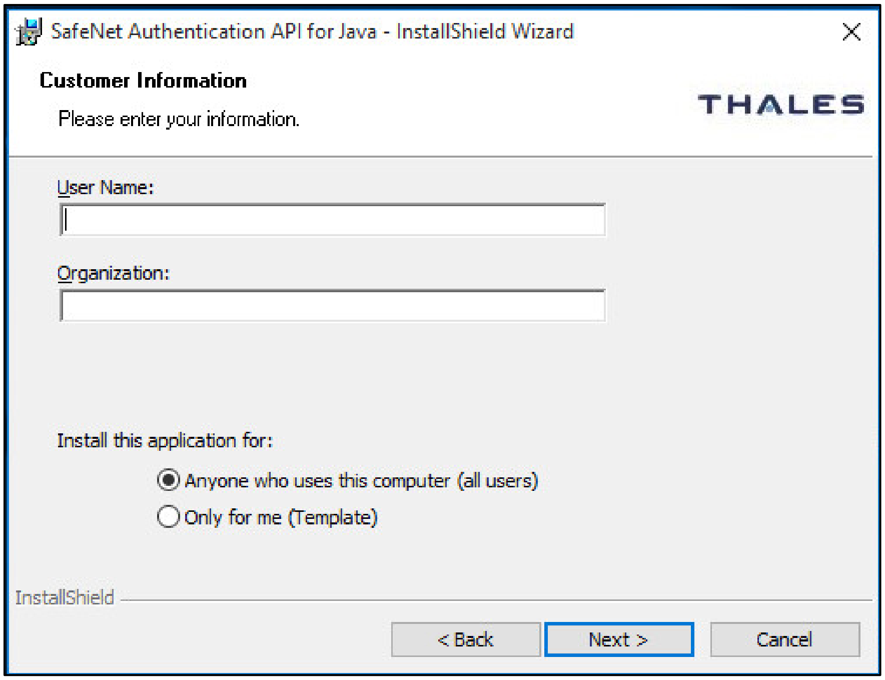
○ I do not accept the terms in the license agreement

4. On the **Customer Information** window, perform the following steps:

   a. In the **User Name** field, enter your user name.

   b. In the **Organization** field, enter the name of your **Organization** (any names can be used).

   c. Click **Next**.



Customer Information

Please enter your information.

User Name:

Organization:

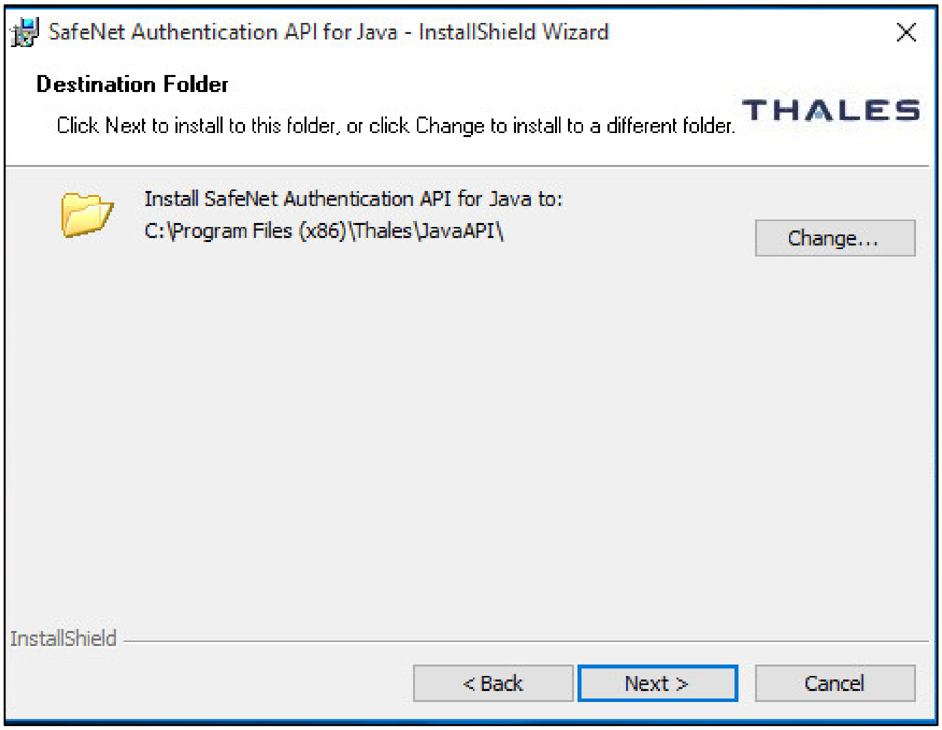
Install this application for:

● Anyone who uses this computer (all users)

○ Only for me (Template)

5. On the **Destination Folder** window, perform one of the following steps:

   - To change the installation folder, click **Change** and navigate to the required folder, and then click **Next**.
   - To accept the default installation folder as displayed, click **Next**.



SafeNet Authentication API for Java - InstallShield Wizard

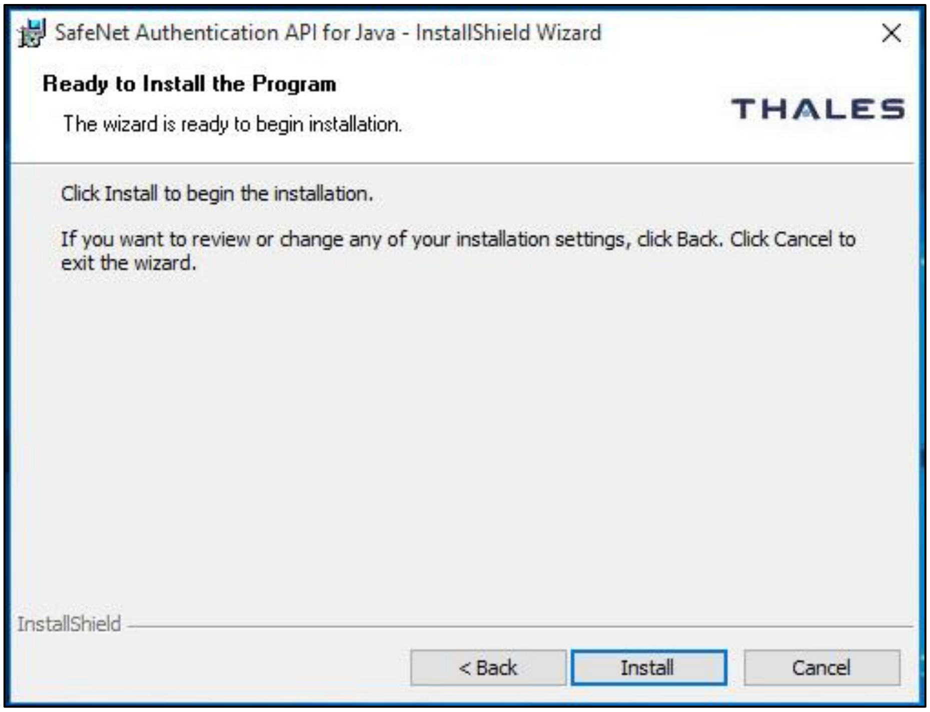
Destination Folder

Click Next to install to this folder, or click Change to install to a different folder.

Install SafeNet Authentication API for Java to:
C:\Program Files (x86)\Thales\JavaAPI\

NOTE: We strongly recommend installing the application in a system protected folder accessible only by an account with management (read) privileges and by the account that initiates the API call.
The default location is in the **Program Files** folder, which is not read protected. This means that the location must be changed to a protected location that cannot be read by non-administrative accounts.

6. On the **Ready to Install the Program** window, click **Install**.

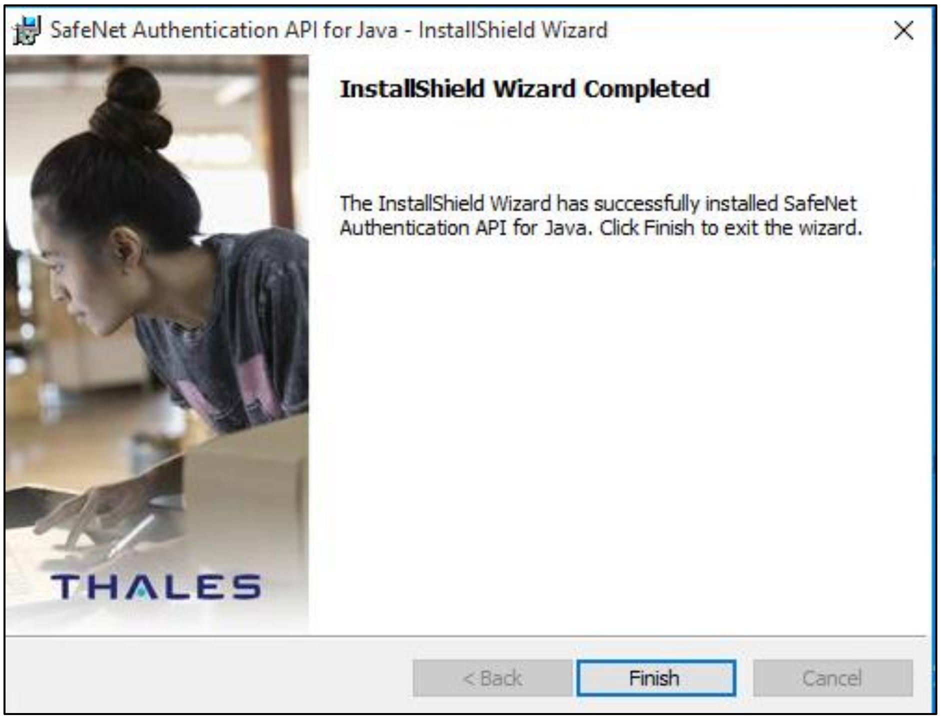


SafeNet Authentication API for Java - InstallShield Wizard
Ready to Install the Program
The wizard is ready to begin installation.
THALES
InstallShield
< Back
Install
Cancel

Click Install to begin the installation.

