บริการ TEDA e-Timestamping

Version: 2.1

October 6, 2021

ETDA
สพธอ
www.etda.or.th

Change History

Date	Version	Description	Author
01/02/2019	0.1	รวบรวมจากคู่มือที่มีอยู่ เพื่อเป็น version เริ่มต้น 1) คู่มือ_OpenSSL for Windows_20181024.docx 2) คู่มือ_Runnable jar for PDFTimestamping_20181227_win.docx 3) Timestamp API Specification_V2.0.docx	Jariya S.
11/02/2019	1.0	Baseline document	Jariya S.
25/04/2019	1.1	Change test.time to time-test	Pragasit C.
09/10/2019	1.2	Update authentication step for TSA-API	Natham T.
12/02/2020	1.3	Correct test TSA API url and add negative response example for API key	Natham T.
11/06/2020	1.4	Remove invalid API URL	Natham T.
03/02/2021	1.5	1) Update P12 creation step to support certificate that has intermediate chain 2) Update sample timestamp source code screenshot	Natham T.
17/02/2021	1.6	เพิ่มส่วน “การขอใช้บริการ”, “ขั้นตอนที่ 4 การติดตั้ง KeyStoreFile เพื่อทดสอบฯ” และ “การติดตั้ง Timestamp Proxy Server”	Jariya S.
03/03/2021	1.7	เพิ่มภาคผนวก ตัวอย่างหนังสือขอใช้บริการ	Avatsada C.
10/08/2021	1.8	แก้ไขเนื้อหา แก้ไขคำผิด และปรับปรุงรูปจากระบบ TEDA Web Validation Portal	Napon S.
27/08/2021	2.0	Baseline document	Jariya S.
04/10/2021	2.1	เพิ่ม support-center@etda.or.th ในบทที่ 1 การขอใช้บริการ	Jariya S.

ภาพรวม

e-Timestamping คือ บริการประทับรับรองเวลาอิเล็กทรอนิกส์ เพื่อรับรองการมีอยู่ของเอกสาร ณ เวลานั้น ๆ และสามารถใช้ในการตรวจสอบว่าเอกสารอิเล็กทรอนิกส์ที่ได้รับการประทับรับรองเวลาแล้วนั้น ถูกแก้ไขหรือไม่

ระบบที่ให้บริการประทับรับรองเวลาอิเล็กทรอนิกส์ ของ ETDA หรือ Timestamping Authority (TSA) มีการตรวจสอบความถูกต้องของเวลาบนเครื่องแม่ข่ายที่ใช้อ้างอิง (NTP Server) เทียบกันจาก 3 แหล่ง คือ สถาบันมาตรวิทยาแห่งชาติ, กรมอุทกศาสตร์ กองทัพเรือ และ สพธอ.

หลักการทำงานโดยย่อของบริการนี้คือ ผู้ใช้บริการส่ง hash ของสิ่งที่ต้องการทำ timestamp เข้ามา แล้วระบบจะส่ง timestamp token ที่คู่กับ hash นั้น กลับไปให้ ซึ่งผู้ใช้ฯสามารถนำ timestamp token ไปใช้งานได้ ทั้งนำไปใช้เป็นเวลาในการลงลายมือชื่อดิจิทัล (แทนที่จะใช้เวลาจากเครื่องที่ลงลายมือชื่อดิจิทัล) หรือประกอบเข้ากับสิ่งที่ต้องการประทับรับรองเวลา

โดยหากต่อมามีการแก้ไขเนื้อหาของสิ่งที่ได้ทำ timestamp ไปแล้ว ค่า hash ที่คำนวณใหม่ (โดยใช้ฟังก์ชั่นเดียวกับตอนที่คำนวณเพื่อส่งไปทำ timestamp) ก็จะเปลี่ยนแปลงไปด้วย หากทำการตรวจสอบ ก็จะทราบได้ว่าเกิดการแก้ไขขึ้น ซึ่ง ETDA ได้จัดทำ website และให้บริการ API ที่ใช้ในการตรวจสอบ โดยปัจจุบันรองรับไฟล์ประเภท PDF และ XML ดูรายละเอียดได้ที่บท "การตรวจสอบความถูกต้องของการประทับรับรองเวลา ด้วย TEDA Web Validation Portal"

บริการ e-Timestamping มีอยู่ 2 รูปแบบคือ

1. การรับ-ส่ง message ตามรูปแบบของ RFC3161: สามารถ download ตัวอย่าง source code (JAVA) ได้จาก project PDFTimestamping ใน ETDA GitHub ที่ https://github.com/ETDA/PDFTimestamping หรือหากต้องการประทับรับรองเวลาบนไฟล์ PDF ก็สามารถสร้าง Runnable JAR file จาก project นี้ แล้วเรียกใช้งานได้ โดยดูขั้นตอนได้ที่บท "การสร้าง Runnable JAR file เพื่อปรับทับรับรองเวลาลงบนไฟล์ PDF"
2. HTTP API: ดูรายละเอียดได้ที่บทที่ 6 "Timestamp API Specification"

การยืนยันตัวตนของผู้ขอใช้บริการ สำหรับบริการ **e-Timestamping** ในรูปแบบ **RFC3161** จะใช้การยืนยันตัวตนด้วยรูปแบบ **Certificate Authentication** ส่วนบริการ **e-Timestamping** ในรูปแบบ **HTTP API** จะใช้การยืนยันตัวตนด้วยรูปแบบ **API key** แทนซึ่งสามารถดูรายละเอียดขั้นตอนได้ที่บทต่อๆไป

บทที่ 1 การขอใช้บริการ e-Timestamping

สพธอ. ขอสงวนสิทธิ์ในการให้บริการ e-Timestamping แก่หน่วยงานรัฐเท่านั้น โดยไม่คิดค่าบริการ (ข้อมูล ณ เดือน ต.ค. 2564) ซึ่งการขอใช้บริการมีขั้นตอนดังนี้

การขอทดสอบใช้บริการ

1. ดาวน์โหลดไฟล์ เพื่อกรอกแบบคำขอทดสอบบริการ (TEDA Test Application Form)
2. ทำการสร้าง Certificate Signing Request (CSR) file ตามรายละเอียดในบท "การยืนยันตัวตนผู้ขอใช้บริการด้วยรูปแบบ Certificate Authentication" แต่หากเลือกใช้บริการในรูปแบบ HTTP API ให้ข้ามขั้นตอนนี้ไป
3. ส่งแบบคำขอทดสอบบริการที่กรอกแล้ว มาทางอีเมลถึง support-center@etda.or.th และ eservice@etda.or.th

   To: support-center@etda.or.th; eservice@etda.or.th

   Subject: ขอทดสอบการใช้งานระบบ e-Timestamping

   โดยหากใช้บริการในรูปแบบ RFC3161 ให้แนบ CSR file จากขั้นตอนที่ 2 มาด้วย จากนั้น ETDA จะสร้าง credential ในการเข้าใช้ระบบส่งกลับไปให้ทางอีเมล พร้อม URL ของระบบทดสอบ
4. เมื่อทำการติดตั้ง credential ที่ได้รับจาก ETDA ลงในระบบของผู้ใช้บริการแล้ว สามารถทดสอบบริการโดยส่งไฟล์เข้ามาทำ Timestamp ที่ระบบทดสอบตาม URL ที่แจ้งไว้

การขอใช้บริการบนระบบปฏิบัติการจริง

1. ทำหนังสือขอใช้บริการจากหัวหน้าหน่วยงาน มาถึงผู้อำนวยการสำนักงานพัฒนาธุรกรรมทางอิเล็กทรอนิกส์ (ดูตัวอย่างได้จาก ภาคผนวก)
2. ดาวน์โหลดไฟล์ เพื่อกรอกแบบคำขอใช้บริการ (TEDA Application Form)
3. ทำการสร้าง Certificate Signing Request (CSR) file สำหรับใช้บนระบบปฏิบัติการจริง แต่หากเลือกใช้บริการในรูปแบบ HTTP API ให้ข้ามขั้นตอนนี้ไป
4. ส่งแบบคำขอใช้บริการที่กรอกแล้ว พร้อมหนังสือขอใช้บริการ และไฟล์ที่ทำ Timestamp จากระบบทดสอบ มาทางอีเมลถึง support-center@etda.or.th และ eservice@etda.or.th

   To: support-center@etda.or.th; eservice@etda.or.th

   Subject: ขอใช้งานระบบ e-Timestamping

   โดยหากใช้บริการในรูปแบบ RFC3161 ให้แนบ CSR file จากขั้นตอนที่ 3 มาด้วย จากนั้น ETDA จะสร้าง credential ในการเข้าใช้ระบบส่งกลับไปให้ทางอีเมล พร้อม URL ของระบบปฏิบัติการจริง

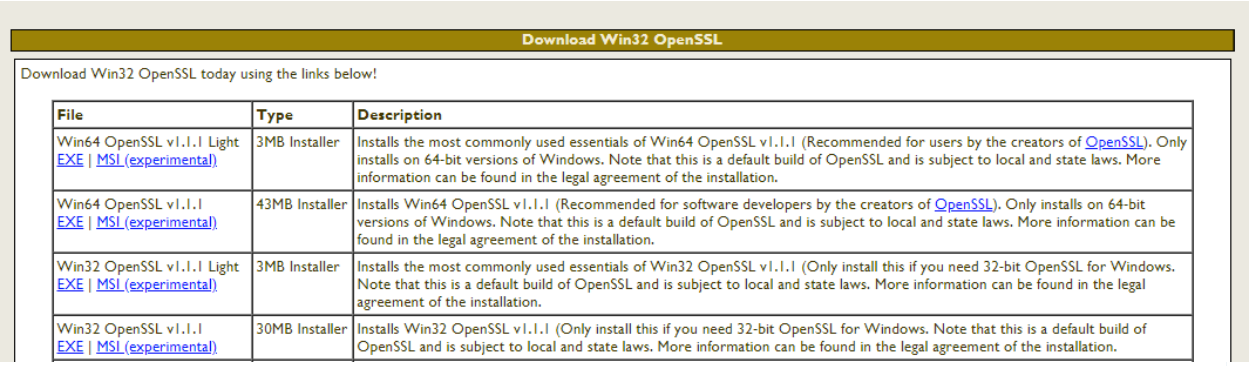
5. เมื่อทำการติดตั้ง credential ที่ได้รับจาก ETDA ลงในระบบของผู้ใช้บริการแล้ว สามารถใช้บริการโดยส่งไฟล์เข้ามาทำ Timestamp ได้ที่ระบบปฏิบัติการจริงตาม URL ที่แจ้งไว้

บทที่ 2 การยืนยันตัวตนผู้ขอใช้บริการด้วยรูปแบบ Certificate Authentication

การเรียกใช้บริการ e-Timestamping จะต้องมีขั้นตอนในการยืนยันตัวตนสำหรับเข้าใช้บริการ โดยในส่วนนี้จะเป็นการอธิบายขั้นตอนในการสร้าง private key และ Certificate Signing Request (CSR) file (เพื่อส่งให้ ETDA ในการลงทะเบียนใบรับรอง) โดยใช้ OpenSSL บน Windows ซึ่งมีขั้นตอนดังต่อไปนี้

ขั้นตอนที่ 2.1: การติดตั้ง OpenSSL

1. เปิดลิงก์ต่อไปนี้ในเบราว์เซอร์ของคุณ https://slproweb.com/products/Win32OpenSSL.html
2. เลื่อนไปที่หัวข้อ **“Download Win32 OpenSSL"**
3. ในคู่มือนี้ เลือกเป็น Win64 OpenSSL v.1.1.1 (ในตัวอย่างเลือกเป็นแบบ 64 บิต ซึ่งอาจเลือกให้ตรงกับคอมพิวเตอร์ของคุณได้ โดยการเลือก 32 บิตหรือ 64 บิต จะมีผลต่อชื่อโฟลเดอร์ ในขั้นตอนที่ 2.2 และ ขั้นตอนที่ 2.3 และขอให้ใช้ edition อื่นๆที่ไม่ใช่ “Light” edition)



Download Win32 OpenSSL

Download Win32 OpenSSL today using the links below!

File	Type	Description
Win64 OpenSSL v1.1.1 Light EXE | MSI (experimental)	3MB Installer	Installs the most commonly used essentials of Win64 OpenSSL v1.1.1 (Recommended for users by the creators of OpenSSL). Only installs on 64-bit versions of Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win64 OpenSSL v1.1.1 EXE | MSI (experimental)	43MB Installer	Installs Win64 OpenSSL v1.1.1 (Recommended for software developers by the creators of OpenSSL). Only installs on 64-bit versions of Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v1.1.1 Light EXE | MSI (experimental)	3MB Installer	Installs the most commonly used essentials of Win32 OpenSSL v1.1.1 (Only install this if you need 32-bit OpenSSL for Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.
Win32 OpenSSL v1.1.1 EXE | MSI (experimental)	30MB Installer	Installs Win32 OpenSSL v1.1.1 (Only install this if you need 32-bit OpenSSL for Windows. Note that this is a default build of OpenSSL and is subject to local and state laws. More information can be found in the legal agreement of the installation.

4. คลิกไปที่ “EXE” เพื่อทำการดาวน์โหลดโปรแกรม OpenSSL v.1.1.1 และทำการติดตั้งโปรแกรมโดยใช้ค่าเริ่มต้น

ขั้นตอนที่ 2.2: การสร้าง Certificate Signing Request (CSR) โดยใช้ OpenSSL บน Windows

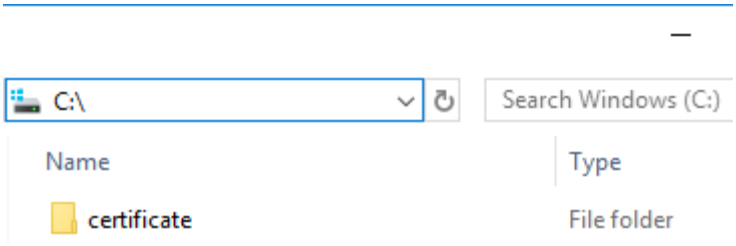
1. คลิกไปที่ปุ่ม **Start** พิมพ์ว่า CMD (Command Prompt)
2. คลิกขวาที่ โปรแกรม CMD และเลือกไปที่ “**Run as Administrator**”
3. จะปรากฏโปรแกรม CMD ขึ้น
4. พิมพ์คำสั่งต่อไปนี้ และกด Enter

   cd C:\Program Files\OpenSSL-Win64\bin
5. จะเห็นว่าบรรทัดที่พิมพ์ เปลี่ยนเป็น **C:\Program Files\OpenSSL-Win64\bin**
6. พิมพ์คำสั่งต่อไปนี้ และกด Enter

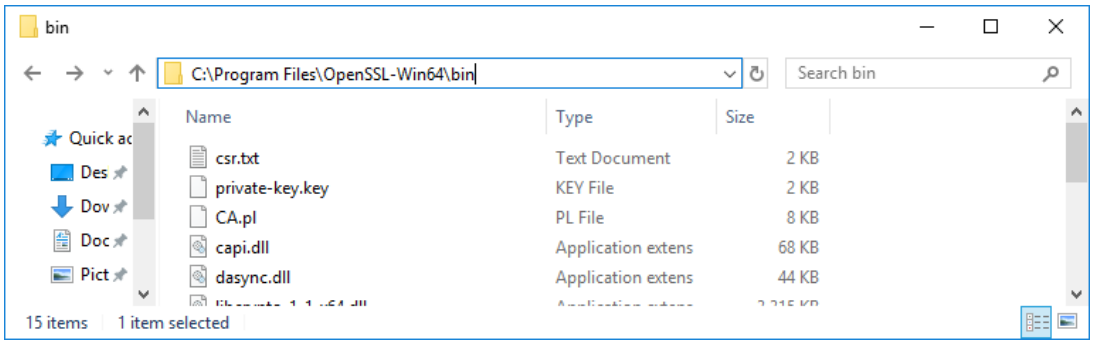
   openssl genrsa -out private-key.key 2048
7. พิมพ์คำสั่งต่อไปนี้ และกด Enter

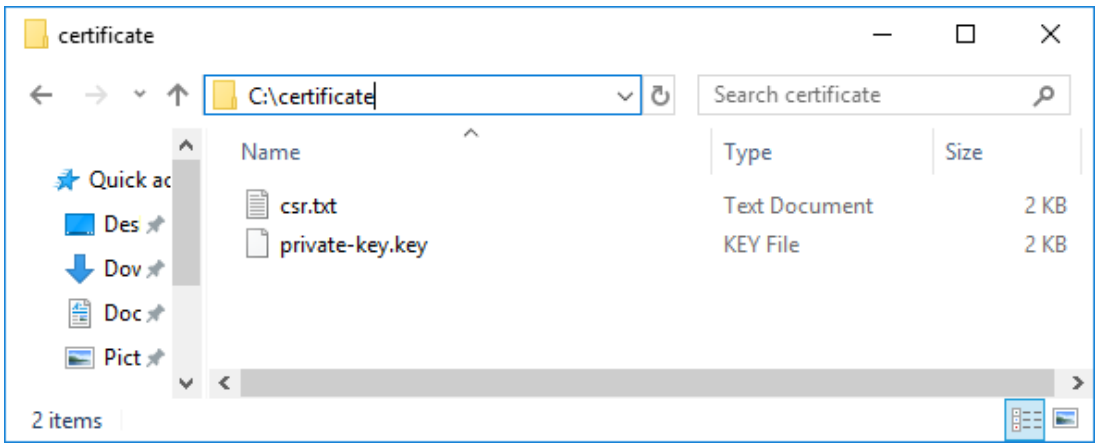
   openssl req -new -key private-key.key -out csr.txt -config cnf\openssl.cnf
8. กรอกข้อมูลองค์กรในฟิลด์ที่ต้องการ:
   - **Country Name:** _____ (ตัวอย่าง: *TH*)
   - **State or Province:** _____ (ตัวอย่าง: *Bangkok*)
   - **Locality or City:** _____ (ตัวอย่าง: *Huai Khwang*)
   - **Company:** _____ (ตัวอย่าง: *A Test Company Ltd.*)
   - **Organizational Unit:** _____ (ตัวอย่าง: *Information Technology Department*)
   - **Common Name:** _____ (ตัวอย่าง: *IT Dept.*)
   - **Email:** _____ (ตัวอย่าง: *itadmin@atest.co.th*)
   - **Challenge Password:** เว้นว่างและกด Enter
   - **Optional company name:** เว้นว่างและกด Enter
9. Private key และ Public key จะถูกสร้างขึ้น โดย Private key (ตัวอย่างในที่นี้ กำหนดชื่อไฟล์เป็น private-key.key) จะถูกสร้างเพื่อใช้สำหรับการถอดรหัส และ Public keys ซึ่งอยู่ในรูปแบบ Certificate Signing Request (ตัวอย่างในที่นี้ กำหนดชื่อไฟล์เป็น csr.txt) จะถูกสร้างเพื่อใช้ในการลงทะเบียนใบรับรอง

10. สร้างโฟลเดอร์ใหม่ (ตัวอย่างในที่นี้ กำหนดชื่อโฟลเดอร์เป็น **certificate**) โดยนำไปไว้ในไดรฟ์ **C**



11. ย้าย private key (ตัวอย่างในที่นี้ ไฟล์อยู่ที่ **C:\Program Files\OpenSSL-Win64\bin\private-key.key**) and CSR file (ตัวอย่างในที่นี้ ไฟล์อยู่ที่ **C:\Program Files\OpenSSL-Win64\bin\csr.txt**) ไปไว้ที่โฟลเดอร์ใหม่ที่เพิ่งสร้างขึ้น (ตัวอย่างในที่นี้ กำหนดเป็น **C:\certificate**)





12. CSR file (ตัวอย่างในที่นี้ กำหนดชื่อไฟล์เป็น csr.txt) ในขณะนี้พร้อมแล้วที่จะทำการลงทะเบียนใบรับรอง
13. ส่ง CSR file ไปให้ ETDA เพื่อทำการลงทะเบียนใบรับรอง

ขั้นตอนที่ 2.3: การรวม certificate กับ private key โดยใช้ OpenSSL บน Windows

1. เมื่อ ETDA ส่งไฟล์ certificate สำหรับ authentication (ตัวอย่างในที่นี้ กำหนดชื่อไฟล์เป็น eservice.cer) และไฟล์ certificate ของ CA ที่ออก certificate สำหรับ authentication (ตัวอย่างในที่นี้ กำหนดชื่อไฟล์เป็น authenCA.cer) กลับมา ให้ย้าย certificate ที่ ETDA ส่งมา ไปไว้ที่โฟลเดอร์เดียวกับที่เก็บ private key (ตัวอย่างในที่นี้ กำหนดเป็น **C:\certificate**)
2. คลิกไปที่ปุ่ม Start พิมพ์ว่า CMD (Command Prompt)
3. คลิกขวาที่ โปรแกรม CMD และเลือกไปที่ “Run as Administrator”
4. จะปรากฏโปรแกรม CMD ขึ้น
5. พิมพ์คำสั่งต่อไปนี้ และกด Enter

   cd C:\Program Files\OpenSSL-Win64\bin
6. จะเห็นว่าบรรทัดที่พิมพ์เปลี่ยนเป็น **C:\Program Files\OpenSSL-Win64\bin**
7. พิมพ์คำสั่งต่อไปนี้ และกด Enter

   openssl pkcs12 -export -in c:\certificate\eservice.cer -certfile c:\certificate \authenCa.cer -inkey c:\certificate\private-key.key -out eservice.p12
8. กรอก Export Password และยืนยัน Export password อีกครั้ง
9. **certificate กับ private key จะถูกรวมกันเป็น keystoreFile (**ตัวอย่างในที่นี้ กำหนดชื่อ keystoreFile **เป็น eservice.p12)**