If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

7. Once the installation process completes, the **InstallShield Wizard Completed** window is displayed.
8. Click **Finish** to exit the installation wizard.



SafeNet Authentication API for Java - InstallShield Wizard
InstallShield Wizard Completed
THALES
< Back
Finish
Cancel

The InstallShield Wizard has successfully installed SafeNet Authentication API for Java. Click Finish to exit the wizard.

Installing SafeNet Authentication API for Java – Linux

Perform the following steps to install the SafeNet Authentication API for Java on Linux:

1. Install the JavaAPI on your Linux Machine, and execute one of the following commands:

   > For RPM package: **`rpm -ivh <release_package_name>.rpm`**

   > For DEB package: **`dpkg -i <release_package_name>.deb`**

2. The package will be installed at the following location for use: **`/usr/local/Thales/javaapi`**

> **NOTE:** After installing the API, we recommend changing the permission settings of the key file:
>
> 1. Navigate to the following path: **`/usr/local/Thales/javaapi/bsidKey`**
> 2. Execute the following command: **`chmod 444 Agent.bsidkey`**

Installing SafeNet Authentication API for Java – IBM Advanced Interactive eXecutive

Perform the following steps to install the SafeNet Authentication API for Java on IBM Advanced Interactive eXecutive:

1. Copy the JavaAPI package to your local disk, and ensure that the following **path** is present and available: **`/usr/local`**
2. From the package's location, extract the JavaAPI package using the following command: **`cd /usr/local && tar -xvf /<release_package_name>.tar`**
3. The JavaAPI package will be extracted at the following path: **`/usr/local/Thales/javaapi`**

> **NOTE:** After installing the API, we recommend changing the permission settings of the key file:
>
> 1. Navigate to the following path: **`/usr/local/Thales/javaapi/bsidKey`**
> 2. Execute the following command: **`chmod 444 Agent.bsidkey`**

Upgrading SafeNet Authentication API for Java

Upgrade from earlier versions of the SafeNet Authentication API for Java to version 1.3.0 is not supported. You need to uninstall the previously installed version of the agent and then install the latest version.

> **NOTE:** Before uninstalling the agent, ensure to take a backup of the *ini* file. You can manually update the customized configuration values in the latest file.

CHAPTER 3: API Java Class

The SafeNet Authentication API is represented by a single Java class **`CRYPTOCardAPI`**. The Java class is a singleton class with no public constructor. Class instance can be acquired using the **`getInstance`** public method.

Methods and Functions

Method

public static CRYPTOCardAPI getInstance()

This public method returns a singleton instance of the `CRYPTOCardAPI` class. This method also performs some internal initializations.

> **NOTE:** We recommend calling the ***getInstance()*** method and setting ***INIPath*** once, and using the same for the life of the application.

Functions

public synchronized void setINIPath(String path)

This function is used to set the location from where the API should load the INI file. If this function is not called before calling *LoadJNILibrary*, the INI file will be loaded from one of the following locations:

- Default deployment location.
- If the INI file is not persent at the default location, the current execution path will be explored.

public synchronized void LoadJNILibrary() throws UnsatisfiedLinkError, Exception

This function performs initialization for the API settings by reading the INI file.

public void Authenticate(String[] arrData)

This function provides SafeNet server authentication and challenge generation functionality. It must be called only after initialization and loading are successfully completed.

String[] arrData Details

Array Element	Stored Value	Response
arrData[0]	UserName	Input value

arrData[1]	Organization	Input value (optional) Normally blank, except in some special cases.
arrData[2]	OTP	Input value
arrData[3]	Challenge	System-returned value
arrData[4]	State	Input and Output value
arrData[5]	ChallengeData	System-returned value
arrData[6]	ChallengeMessage	System-returned value
arrData[7]	ReturnedResult	System-returned value
arrData[8]	BothServersDown	System-returned value
arrData[9]	ErrorMessage	System-returned value
arrData[10]	InIPAddress	Input value (optional) Service Provider IP address Normally blank, except in some special cases.

where,

- **UserName** – A string representing the user name of the individual who is authenticating.
- **Organization** – A string representing the organization to which the individual who is authenticating belongs. This currently should be passed as an empty string to represent the default organization.
- **OTP** – A string representing the user’s passcode. This element may also be set to an empty string or to a single character to indicate if a challenge is required. If a challenge generation is required, send empty state in **arrData[4]** as well. This may take form of either:
  - **[PIN+OTP]** – Server-side PIN authentication.
  - **[OTP]** – Token-side PINs or no PIN.
  - **[PIN]** – When responding to a server-side user changeable PIN change request.
  - **[OTP+PIN]** – If it is configured this way in the SafeNet server.
  - **[StaticPassword]** – User has a static password enabled or is responding to a static password change.
- **Challenge** – A string that may be populated with a challenge/ PIN change/ outer window authentication message.
- **State** – A string that may be populated with a state attribute. When returning a challenge, the same state must be passed back to the server, which was returned by the challenge generation call.

> **ChallengeData** – Data returned as the response to the challenge.

> **ChallengeMessage** – Returned user message appended with the challenge.

> **ReturnedResult** – Returned result (String):

- **0** - Authentication Failed
- **1** - Authentication Succeeded
- **2** - Challenge
- **3** - Server provided PIN
- **4** - User needs to provide PIN
- **5** - Authentication in outer window. Re-authenticate.
- **6** - User must change their static password.
- **7** - Static password change does not satisfy policies.
- **8** - PIN provided doesn’t meet requirements. Please provide a new PIN.

> **BothServersDown** – If both SafeNet servers (primary and secondary) are down, value is 1, otherwise 0.

> **ErrorMessage** – The error message (for logging or the client), if any.

> **InIPAddress** – A string representing the IP address from which the authentication request came. If this parameter is an empty string, the SafeNet server will attempt to detect the IP from which the authentication request came. Under normal circumstances, this should be left empty.

NOTE: Passing **NULL** as a parameter(s) should be avoided. Instead, array must be initialized with empty strings.

Public void checkServerStatus(String[] arrData)

This function checks the status of the primary and secondary SafeNet server. It must be called only after initialization and loading get successfully completed.

String[] arrData Details

Array Element	Stored Value	Response
arrData[7]	ReturnedResult	System-returned value
arrData[8]	Servers’ Status	System-returned value
arrData[9]	ErrorMessage	System-returned value

where,

> **ReturnedResult** – A string return value.

- **0** - If failure.

- **Servers' status** – Represents the status of the servers.
  - **0** – If either primary or secondary or both servers are up and running.
  - **1** – If both servers are down.
- **ErrorMessage** –The error message (for logging or the client), if any.

public void VerifySignature(String[] arrData)

This function verifies the token's signature for a given hash. The system-returned value could be:

- **0** – Signature is incorrect for the provided hash.
- **1** – Signature is correct for the provided hash.

String[] arrData Details

Array Element	**Stored Value**	**Response**
arrData[0]	SerialNumber	Input value
arrData[1]	Hash	Input value
arrData[2]	Signature	Input value
arrData[3]	ReturnedResult	System-returned value

where,

- **SerialNumber** – A string representing the token's serial number.
- **Hash** – A string representing the hash value to verify.
- **Signature** – A string representing the signature (OTP) to verify the provided hash.
- **ReturnedResult** – A string return value.
  - **1** - Success
  - **0** - Failure

> **NOTE:** Passing **NULL** as a parameter(s) should be avoided. Instead, array must be initialized with empty strings.

public BufferedImage getGridSureGrid(String BSIDChallenge) throws Exception

This function creates and returns a GrIDsure grid from the received SafeNet server challenge.

`getGridSureGrid` method returns instance of the BufferedImage class (bitmap).

public String getSoapPayload(String[] arrData) throws Exception

This function gives the API user, an ability to use their own SOAP transport layer. The API itself uses the following java packages for SOAP calls:

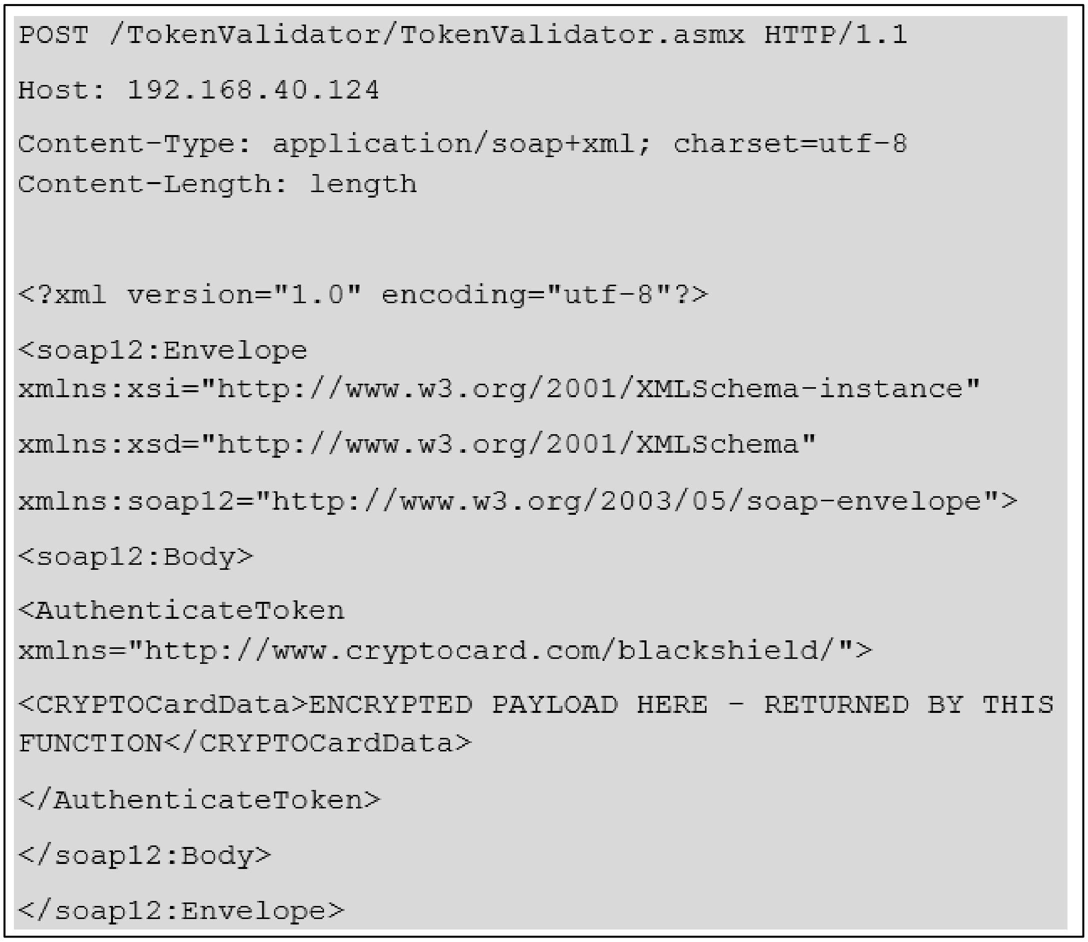
```
java.net.Authenticator;

java.net.HttpURLConnection;

java.net.InetSocketAddress;

java.net.MalformedURLException;

java.net.Proxy;

java.net.Socket;

java.net.SocketAddress;

java.net.URL;
```