หมายเหตุ:

1. โปรดเก็บรหัสผ่าน (Export Password/**keystorePassword**) ไว้ในที่ปลอดภัย และเก็บแยกกับ keystoreFile (.P12)

2. หากทำการเรียกใช้บริการ e-Timestamping แล้วพบ ข้อผิดพลาดแจ้ง “could not create SSL/TLS secure channel.” โปรดทำตาม ขั้นตอนที่ 2A.2 การติดตั้ง Trust Root CA กับเครื่องที่ใช้บริการ e-Timestamping

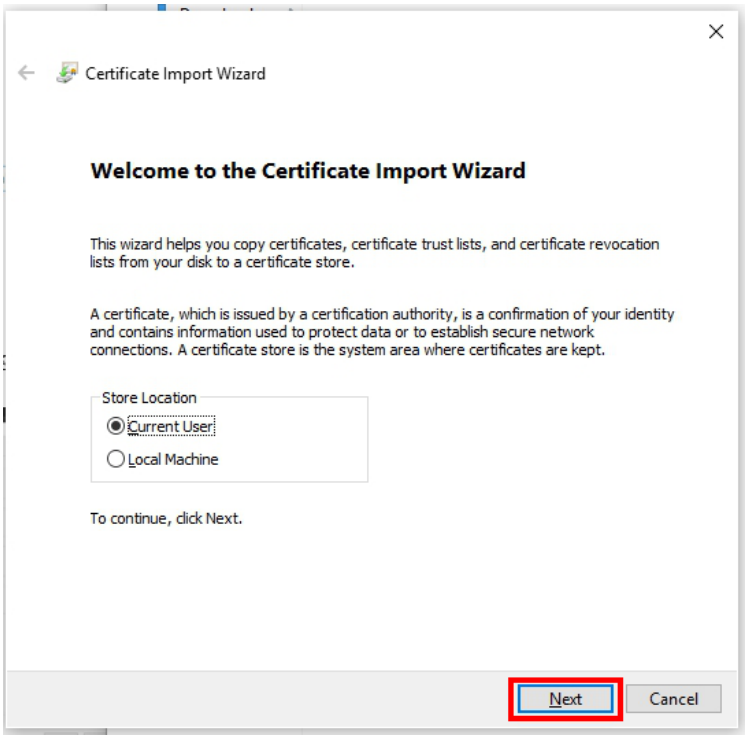
การทดสอบเรียก timestamp ผ่าน Adobe Acrobat Reader DC

KeyStoreFile ที่ได้มาจาก ขั้นตอนที่ 2.3 จะใช้เป็น certificate ในการยืนยันตัวตนผู้ใช้บริการ ในการเรียกใช้ timestamp ไปที่ server ของ ETDA โดยหัวข้อนี้ มีจุดประสงค์เพื่อเป็น quick guide เพื่อทดสอบให้เห็นว่าจาก client เครื่องหนึ่งๆ สามารถเรียกใช้ timestamp ได้อย่างไร ผ่านการเรียกโปรแกรมที่มีการใช้งานแพร่หลายอย่าง Adobe Acrobat Reader DC แต่หากหน่วยงานทำการติดตั้ง Timestamp Proxy Server เอง ให้ข้ามขั้นตอนที่ 2A.1 – 2A.2 ไปขั้นตอนที่ 2A.3 ได้เลย

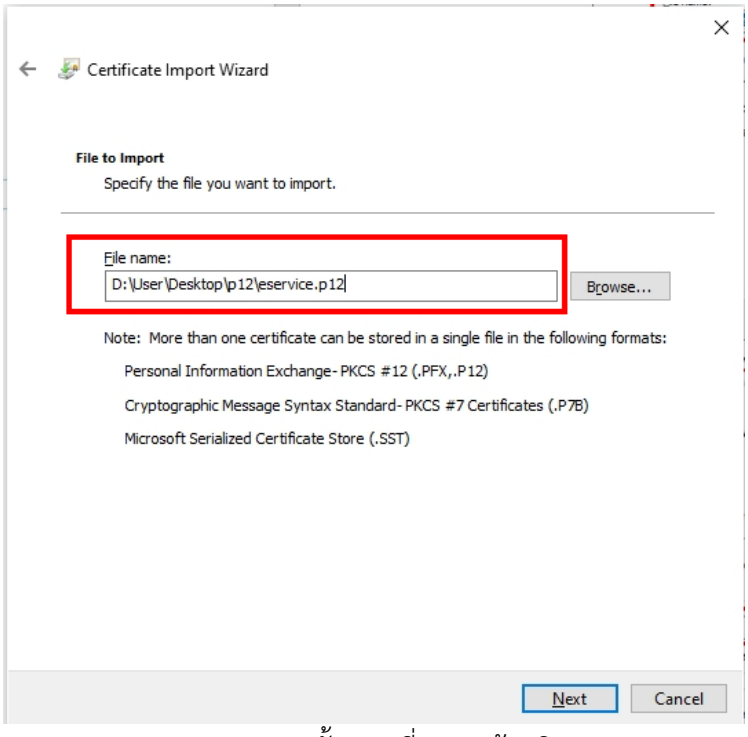
ขั้นตอนที่ 2A.1: การติดตั้ง KeyStoreFile

1. ดับเบิ้ลคลิก KeyStoreFile

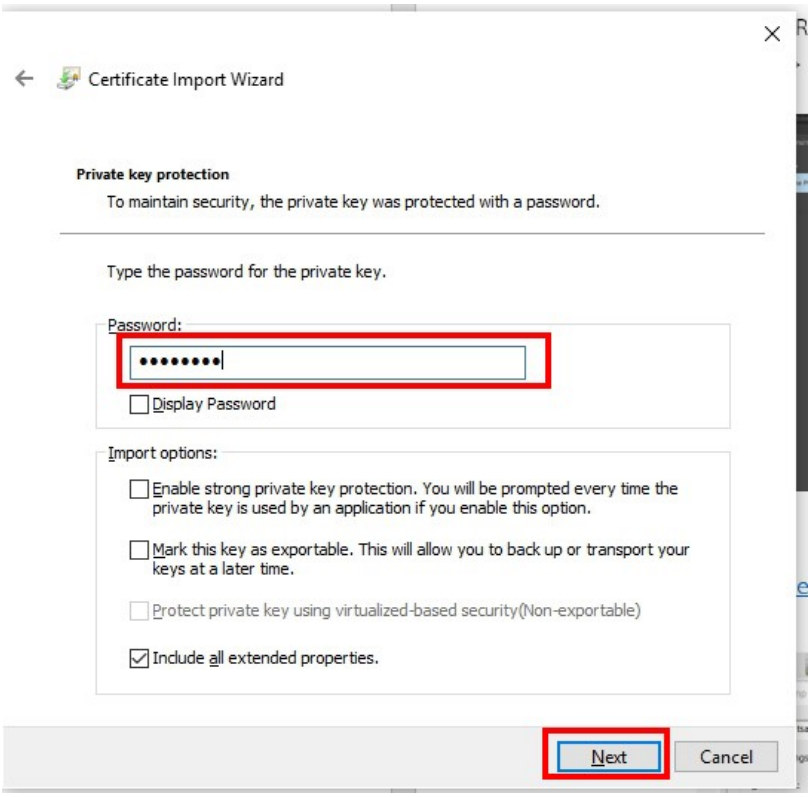
2. คลิก Next



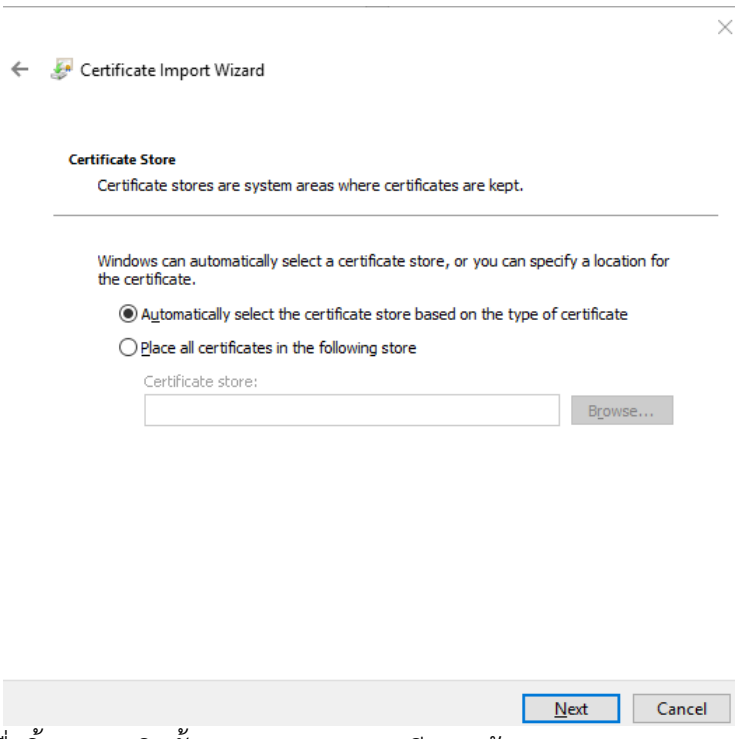
3. ตรวจสอบ Path File (ซึ่งจะแสดงค่าเริ่มต้นเป็น path ที่วางไฟล์นั้นอยู่) แล้วคลิก Next



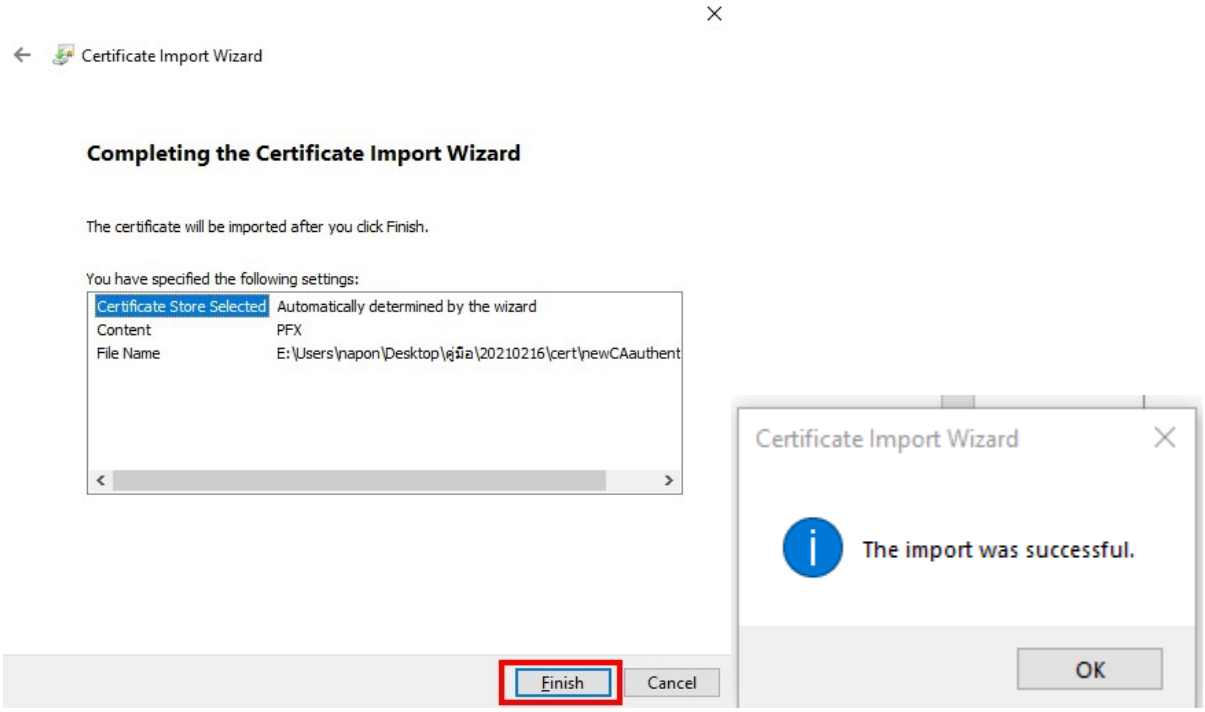
4. กรอกรหัสผ่าน (keystorePassword) จากขั้นตอนที่ 2.3 แล้วคลิก Next