The API fully handles HTTP, HTTPS and HTTP/ HTTPS via HTTP Proxy with basic authentication. If, for any reason, it is desired not to use the built-in functionality, this function can be used to get the input parameter required to make a SOAP based authentication call.

Call to this procedure will return an encrypted SOAP payload accepted by the SafeNet server. This payload should be enclosed inside a SOAP envelop to be sent to the SafeNet server.

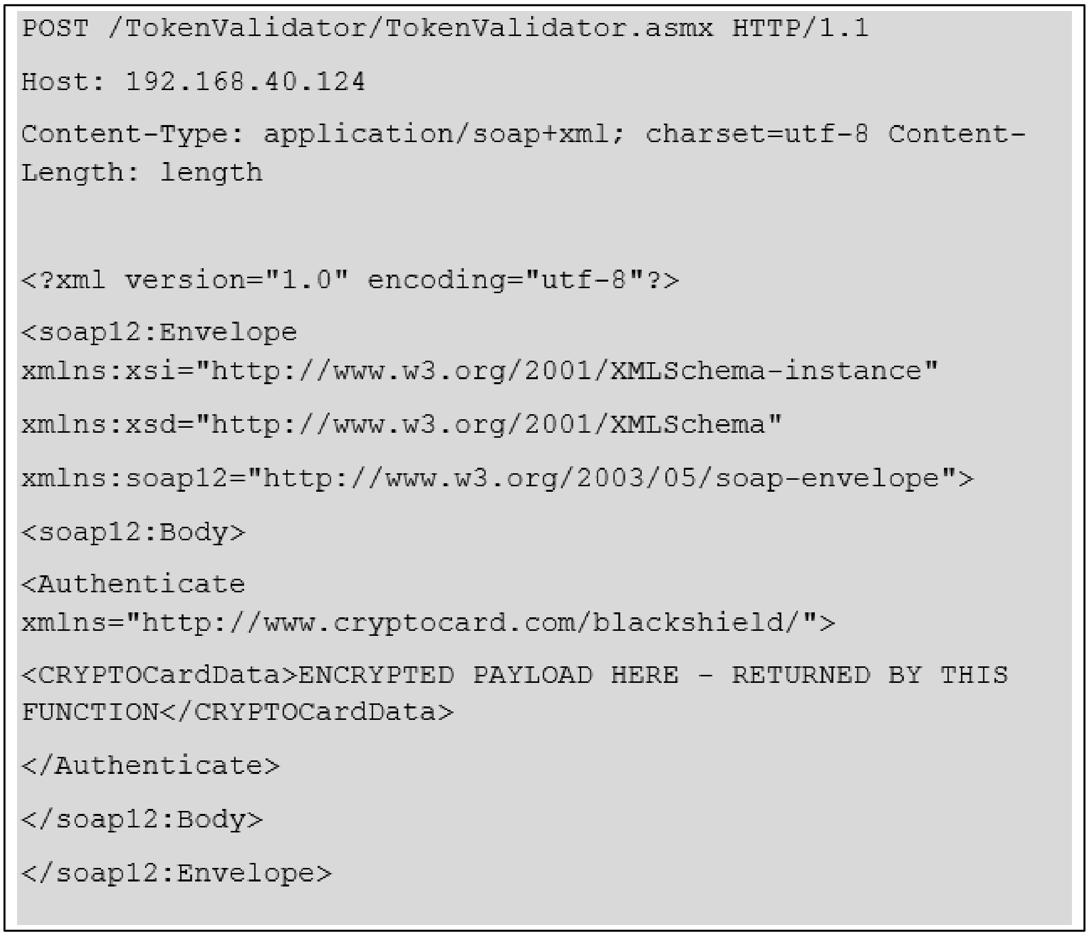
Example: SOAP Request With Headers and Soap Envelope

The following is a sample SOAP 1.2 request and response. The place holders shown needs to be replaced with actual values.



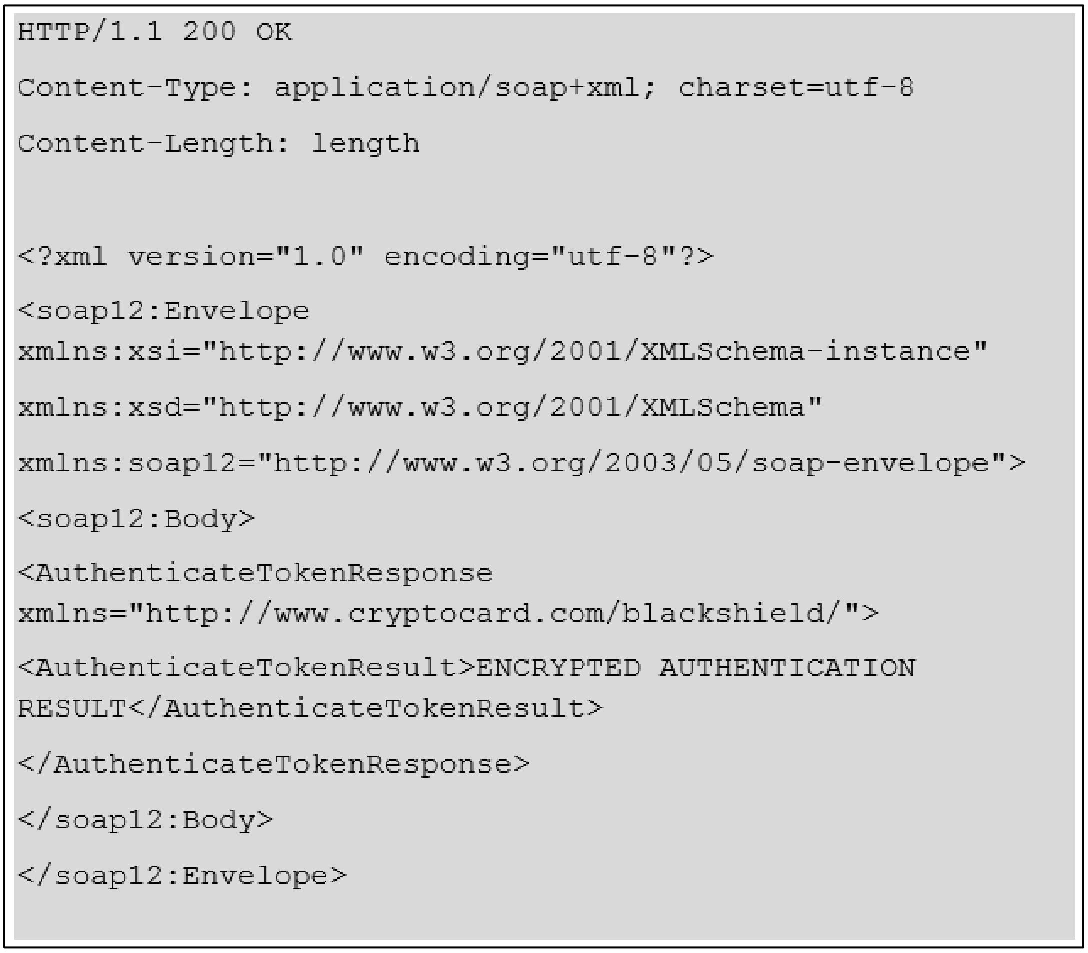
```
POST /TokenValidator/TokenValidator.asmx HTTP/1.1

Host: 192.168.40.124

Content-Type: application/soap+xml; charset=utf-8
Content-Length: length


<?xml version="1.0" encoding="utf-8"?>

<soap12:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:xsd="http://www.w3.org/2001/XMLSchema"

xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">

<soap12:Body>

<AuthenticateToken
xmlns="http://www.cryptocard.com/blackshield/">

<CRYPTOCardData>ENCRYPTED PAYLOAD HERE - RETURNED BY THIS
FUNCTION</CRYPTOCardData>

</AuthenticateToken>

</soap12:Body>

</soap12:Envelope>
```

For older SafeNet server version:



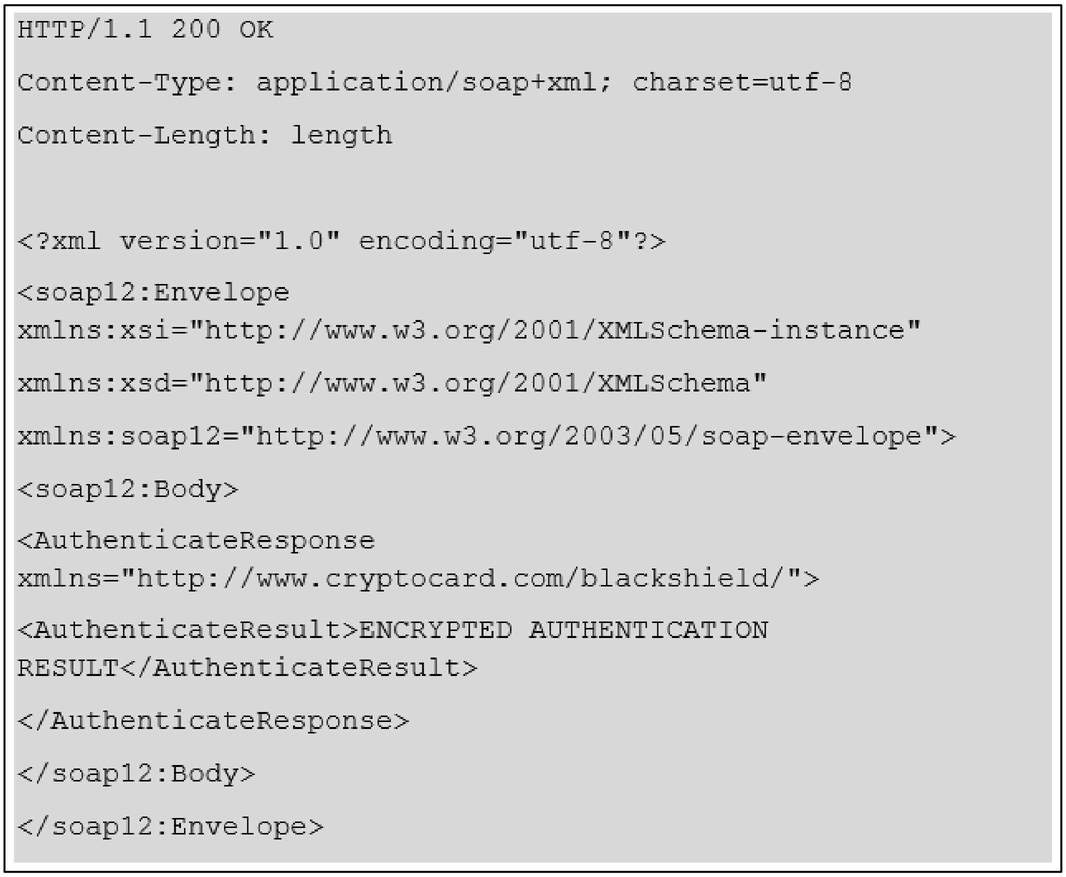
```
POST /TokenValidator/TokenValidator.asmx HTTP/1.1

Host: 192.168.40.124

Content-Type: application/soap+xml; charset=utf-8 Content-
Length: length


<?xml version="1.0" encoding="utf-8"?>

<soap12:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:xsd="http://www.w3.org/2001/XMLSchema"

xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">

<soap12:Body>

<Authenticate
xmlns="http://www.cryptocard.com/blackshield/">

<CRYPTOCardData>ENCRYPTED PAYLOAD HERE - RETURNED BY THIS
FUNCTION</CRYPTOCardData>

</Authenticate>

</soap12:Body>

</soap12:Envelope>
```

If the SOAP call is successful, you should receive the following response:



```
HTTP/1.1 200 OK

Content-Type: application/soap+xml; charset=utf-8

Content-Length: length


<?xml version="1.0" encoding="utf-8"?>

<soap12:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:xsd="http://www.w3.org/2001/XMLSchema"

xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">

<soap12:Body>

<AuthenticateTokenResponse
xmlns="http://www.cryptocard.com/blackshield/">

<AuthenticateTokenResult>ENCRYPTED AUTHENTICATION
RESULT</AuthenticateTokenResult>

</AuthenticateTokenResponse>

</soap12:Body>

</soap12:Envelope>
```

For older SafeNet server version:



```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap12:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">
<soap12:Body>
<AuthenticateResponse
xmlns="http://www.cryptocard.com/blackshield/">
<AuthenticateResult>ENCRYPTED AUTHENTICATION
RESULT</AuthenticateResult>
</AuthenticateResponse>
</soap12:Body>
</soap12:Envelope>
```

public String[] getResultFromAuthenticateResult(String encryptedAuthenticateResult) throws Exception

To get the result back, pass the ENCRYPTED AUTHENTICATION RESULT to the `getResultFromAuthenticateResult` function. This function throws an exception if the input(s) is invalid:

- Array Length is less than 11 characters
- No User Name
- Encryption Failure

All input and output values must follow the details provided in the `Authenticate` function.

String[] arrData Details

Array Element	Stored Value	Response
arrData[0]	UserName	Input value
arrData[1]	Organization	Input value (optional) Normally blank, except in special cases.
arrData[2]	OTP	Input value
arrData[3]	Challenge	System-returned value
arrData[4]	State	Input and Output value
arrData[5]	ChallengeData	System-returned value
arrData[6]	ChallengeMessage	System-returned value
arrData[7]	ReturnedResult	System-returned value
arrData[8]	BothServersDown	Not Applicable
arrData[9]	ErrorMessage	System-returned value
arrData[10]	InIPAddress	Input value

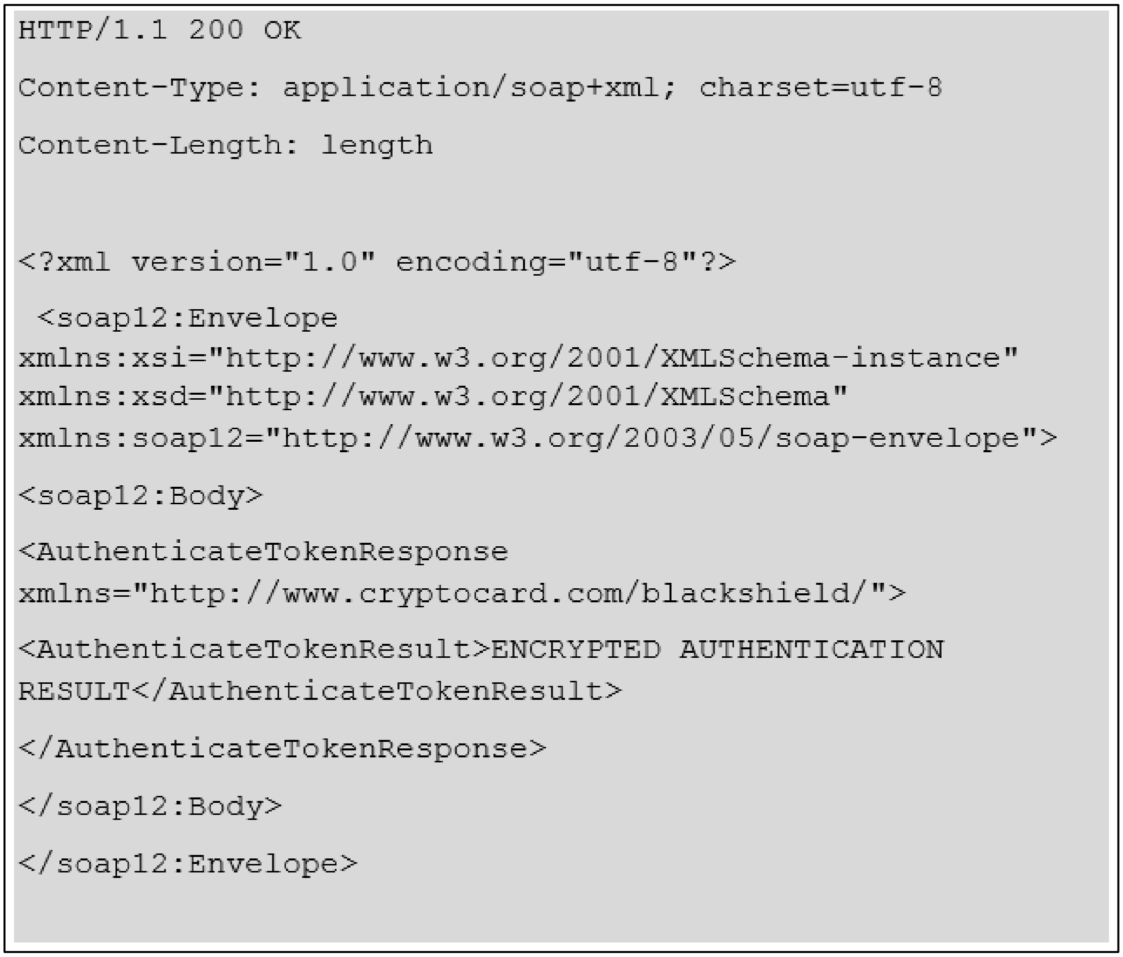
This function is called after obtaining an encrypted result from a SOAP call to the SafeNet server. This function throws and exception if the input is null, empty or it fails to decrypt. It returns the following array:

Challenge	= 0	System-returned value (If returned by Authentication Service)
State	= 1	System-returned value (If returned by Authentication Service)
ChallengeData	= 2	System-returned value (If returned by Authentication Service)
ChallengeMessage	= 3	System-returned value
ReturnedResult	= 4	System-returned value