5. คลิก Next



6. คลิก Finish เพื่อสิ้นสุดการติดตั้ง KeyStoreFile จะมีการแจ้ง import success ตามภาพ

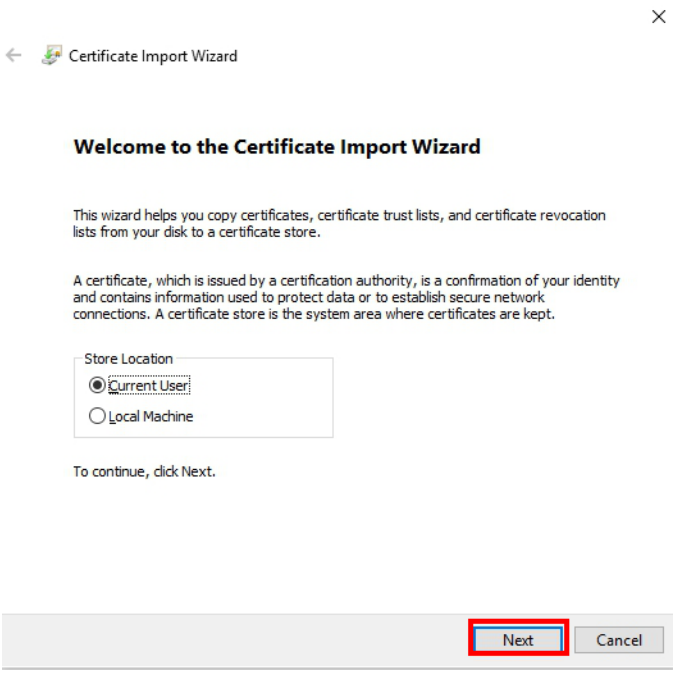


ขั้นตอนที่ 2A.2 : การติดตั้ง Trust Root CA

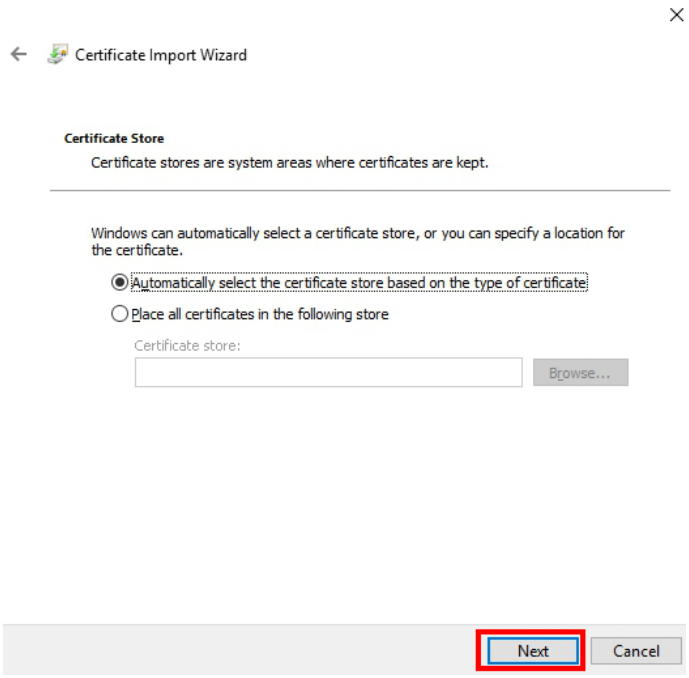
1. ดับเบิ้ลคลิก Root Certificate (ตัวอย่างในที่นี้ กำหนดชื่อ Root Certificate เป็น authenCA.cer)

2. คลิก Install Certificate...

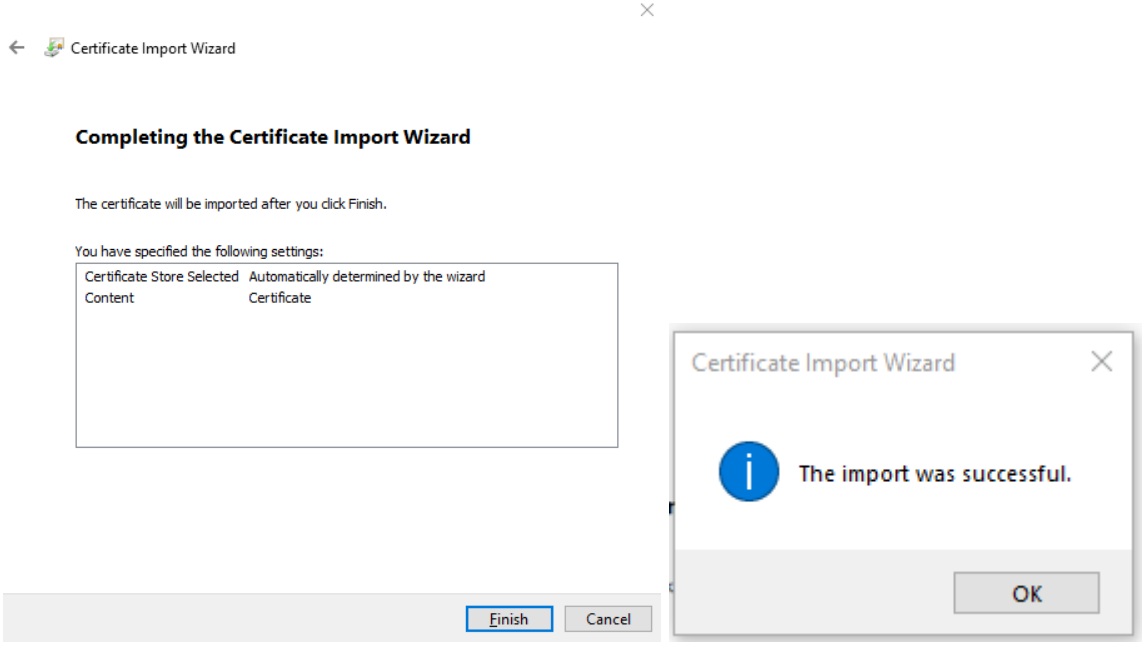
3. คลิก Next



4. คลิก Next

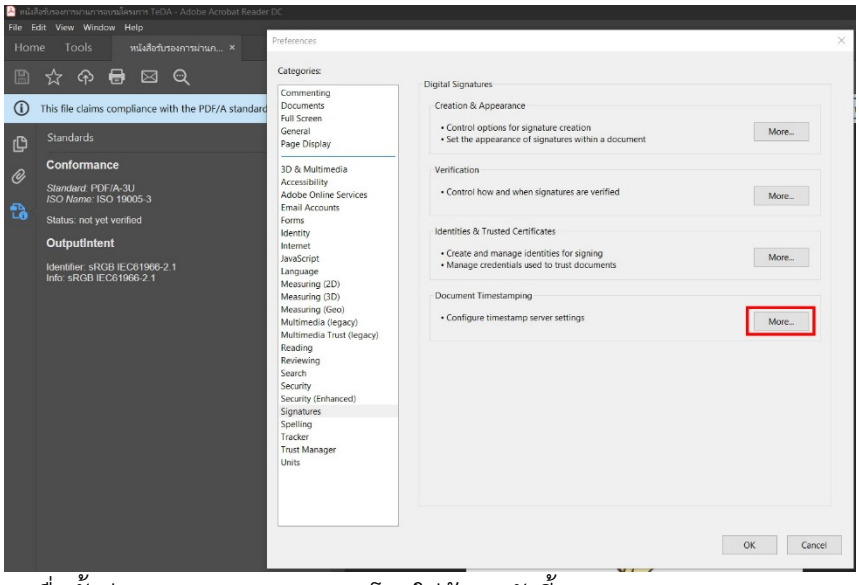


5. คลิก Finish เพื่อสิ้นสุดการติดตั้ง Root Certificate จะมีการแจ้ง import success ตามภาพ



ขั้นตอนที่ 2A.3: การ configure Timestamp ในโปรแกรม Adobe Acrobat Reader DC

1. เปิดโปรแกรม Adobe Acrobat Reader DC
2. ไปที่เมนู Edit -> Preference -> Signatures คลิก More... ที่ Configure timestamp server settings