ErrorMessage	= 5	System-returned value

Example: SOAP Request With Headers and Soap Envelop

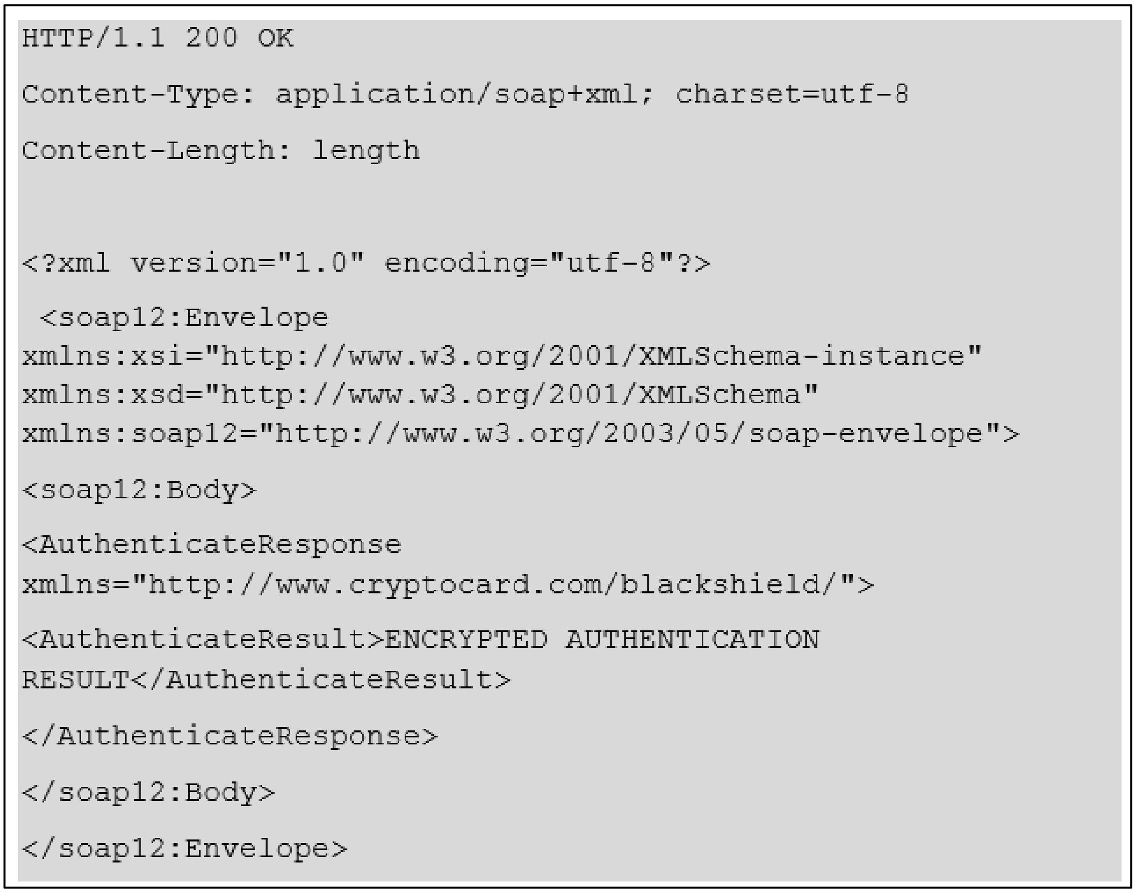
If SOAP call is successful, you should receive the following response:



```
HTTP/1.1 200 OK

Content-Type: application/soap+xml; charset=utf-8

Content-Length: length


<?xml version="1.0" encoding="utf-8"?>

 <soap12:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">

<soap12:Body>

<AuthenticateTokenResponse
xmlns="http://www.cryptocard.com/blackshield/">

<AuthenticateTokenResult>ENCRYPTED AUTHENTICATION
RESULT</AuthenticateTokenResult>

</AuthenticateTokenResponse>

</soap12:Body>

</soap12:Envelope>
```

`encryptedAuthenticateResult` is value of **`AuthenticateTokenResult`** returned by the SafeNet server.

For older SafeNet server version:



```
HTTP/1.1 200 OK

Content-Type: application/soap+xml; charset=utf-8

Content-Length: length


<?xml version="1.0" encoding="utf-8"?>

 <soap12:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://www.w3.org/2003/05/soap-envelope">

<soap12:Body>

<AuthenticateResponse
xmlns="http://www.cryptocard.com/blackshield/">

<AuthenticateResult>ENCRYPTED AUTHENTICATION
RESULT</AuthenticateResult>

</AuthenticateResponse>

</soap12:Body>

</soap12:Envelope>
```

`encryptedAuthenticateResult` is value of **`AuthenticateResult`** returned by the SafeNet server.

CHAPTER 4: Configuring SafeNet Authentication API for Java

IMPORTANT: Always work in **Run as administrator** mode when installing, configuring, or uninstalling the SafeNet Authentication API for Java.

Configuring SafeNet Authentication API for Java – Windows

To configure SafeNet Authentication API for Java in Windows, use the Java API Manager.

Java API Manager

To open the Java API Manager, perform the following steps:

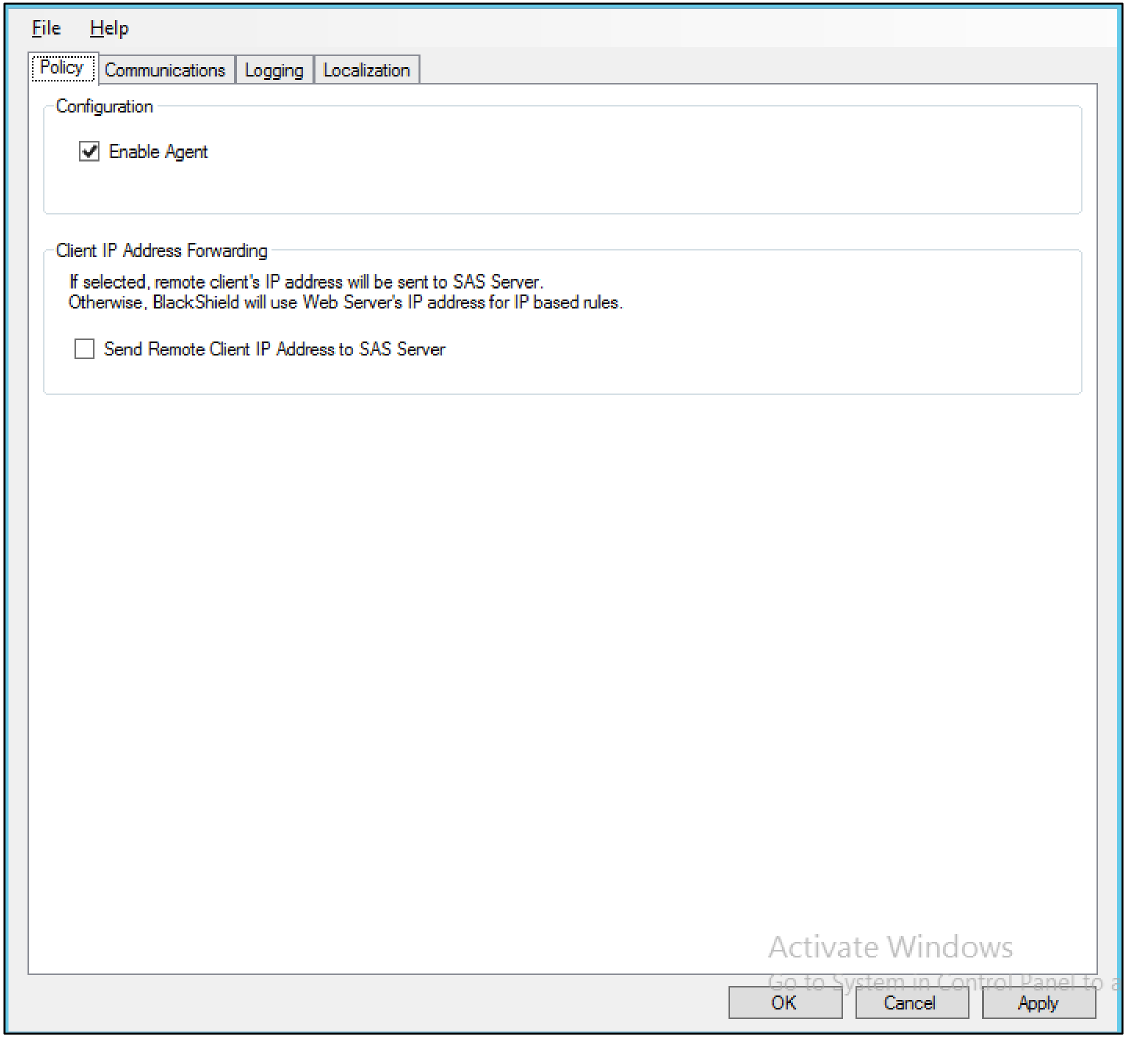
1. Navigate to the following path:
   `Program Files\Thales\JavaAPI`
2. Execute the **`JavaAPIManager.exe`** file.

   NOTE: The `JavaAPIManager.exe` file uses the following files for configuration:
   `\Program Files\Thales\JavaAPI\JavaAPIManager.exe`
   `\Program Files\Thales\JavaAPI\Nini.dll`

Configuring Policy Settings

To configure policy settings, perform the following steps:

1. Select **Policy** tab.



Configuration

☑ Enable Agent

Client IP Address Forwarding

If selected, remote client's IP address will be sent to SAS Server.
Otherwise, BlackShield will use Web Server's IP address for IP based rules.