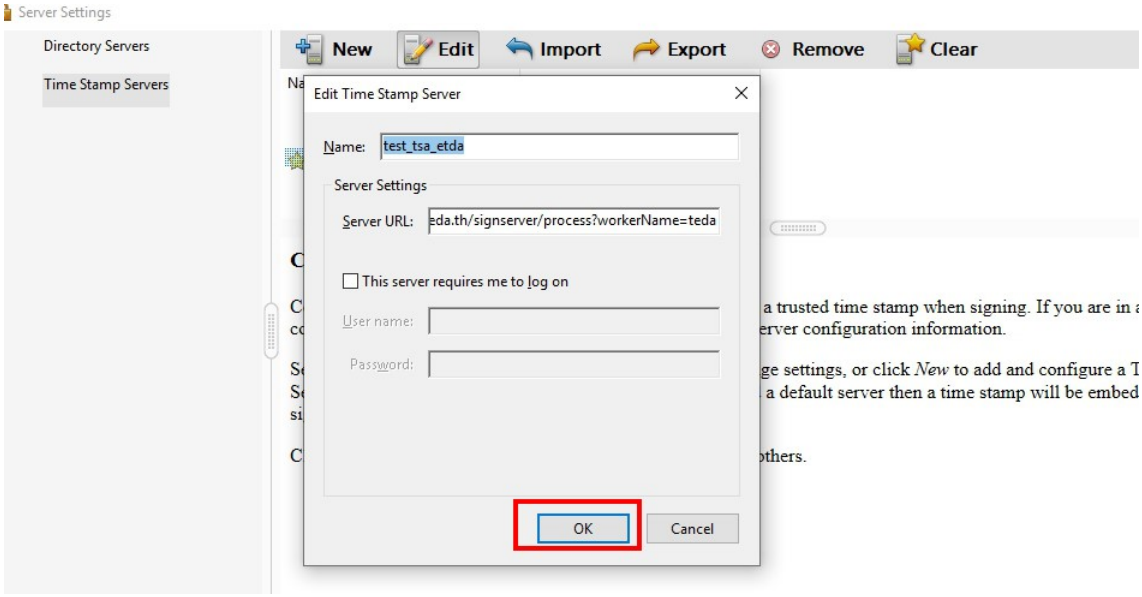


3. คลิก New เพื่อตั้งค่า timestamp server โดยใส่ข้อมูลดังนี้

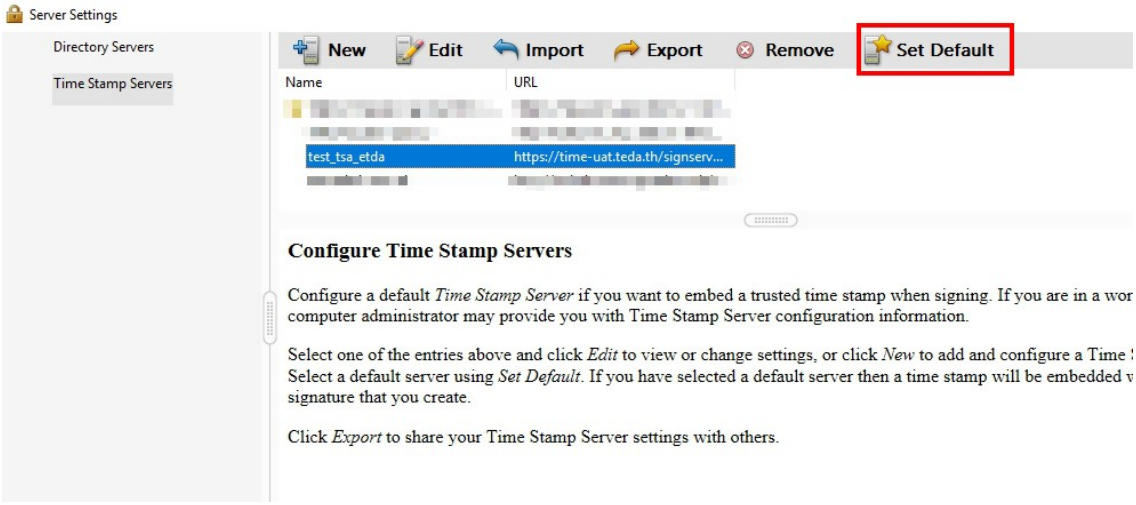
   Name: ตัวอย่างในที่นี้ กำหนดเป็น test tsa etda

   Server URL: ใส่ URL ระบบ timestamp ของ ETDA ตามที่ได้แจ้งไว้ <u>หรือ</u> URL ของ Timestamp Proxy Server ที่หน่วยงานได้ทำการติดตั้งไว้

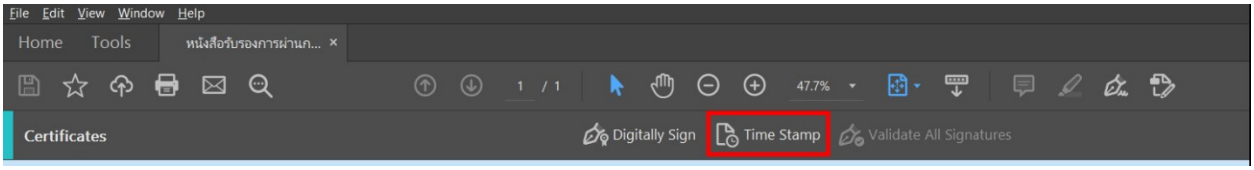
   แล้วคลิก OK



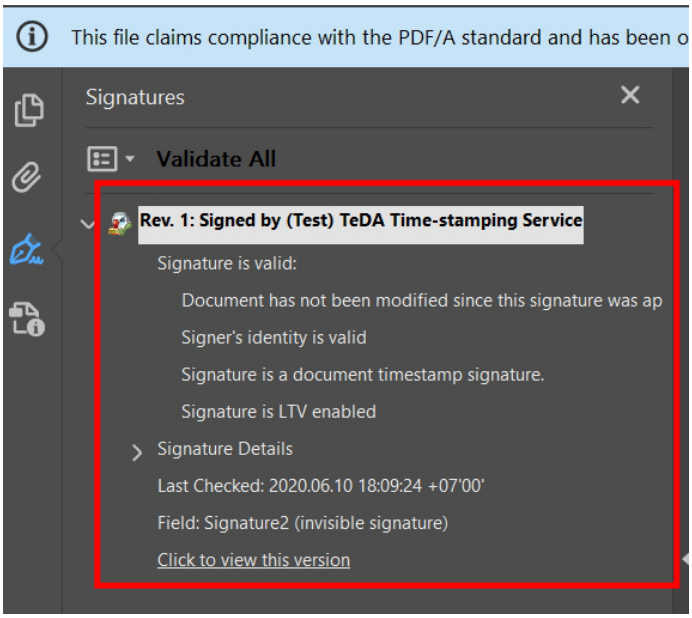
4. เลือกข้อมูล timestamp server จากขั้นตอนก่อนหน้า แล้วคลิก Set Default จะมีรูปดาวขึ้นหน้าชื่อ จากนั้นจึงปิดหน้าตั้งค่า



5. เปิดไฟล์ PDF ที่ต้องการทำ timestamp ไปที่เมนู Tools เลือก icon Certificates จะพบว่ามีแถบเครื่องมือขึ้นมา ให้คลิกเลือก Time Stamp



6. หากมีหน้าต่างให้เลือก Certificate ให้เลือก Certificate ที่ทำการติดตั้งไปในขั้นตอนที่ 2A.1 แล้วทำการบันทึกไฟล์นี้ ด้วยชื่อที่ต่างออกไป เพื่อให้ทราบว่าเป็นไฟล์ที่มีการทำ timestamp เพิ่ม
7. เมื่อเรียก timestamp ได้สำเร็จ ในแถบ Signatures (แถบซ้ายมือ ที่เป็นรูปปากกา) ของเอกสาร จะมีรายละเอียด Timestamp ที่เพิ่งทำไปปรากฏขึ้น

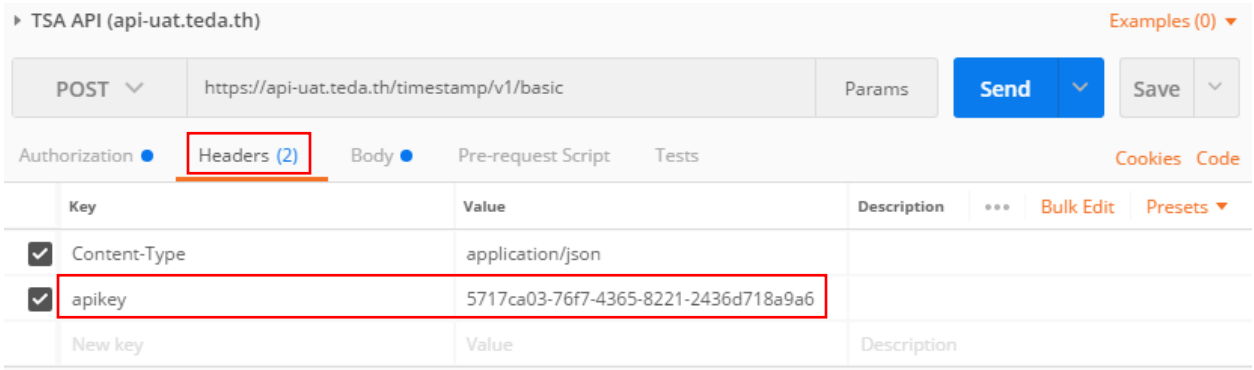


ซึ่งหาก configure timestamp server ในขั้นตอนนี้แล้ว แม้แต่ในการลงลายมือชื่อดิจิทัล ก็จะเรียกใช้เวลาของ timestamp server เป็น signing time (ไม่ใช้เวลาของเครื่องคอมพิวเตอร์ที่เป็น local time) ซึ่งเพิ่มความน่าเชื่อถือให้เอกสารอิเล็กทรอนิกส์ได้ดียิ่งขึ้น

บทที่ 3 การยืนยันตัวตนผู้ขอใช้บริการด้วยรูปแบบ API key

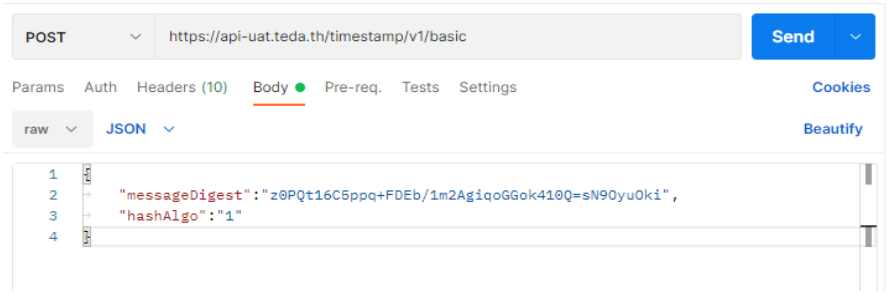
การเรียกใช้บริการ e-Timestamping ในรูปแบบ HTTP API ใช้วิธีการยืนยันตัวตนด้วยรูปแบบ API key สำหรับเข้าใช้บริการ โดยในส่วนนี้จะเป็นการอธิบายขั้นตอนในการใช้งาน API key ที่ได้รับหลังจากยื่นใบคำขอ

1. ระบุค่า API key ลงใน header ของ HTTP request โดยระบุ key = “apikey” และระบุ value = [ค่า API key ที่ได้รับมา] โดยในตัวอย่างแรก จะแสดงการระบุค่า API key ใน tool สำหรับทดสอบเรียก HTTP ที่ชื่อว่า postman และตัวอย่างที่ 2 จะแสดงการระบุค่า API key ใน HttpPost และ HttpClient ซึ่งเป็น library สำหรับเรียก HTTP request ประเภทหนึ่งของภาษา Java




```
1. HttpPost request = new HttpPost(tsaUrl);
2. request.addHeader("content-type", "application/json");
3. request.addHeader("apikey", "5717ca03-76f7-4365-8221-2436d718a9a6");
4. HttpResponse result = httpclient.execute(request);
```

2. เมื่อทำการระบุค่า API key และ parameter ต่างๆ ใน body (รายละเอียดเพิ่ม HTTP Request : Timestamp request) ของ HTTP request เสร็จเรียบร้อยแล้ว จะสามารถเรียกใช้งาน e-Timestamping ในรูปแบบ HTTP API ได้



บทที่ 4 การติดตั้ง Timestamp Proxy Server

เพื่อเป็น proxy server สำหรับจัดการ certificate authentication ของหน่วยงานผู้ใช้บริการ และเป็นตัวกลางในการเรียก timestamp มาที่ ETDA

1. ความต้องการขั้นต่ำของ server CPU = 2 cores, RAM = 4 GB, HDD = 50 GB
2. ติดตั้งด้วย NGINX (version 1.9.0 ขึ้นไป) และระบุ certificate authentication ไว้ที่ NGINX
3. ตั้งค่า NGINX ให้ forward transaction ที่มีการเรียก timestamp มาด้วย port 443 ให้ส่งไปที่ระบบ e-timestamping ของ ETDA

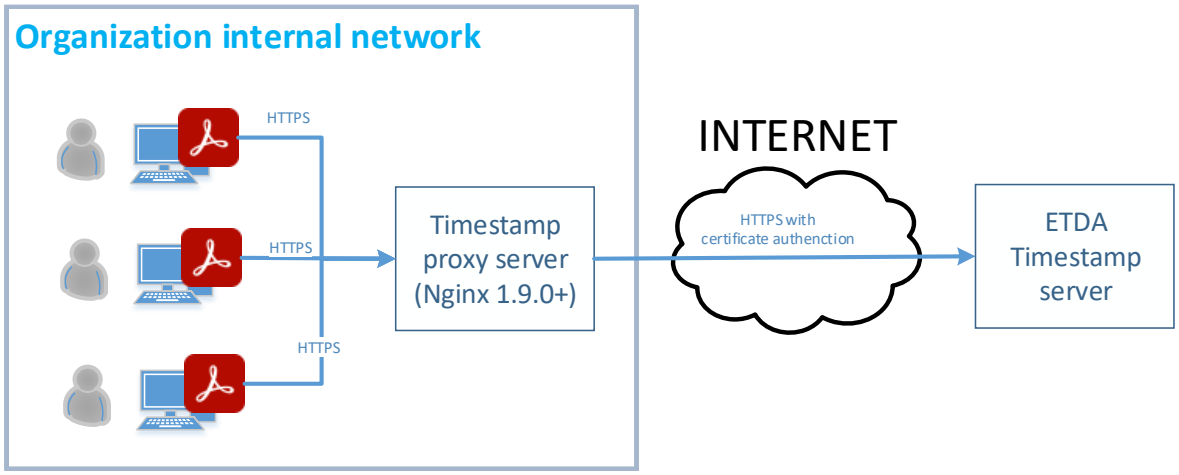
   ตัวอย่าง config nginx

```
location /upstream {
    proxy_pass                  https://backend.example.com;  //url ของ tsa etda
    proxy_ssl_certificate       /etc/nginx/client.pem; //certificate ที่ทาง etda ส่งให้ (PEM formate)
    proxy_ssl_certificate_key   /etc/nginx/client.key; //private key (PEM formate)
    proxy_ssl_protocols         TLSv1 TLSv1.1 TLSv1.2;
    proxy_ssl_ciphers           HIGH:!aNULL:!MD5;
    proxy_ssl_trusted_certificate  /etc/nginx/trusted_ca_cert.crt; //trust root ที่ทาง etda ส่งให้ (PEM formate)

    proxy_ssl_verify    on;
    proxy_ssl_verify_depth    2;
    proxy_ssl_session_reuse   on;
}
```

4. ประกาศ domain name ของ Timestamp Proxy Server ภายใน internal network ของหน่วยงาน โดยไม่ควรตั้งให้สามารถเข้าถึง Timestamp Proxy Server จากอินเตอร์เน็ตได้

High level network diagram



บทที่ 5 การสร้าง Runnable JAR file เพื่อประทับรองรองเวลาไฟล์ PDF

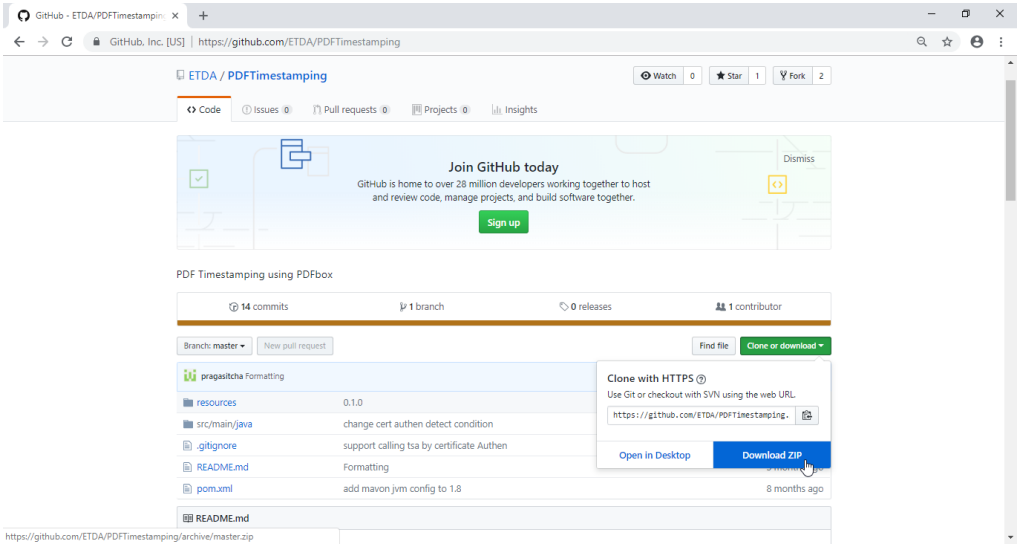
เพื่อเพิ่มความสะดวกในการพัฒนาโปรแกรมร่วมกับภาษาอื่น สามารถเรียกใช้บริการ e-Timestamping ผ่าน Runnable JAR file โดยการนำ JAVA library (project PDFTimestamping) มาทำเป็น Runnable JAR file สำหรับการประกอบ timestamp token ที่ได้จากระบบ กับ PDF ไฟล์ที่เลือก

ความต้องการพื้นฐานบนระบบปฏิบัติการ Windows

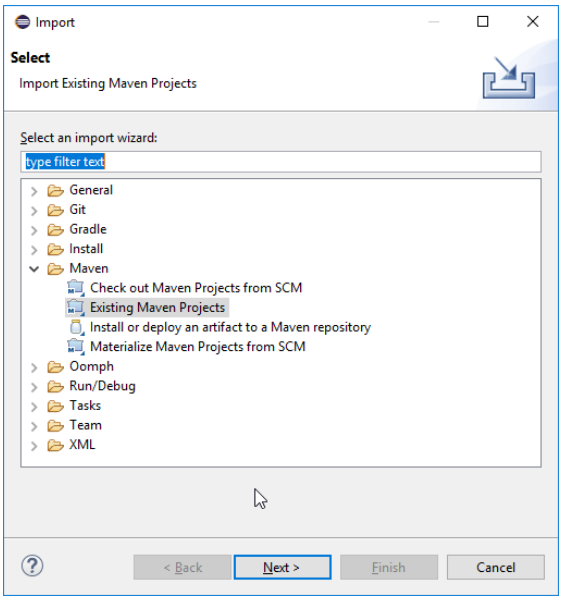
- Java 1.8 ขึ้นไป (https://www.java.com/en/download/)
- โปรแกรม Eclipse Oxygen for java EE version 4.7 ขึ้นไป (https://www.eclipse.org/downloads/packages/)
- ไฟล์ Keystore และ Password Keystore (โดยสามารถทำตามขั้นตอน “การสร้าง private key และ CSR file” ในบทก่อนหน้า)

ขั้นตอนที่ 5.1: การ import project PDFTimestamping

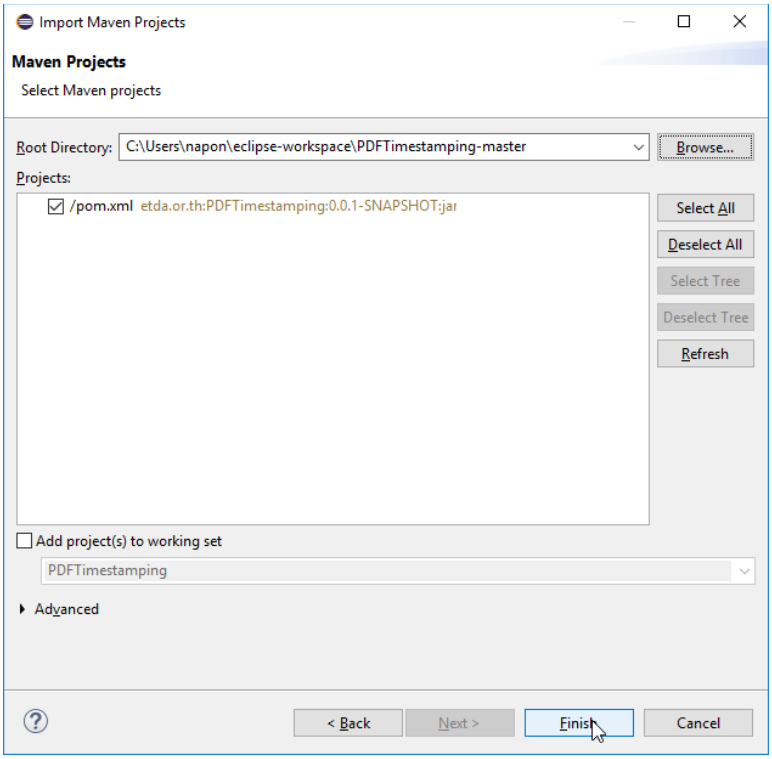
1. เข้าไป Download file project PDFTimestamping จาก GitHub (https://github.com/ETDA/PDFTimestamping) และ unzip file



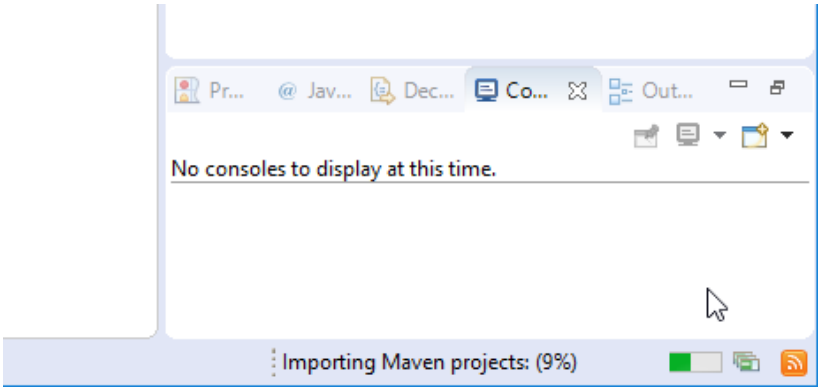
2. เปิดโปรแกรม Eclipse เลือกเมนู File เลือก import เลือกเป็น Maven > Existing Maven Projects