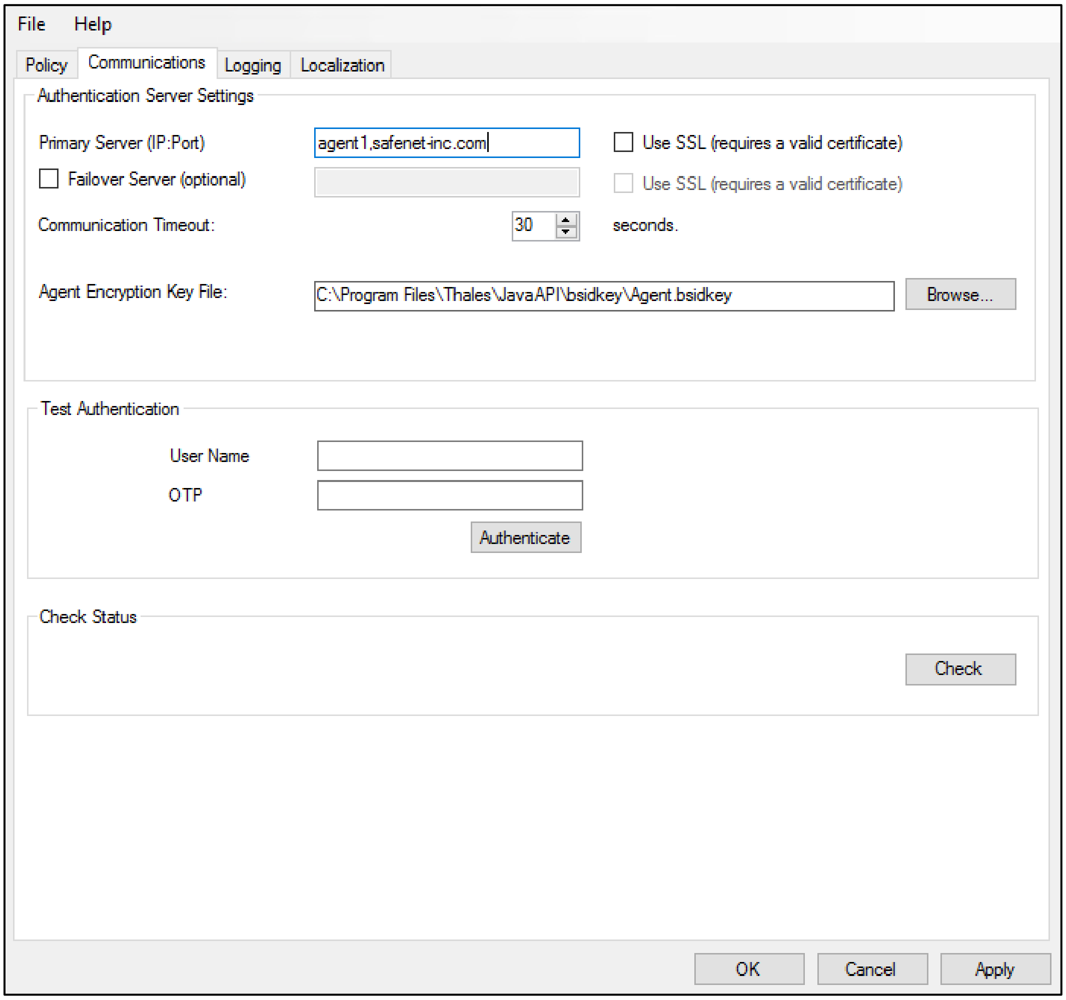
□ Send Remote Client IP Address to SAS Server

2. To send the remote client IP address to the SafeNet server, select **Send Remote Client IP Address to SAS Server checkbox.** Clear the checkbox to use the agent's IP address.

Configuring Communications Settings

To configure communications settings, perform the following steps:

1. Select **Communications** tab.



Primary Server (IP:Port) agent1,safenet-inc.com □ Use SSL (requires a valid certificate)

□ Failover Server (optional) □ Use SSL (requires a valid certificate)

Communication Timeout: 30 seconds.

Agent Encryption Key File: C:\Program Files\Thales\JavaAPI\bsidkey\Agent.bsidkey

2. To connect primary and failover server(s), enter the following fields:
   - **Primary Server (IP:Port)** - (Select **Use SSL**, if required)
   - **Failover Server (optional)** - (Select **Use SSL**, if required)
3. To specify the maximum timeout value for authentication requests sent to the SafeNet server, enter the value (in seconds) in the **Communication Timeout** field.
4. Enter the location of the SafeNet server key file in the **Agent Encryption Key File** field. Browse to the Agent.bsidkey file located at the following path:

   ```
   Program Files\Thales\JavaAPI\bsidkey\Agent.bsidkey
   ```

5. To verify authentication, perform the following steps:

   a. Enter the following fields:
      - **User Name**
      - **OTP**

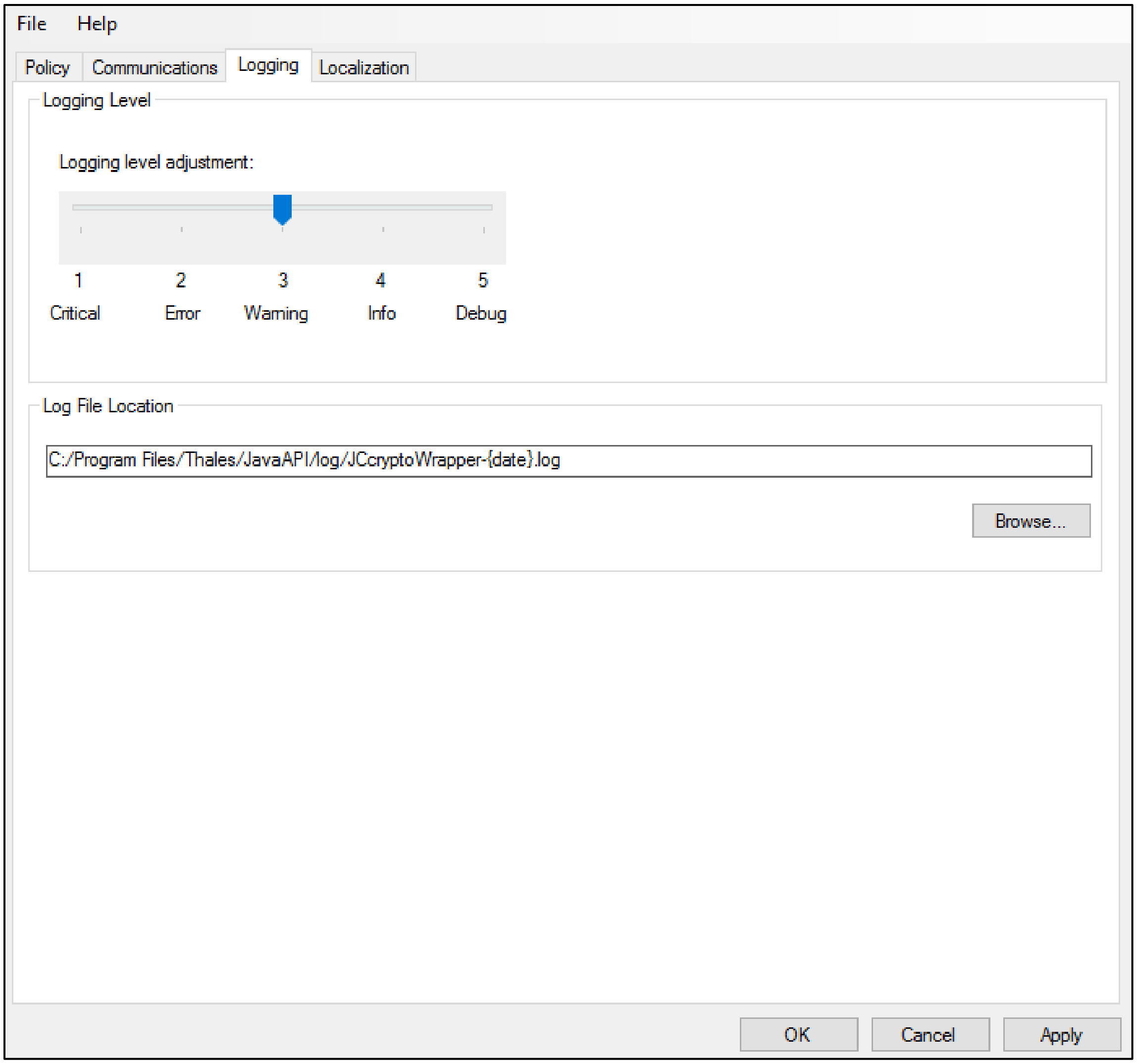
b. Click **Authenticate**. A message is displayed indicating if authentication succeeded or failed.

6. To test if the SafeNet server is running or not**,** Click **Check** in Check Status section.

Configuring Logging Settings

To configure log settings, perform the following steps:

1. Select **Logging** tab.



C:/Program Files/Thales/JavaAPI/log/JCcryptoWrapper-{date}.log

2. Drag the pointer on the **Logging level adjustment** scale to adjust the logging level:
   - **1**: **Critical** (Only critical)
   - **2**: **Error** (Critical and errors)
   - **3**: **Warning** (Critical, errors, and warnings)
   - **4**: **Info** (Critical, errors, warnings, and information messages)
   - **5**: **Debug** (All available information)

   The Java Authentication API will log messages to the file path defined in the **`JCryptoWrapper.ini`** configuration file.

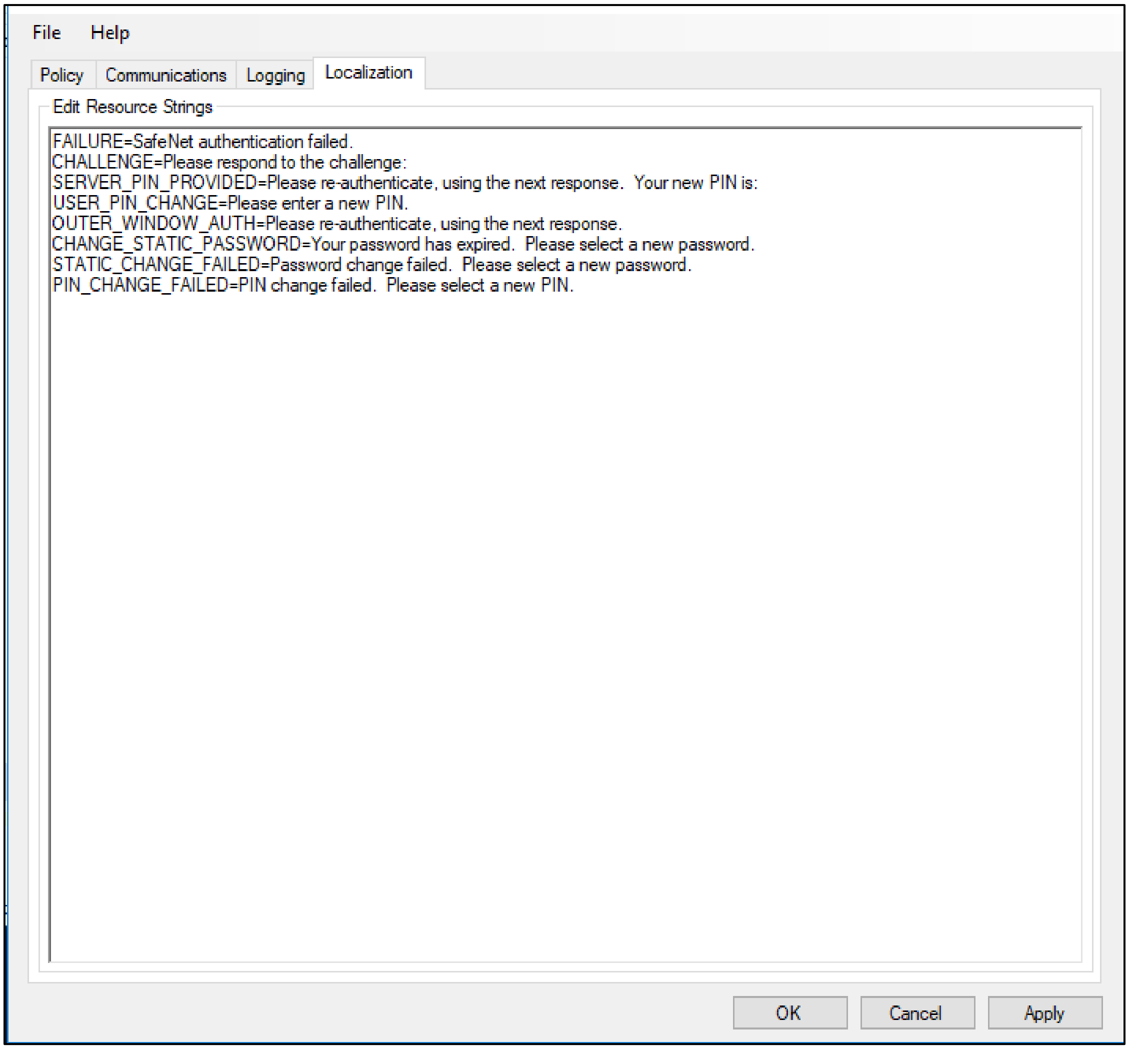
3. **Log File Location** field lets you specify the location where the log files will be saved. The default log file location is: `Program Files\Thales\JavaAPI\log`

   If you change the default file location, ensure that the folder is accessible to all the required users.

Viewing Localization Settings

To view the localization settings, perform the following steps:

1. Select **Localization** tab.



```
FAILURE=SafeNet authentication failed.
CHALLENGE=Please respond to the challenge:
SERVER_PIN_PROVIDED=Please re-authenticate, using the next response.  Your new PIN is:
USER_PIN_CHANGE=Please enter a new PIN.
OUTER_WINDOW_AUTH=Please re-authenticate, using the next response.
CHANGE_STATIC_PASSWORD=Your password has expired.  Please select a new password.
STATIC_CHANGE_FAILED=Password change failed.  Please select a new password.
PIN_CHANGE_FAILED=PIN change failed.  Please select a new PIN.
```

NOTE: The localized text cannot be edited on the **Localization** tab interface. It must be edited in the INI file.

Configuring INI

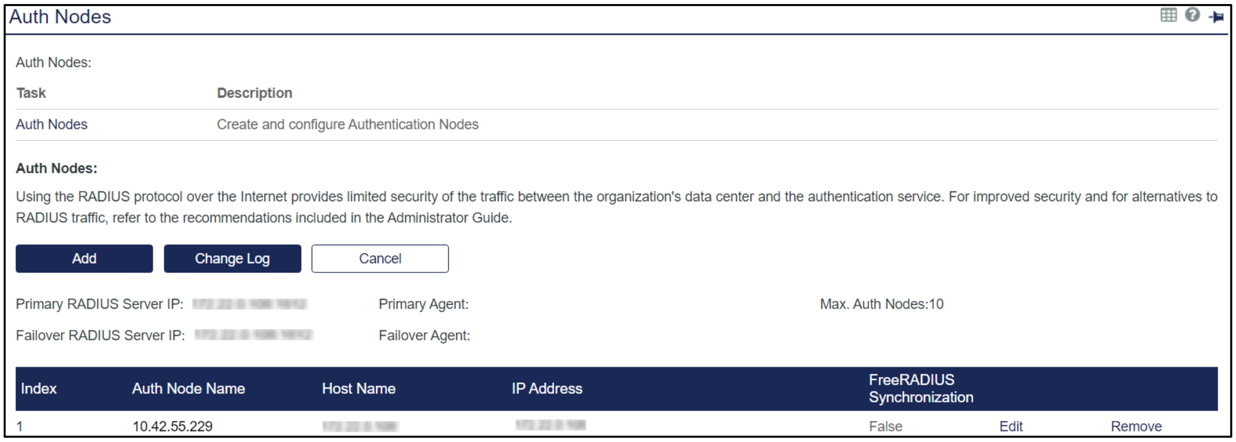
To configure the server details in the INI file, perform the following steps:

1. Edit `JCryptoWrapperWin.ini` (available at `C:/Program Files/Thales/JavaAPI/log/JCryptoWrapperWin.ini`) with the following server details:

Configuration	Description	Default Value
SWITCH_OVER_COUNT	If present, this attempts to switch over to the primary server after a configured number of calls to the secondary server. Valid range: **10** to **99**	**10**
CallTimeout	Time within which the connection between the client and the server must be established. It corresponds to the *Communication Timeout* setting in the Management Console. Valid range: **10** to **99 seconds**	**10 seconds**
RequestTimeout	Time within which a response must be returned before the server terminates the connection. Valid range: **10** to **120 seconds**	**60 seconds**

2. To add IP address to the SAS Auth Nodes tab, perform the following steps:

   a. On the Management console, select **Comms > Auth Nodes** and select the **Auth Nodes** link.



Auth Nodes

Auth Nodes:

Task	Description
Auth Nodes	Create and configure Authentication Nodes

Auth Nodes:

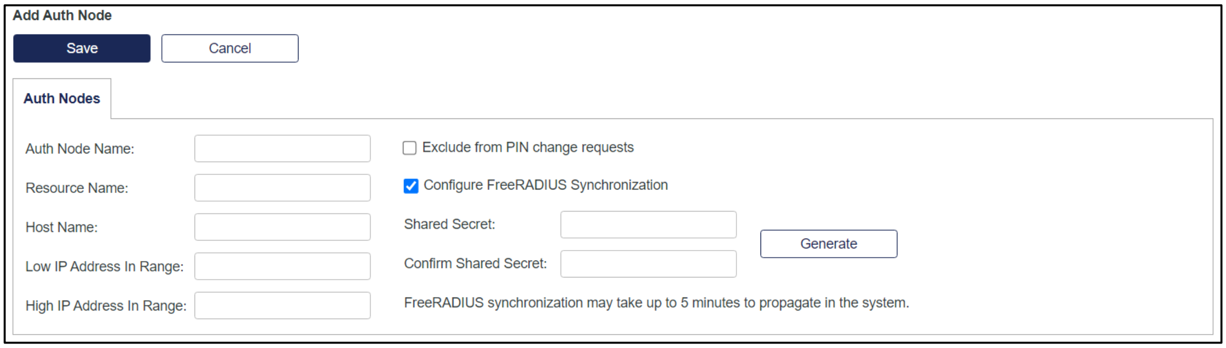
Using the RADIUS protocol over the Internet provides limited security of the traffic between the organization's data center and the authentication service. For improved security and for alternatives to RADIUS traffic, refer to the recommendations included in the Administrator Guide.

Primary RADIUS Server IP: [redacted]   Primary Agent: Max. Auth Nodes:10

Failover RADIUS Server IP: [redacted]   Failover Agent:

Index	Auth Node Name	Host Name	IP Address	FreeRADIUS Synchronization		
1	10.42.55.229	[redacted]	[redacted]	False	Edit	Remove

   b. Click the **Add** button and enter the information on the **Auth Nodes** tab.



Add Auth Node

Save | Cancel

Auth Nodes

Auth Node Name:	
Resource Name:	
Host Name:	
Low IP Address In Range:	
High IP Address In Range:	

□ Exclude from PIN change requests

☑ Configure FreeRADIUS Synchronization

Shared Secret:

Confirm Shared Secret:

Generate

FreeRADIUS synchronization may take up to 5 minutes to propagate in the system.

c. Click **Save**.

Configuring SafeNet Authentication API for Java – Linux

To configure Authentication API for Java in Linux, perform the following steps:

1. Edit **`JCryptoWrapper.ini`** (available at **`/usr/local/Thales/javaapi/ini`**) with the following server details:

Configuration	Description	Default Value
PrimaryProtocol	Select **http / https**.	
PrimaryServer	Enter the primary SafeNet server host.	
PrimaryServerPort	Enter the port number (for example, **80/443**).	
SWITCH_OVER_COUNT	If present, this attempts to switch over to the primary server after a configured number of calls to the secondary server. Valid range: **10** to **99**	**10**
CallTimeout	Time within which the connection between the client and the server must be established. It corresponds to the *Communication Timeout* setting in the Management Console. Valid range: **10** to **99 seconds**	**10 seconds**
RequestTimeout	Time within which a response must be returned before the server terminates the connection. Valid range: **10** to **120 seconds**	**60 seconds**

2. To add IP address to the SAS Auth Nodes tab, refer to Step 2 of Configuring INI section.

Running the Sample

> **IMPORTANT:** Before running the sample, ensure that the current working directory is: **`usr/local/Thales/javaapi`**

After completing the steps illustrated in Configuring SafeNet Authentication API for Java, you need to perform the following additional actions to run the sample:

- For Authentication call: **`java –jar TestAPI <user> <password>`**
- For Signature verification: **`java –jar TestAPI <token ID> <HASH> <Generated OTP>`**

CHAPTER 5: Use Case Scenarios

The SafeNet server architecture supports the use of a token-side or a server-side PIN in either **QuickLog** or **Challenge-Response** mode. In addition, the application using the API must support challenges, inner / outer window authentication, and static password authentication. The following sections discuss these features in detail.

> **LOCALIZATION NOTE:** To support localization, SafeNet server returns only necessary data in its challenge messages. The application is required to construct a localized version of it, to display to the client.
> For example, SafeNet server would return only **19863257**, but the application would display, **Please respond to the challenge: 19863257**.