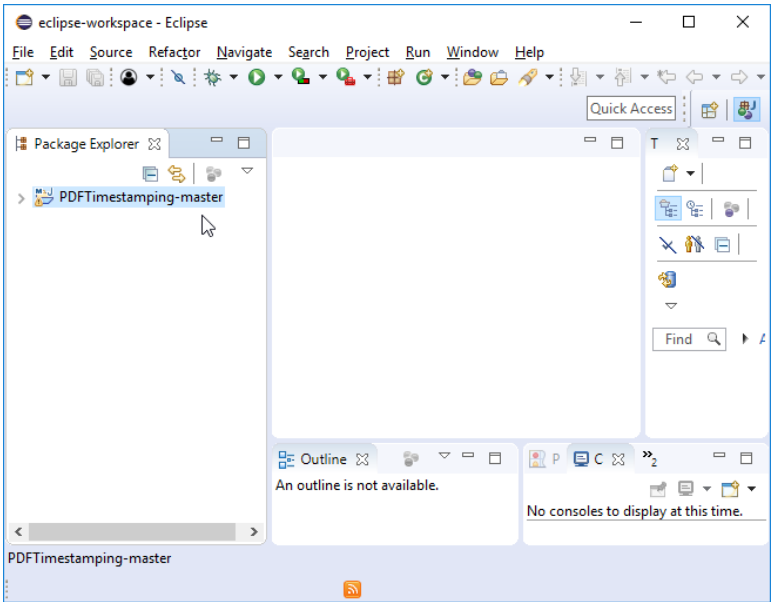
3. กดปุ่ม browse แล้วเลือกโฟลเดอร์ของ project PDFTimestamping ที่ทำการ unzip file แล้ว และทำการกด Finish



4. รอให้ Maven import dependency download เสร็จ



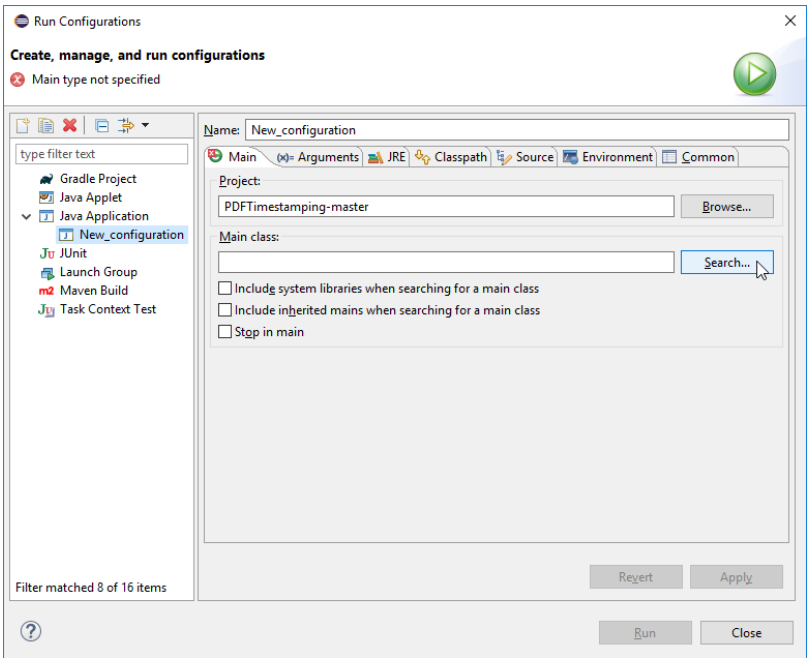
5. จะมี project PDFTimestamping ขึ้นอยู่ด้านข้างของโปรแกรม



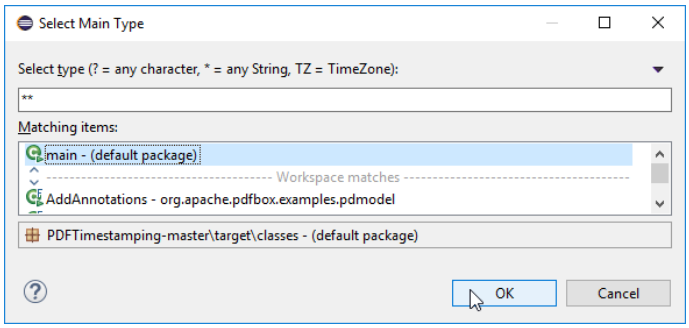
หมายเหตุ: หากไม่สามารถ import project ได้ สามารถเลื่อนลงไปดูในหัวข้อ “การแก้ปัญหา กรณี import project ไม่สมบูรณ์” ได้

ขั้นตอนที่ 5.2: การสร้าง JAR file

1. คลิกขวาที่ project PDFTimestamping เลือก Run As > Run Configurations เลือกแถบ Java Application



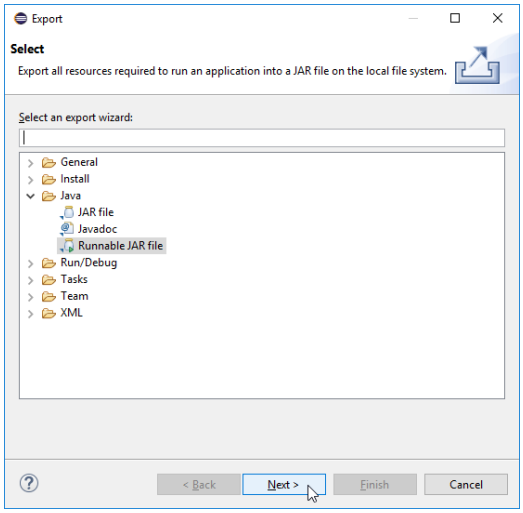
2. ดูตรง Main Class กดปุ่ม search และเลือกเป็น main – (default package)



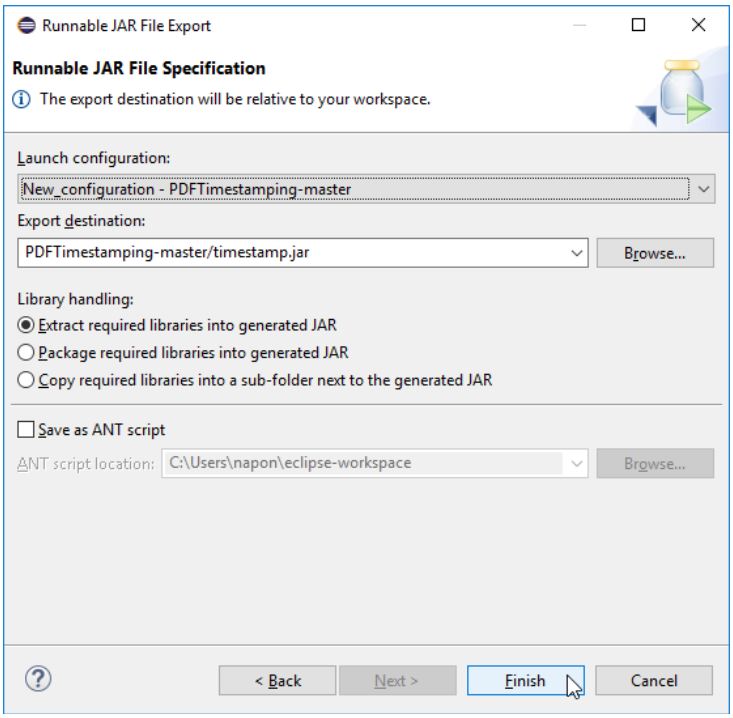
3. จากนั้นกด Apply และ Run

   ตัว Eclipse จะ return error เพราะไม่พบ การตั้งค่า Argument ไม่ต้องสนใจเพราะเราต้องการตัว execute jar file

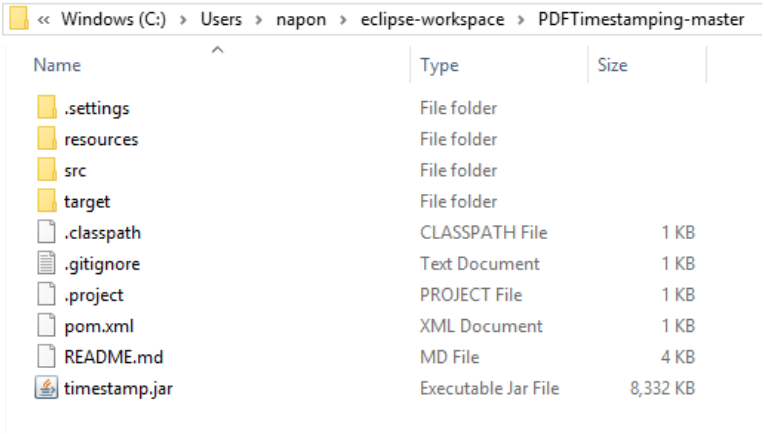
4. คลิกขวาที่ project PDFTimestamping เลือก Export เลือก Java และเลือกไปที่ Runnable JAR file



5. ทำการเลือก Launch configuration ไปเป็น main – PDFTimestamping และกำหนดตำแหน่งที่ต้องการวาง JAR file



6. เมื่อทำการสร้างเสร็จ จะเห็นไฟล์ปรากฏขึ้นในตำแหน่งที่เลือกไว้



ขั้นตอนที่ 5.3: การนำ Runnable JAR file มาใช้

คือ การนำ JAR file ที่ได้มา run โดยสามารถใช้ภาษาต่างๆเช่น C# .net python มาเรียกใช้

ตัวอย่างเช่น

Code C#

```
using System;
using System.Diagnostics;

namespace ConsoleApp2
{
    0 references
    class Program
    {
        0 references
        static void Main(string[] args)
        {
            String CMD = "/c java -jar \"Timestamp.jar\" \"pdfA3.pdf\" \"https://time-uat.teda.th/signserver/process?workerName=teda\" \"Napon.p12\" \"Napon\" \"PKCS12\" \"2\" ";
            Process.Start("CMD.exe", CMD);
        }
    }
}
```

โดย process.Start(“cmd.exe”,CMD) จะเป็นการเรียก cmd.exe และใส่คำสั่งตาม string CMD ซึ่งใน string CMD จะเป็นการเรียก JAR file และใส่ค่า parameter ลงไป คือ

JAR file : ชื่อ JAR file ที่ทำการสร้างออกมา (ตัวอย่างในที่นี้ กำหนดชื่อไฟล์เป็น **timestamp.jar**) ที่ชี้ไปยัง JAR file

inputFile : ชื่อไฟล์ PDF ที่ต้องการทำ Timestamp (ตัวอย่างในที่นี้ กำหนดชื่อไฟล์เป็น **pdfA3.pdf**)

TSAURL : URL ของบริการ TSA

โดยทาง ETDA จะส่ง URL ให้พร้อมกับไฟล์ certificate ทางอีเมล หลังมีการขอใช้บริการหรือขอทดสอบบริการ