> **Mode:** Tokens can operate in either **Challenge-Response** or **QuickLog** mode.
>
> **QuickLog** mode is recommended because it simplifies the login experience (and strengthens security) by eliminating the requirement to enter a challenge into a token to get an OTP. QuickLog do not rely on time to remain synchronized with the server. Instead, each time an event-based token is activated, a new token code is generated.

Basic Authentication

The communication between the application and the server uses challenge messages and states, similar to the RADIUS protocol. The following scenario shows the most basic interaction between the application and the server:

1. The application issues an authentication request that includes the user name, and a passcode.
2. The server responds with one of nine possible return codes, as outlined in Returned Result.

Challenge-Response Authentication

The challenge message and state attribute issued from the authenticating server are central to the concept of challenge-response authentication, outer window authentication, and server-side PIN changes. This mechanism is employed to authenticate tokens in challenge-response mode in the following manner:

1. The application issues an authentication request that includes the user name, and an empty passcode.
2. The server responds with a challenge message containing a challenge string. For example, **Challenge: 19863257**, and a state attribute.
3. The authenticating application responds to the challenge by issuing another authentication request that includes the same user name, a response, and the state attribute.

> **LOCALIZATION NOTE:** To support localization, the SafeNet server returns necessary data only in its challenge messages, and the agent is required to construct a localized version of it to display to the client. For example, the SafeNet server would return only “19863257,” and the SafeNet agent would display “Please respond to the challenge: 19863257.”

Outer Window Authentication

User authentication through inner/ outer window authentication uses challenge messages and state attributes, similar to the Challenge-Response Authentication. In outer window authentication, users provide a match in a large look-ahead window, and respond to a follow-up challenge by providing the exact next OTP from their token. The follow sequence illustrates the process:

1. The application issues an authentication request that includes the user name, and a passcode.
2. The server finds a match for the provided OTP in the outer window, and then issues a challenge to the client containing an outer window authentication string, for example: **Please re-authenticate using the next OTP from your token**, and a state attribute.
3. The authenticating application responds to the challenge by issuing another authentication request that includes the same user name, a response, and the state attribute.

> **NOTE:** Refer the localization note in the **Challenge-Response** section.

PIN Authentication

SafeNet server supports several PIN types:

- No PIN
- Fixed PIN (token-side PIN validation)
- User-changeable PIN (token-side PIN validation)
- Fixed PIN stored on server
- User-changeable PIN stored on server
- Server-changeable PIN stored on server

The SafeNet server authentication mechanism supports incoming passcodes in the following formats:

- [PIN+OTP]
- [OTP]
- [NEWPIN]
- [StaticPassword]
- [null] - empty passcode to request a challenge

PINs stored on the server can be user- or server-changeable. To accommodate this, leverage the challenge framework in the following manner:

User-Changeable PIN Stored on Server

1. The application issues an authentication request that includes the user name, and a passcode.
2. The server finds a match for the provided OTP and determines that the PIN must be changed.
3. The server issues a challenge to the client containing a PIN change string, for example, **Your PIN has expired. Please enter a new PIN** and a state attribute.

   NOTE: Refer the localization note in the **Challenge-Response** section.

4. The authenticating application responds to the challenge by returning a new PIN and the state attribute.

Server-Changeable PIN Stored on Server

1. The application issues an authentication request that includes the user name, organization name, and a passcode.
2. The server finds a match for the provided OTP and determines that the PIN must be changed.
3. The server issues a challenge to the client containing a PIN change string, for example, **Your new PIN is 628. Please re-authenticate using this new PIN and your next passcode** and a state attribute.

   NOTE: Refer the localization note in the **Challenge-Response** section.

4. The authenticating application responds to the challenge by issuing another authentication request that includes the user name, organization name, the new PIN and OTP, and the state attribute.

Static Password Authentication

SafeNet server offers the option of static password authentication, including, enabling the user to change the password. The challenge-response architecture can be used in the following manner:

1. The application issues an authentication request that includes the user name, and a static password.
2. If the user is not required to change the password and it is correct, the server returns access-accept.
3. If the user is required to change the password, a challenge message is issued to the client, for example, **Your password has expired. Please enter a new password** and a state attribute.

   NOTE: Refer the localization note in the **Challenge-Response** section.

4. If a challenge message has been issued in step Error! Reference source not found., the authenticating application responds to the challenge by issuing an authentication request that includes the user name, the new static password, and the state attribute.

CHAPTER 6: Agent Key File and Additional Deployment

Agent Key File

The SafeNet server API uses an encrypted key file to secure communication with the server. To accomplish this, a key file is loaded and registered with agent, and a matching key is registered with the authentication server.

API Example

See the sample code in **TestAPI.java** for an example on how to call the **Authenticate** and **VerifySignature** methods.

To use the API example, the following components are required:

- BSIDJavaAPI.jar
- NetBeans IDE or Eclipse or any other Java development tool

Deploying on Additional Computers

To deploy your completed application to another computer, the following files are required to support the SafeNet server API:

Windows

- `Program Files\Thales\JavaAPI\bsidkey\Agent.bsidkey`
- `Program Files\Thales\JavaAPI\ini\JCryptoWrapperWin.ini`
- `Program Files\Thales\JavaAPI\jar\BSIDJavaAPI.jar`

For 32-bit operating systems, the following files are required:

- `Program Files\Thales\JavaAPI\bin\x86\lib\bc-fips-1.0.2.jar`
- `Program Files\Thales\JavaAPI\bin\x86\lib\BSIDJavaAPI.jar`
- `Program Files\Thales\JavaAPI\bin\x86\lib\ini4j-0.5.4.jar`

For 64-bit operating systems, the following files are required:

- `Program Files\Thales\JavaAPI\bin\x64\lib\bc-fips-1.0.2.jar`
- `Program Files\Thales\JavaAPI\bin\x64\lib\BSIDJavaAPI.jar`
- `Program Files\Thales\JavaAPI\bin\x64\lib\ini4j-0.5.4.jar`

Linux

- `/usr/local/Thales/javaapi/lib/BSIDJavaAPI.jar`
- `/usr/local/Thales/javaapi/lib/bc-fips-1.0.2.jar`
- `/usr/local/Thales/javaapi/lib/ini4j-0.5.4.jar`
- `/usr/local/Thales/javaapi/bsidkey/Agent.bsidkey`
- `/usr/local/Thales/javaapi/ini/JCryptoWrapper.ini`

HTTP Proxy

If your organization uses a proxy server to access an extranet or intranet, you need to also configure proxy settings in the INI file. The agent can work with an HTTP proxy only with basic or anonymous authentication.

`USE_PROXY=0` (to use proxy server, set `1`)

`PROXY_SERVER=127.0.0.1`

`PROXY_PORT=8080`

`PROXY_USER=User` (optional)

`PROXY_PASSWORD=Password`

CHAPTER 7: Troubleshooting

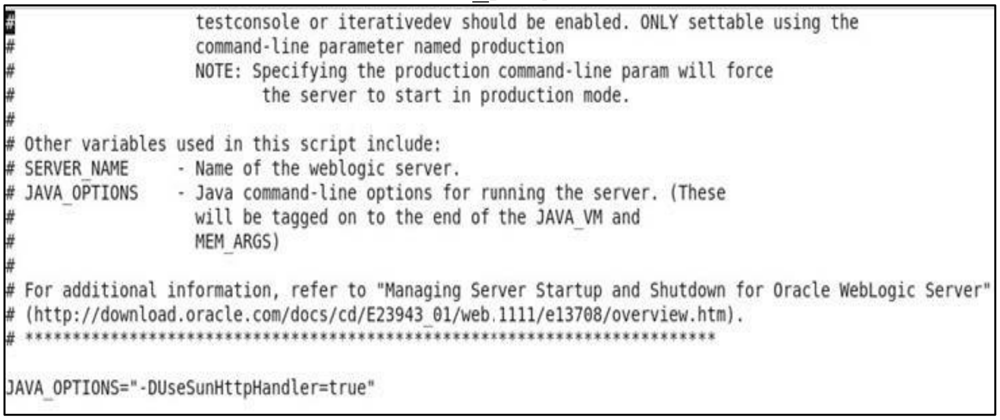
Self-Signed Certificates

For in-house SafeNet server deployments with self-signed certificates, it is recommended to set **IGNORE_CERTIFICATE_ERRORS** parameter (while configuring the INI file) to **1** to avoid Secure Sockets Layer (SSL) errors. If the parameter is set to **1**, the certificate checks will be ignored. The default value **0** ensures that the security certificate checks will be forced while communicating with SafeNet servers. The parameter is valid for both Windows and Linux operating systems.

WebLogic Server SSL Error

You may encounter an error while making an HTTPS connection to the SafeNet server from the WebLogic Server (WLS). If the certificate policy is different in the WebLogic and the stand-alone Java program, it is advised to use the standard Sun SSL implementation. The following setting is mandatory, if you are using the HTTPS protocol.

1. Set `–DuseSunHttpHandler` flag to `true` in the WLS startup script available at the following location: `<WLS-INSTALL PATH>/oracle/Middleware/user_projects/domains/domain/bin/setDomainEnv.sh`



```
#                    testconsole or iterativedev should be enabled. ONLY settable using the
#                    command-line parameter named production
#                    NOTE: Specifying the production command-line param will force
#                           the server to start in production mode.
#
# Other variables used in this script include:
# SERVER_NAME     - Name of the weblogic server.
# JAVA_OPTIONS    - Java command-line options for running the server. (These
#                   will be tagged on to the end of the JAVA_VM and
#                   MEM_ARGS)
#
# For additional information, refer to "Managing Server Startup and Shutdown for Oracle WebLogic Server"
# (http://download.oracle.com/docs/cd/E23943_01/web.1111/e13708/overview.htm).
# ****************************************************************************

JAVA_OPTIONS="-DUseSunHttpHandler=true"
```

2. Restart WLS.

> **NOTE:** Configuring WLS is important to avoid SSL connection, certificate validation, and SSL handshake errors.