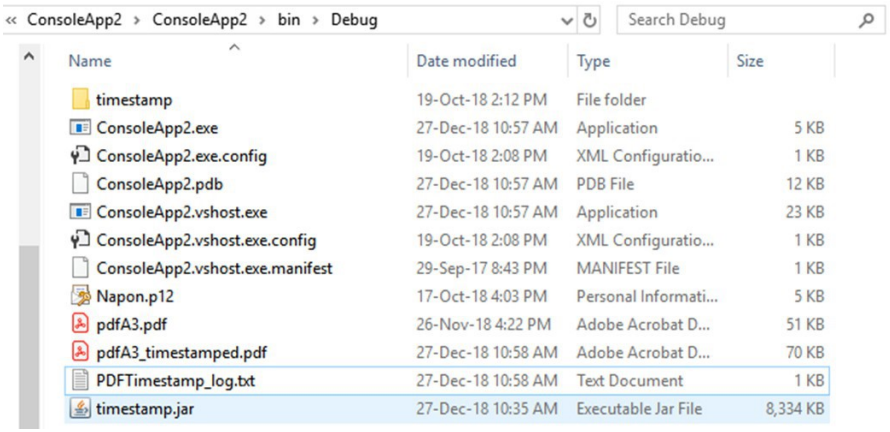
keystoreFile : ชื่อไฟล์ keystore (ตัวอย่างในที่นี้ กำหนดชื่อไฟล์เป็น **Napon.p12**)

keystorePassword : รหัสผ่านของ keystore (ตัวอย่างในที่นี้ กำหนดรหัสผ่านเป็น **Napon**)

keystoreType : ประเภทของไฟล์ keystore (ตัวอย่างในที่นี้ กำหนดประเภทของไฟล์เป็น **PKCS12**)

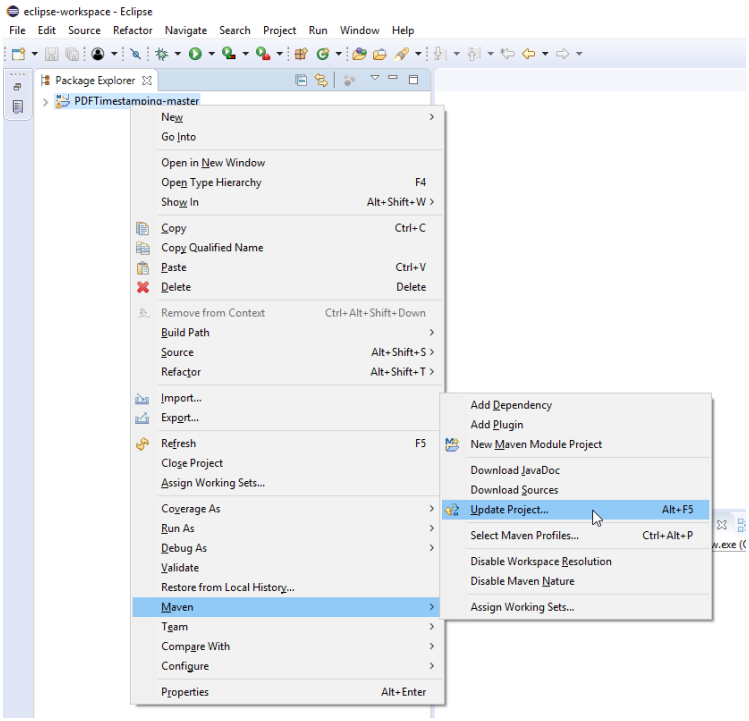
logType : ประเภทของการแสดง log มี 2 ประเภท โดยหากกำหนดเป็น 1 จะเป็นการแสดง log ผ่าน console และหากกำหนดเป็น 2 จะเป็นการแสดง log ผ่านทางไฟล์ PDFTimestamp_log.txt (ตัวอย่างในที่นี้ กำหนดประเภทของการแสดง log เป็น 2)

โดยเมื่อทำการ run เสร็จเรียบร้อย จะได้ output file ที่มีชื่อคล้ายกับ inputFile โดยมีรูปแบบชื่อไฟล์เป็น **ชื่อ inputFile เดิม_timestamped** (ตัวอย่างในที่นี้ จะได้ output file ชื่อ pdfA3_timestamped.pdf) และ log file ชื่อ PDFTimestamp_log.txt ขึ้น

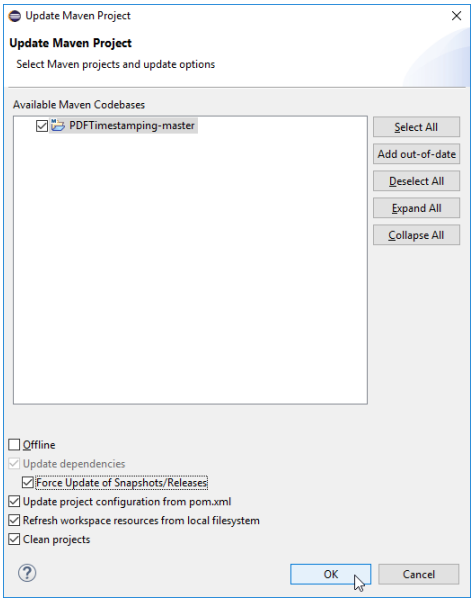


การแก้ปัญหา กรณี **import project** ไม่สมบูรณ์

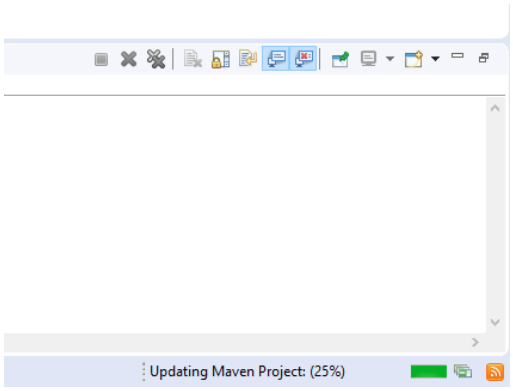
1. คลิกขวาที่ project เลือก Maven > Update project



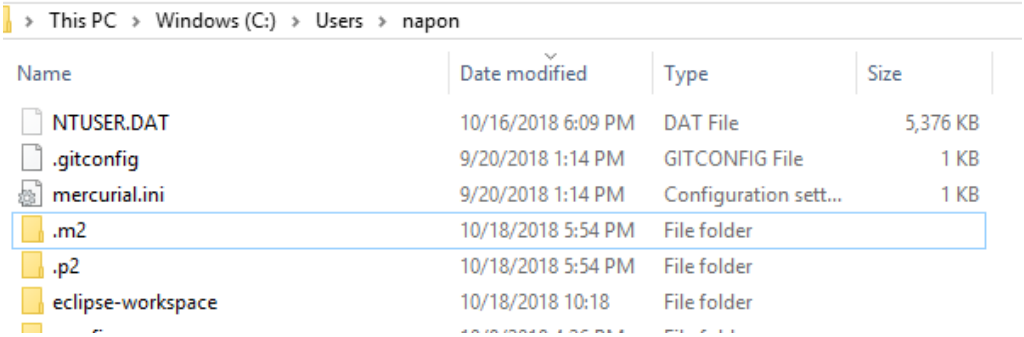
2. คลิกเลือก Force Update of Snapshots/Releases แล้วกด OK



3. รอให้ Maven update จนเสร็จ



4. หากพบว่า**ปัญหาถูกแก้ไขแล้ว ไม่ต้องทำข้อหลังจากนี้** แต่หาก**ยังพบปัญหาขอให้ทำข้อถัดไป**
5. ทำการปิดโปรแกรม Eclipse
6. หาโฟลเดอร์ .m2 ใน drive ที่ทำการลงโปรแกรม Eclipse แล้วทำการลบโฟลเดอร์ .m2 ออก



7. เปิดโปรแกรม Eclipse อีกครั้ง
8. ทำ ข้อที่ 1-3 ใหม่อีกครั้ง

การตรวจสอบ **log file** ที่เกิดจากการ **run JAR File**

หลังจากทำเป็น Runnable JAR File แล้ว เมื่อมีการสั่งรัน JAR File จะทำการอ่านค่าจาก พารามิเตอร์ logType ซึ่งเป็นตัวกำหนด โดยหากกำหนดเป็น 1 จะเป็นการแสดง log ผ่าน console และหากกำหนดเป็น 2 จะเป็นการแสดง log ผ่านทางไฟล์ PDFTimestamp_log.txt

โดยเมื่อกำหนดเป็น 2 แล้วจะทำการตรวจสอบว่ามีไฟล์ PDFTimestamp_log.txt หรือไม่ หากยังไม่มีไฟล์ จะทำการสร้างไฟล์ใหม่ขึ้น แต่หากมีไฟล์อยู่แล้ว จะทำการเขียนไฟล์ต่อจากไฟล์เดิม โดยมีรูปแบบเป็น ปีเดือนวัน เวลา inputFile urlTsa keystoreFile และต่อด้วย ข้อความ log โดยไฟล์ PDFTimestamp_log.txt จะอยู่ที่โฟลเดอร์เดียวกับ Runnable JAR File

ในการเขียน log ลง PDFTimestamp_log.txt แต่ละครั้ง จะเขียน 3 อย่างเพิ่มลงไปคือ

1. วันเวลาในการสั่งรัน
2. ข้อมูล inputFIle, urlTsa, keystoreFile ที่ผู้ใช้ทำการกรอกเข้ามา
3. ข้อความ log ตัวอย่างเช่น

ข้อความ Log	**คำอธิบาย**
********TimeStamp Done**********	สามารถทำ timestamp ได้สำเร็จ
java.io.FileNotFoundException	หาไฟล์ที่ระบุไว้ไม่เจอ
java.io.IOException	มีปัญหากับ input/output
java.lang.StringIndexOutOfBoundsException	ใส่พารามิเตอร์ไม่ครบ
java.net.UnknownHostException	ไม่รู้จักกับ urlTsa ,เชื่อมต่อกับ tsa ไม่ได้
java.security.KeyStoreException	keystore มีปัญหา รหัส keystore ผิด/ไม่ถูกต้อง
java.net.MalformedURLException	ไม่มี urlTsa

หมายเหตุ: ในกรณีที่ไม่มีสิทธิ์ในการเขียน PDFTimestamp_log.txt จะไม่มีการสร้างไฟล์ log หรือเขียนต่อจากไฟล์เดิม แต่จะแสดง log ผ่าน console แทน

ตัวอย่าง PDFTimestamp_log.txt กรณีที่ทำ Timestamp สำเร็จ

Input
"resources/pdfA3.pdf"
"https://time-uat.teda.th/signserver/process?workerName=teda"
"resources/Napon.p12"
"Napon"
"PKCS12"
"2"

Output: PDFTimestamp_log.txt
2018/12/07 10:20:30
inputFile: resources/pdfA3.pdf, tsaUrl: https://time-uat.teda.th/signserver/process?workerName=teda, keystoreFile: resources/Napon.p12
********TimeStamp Done**********

ตัวอย่าง PDFTimestamp_log.txt กรณีที่ทำ Timestamp แล้วเกิดข้อผิดพลาดขึ้น

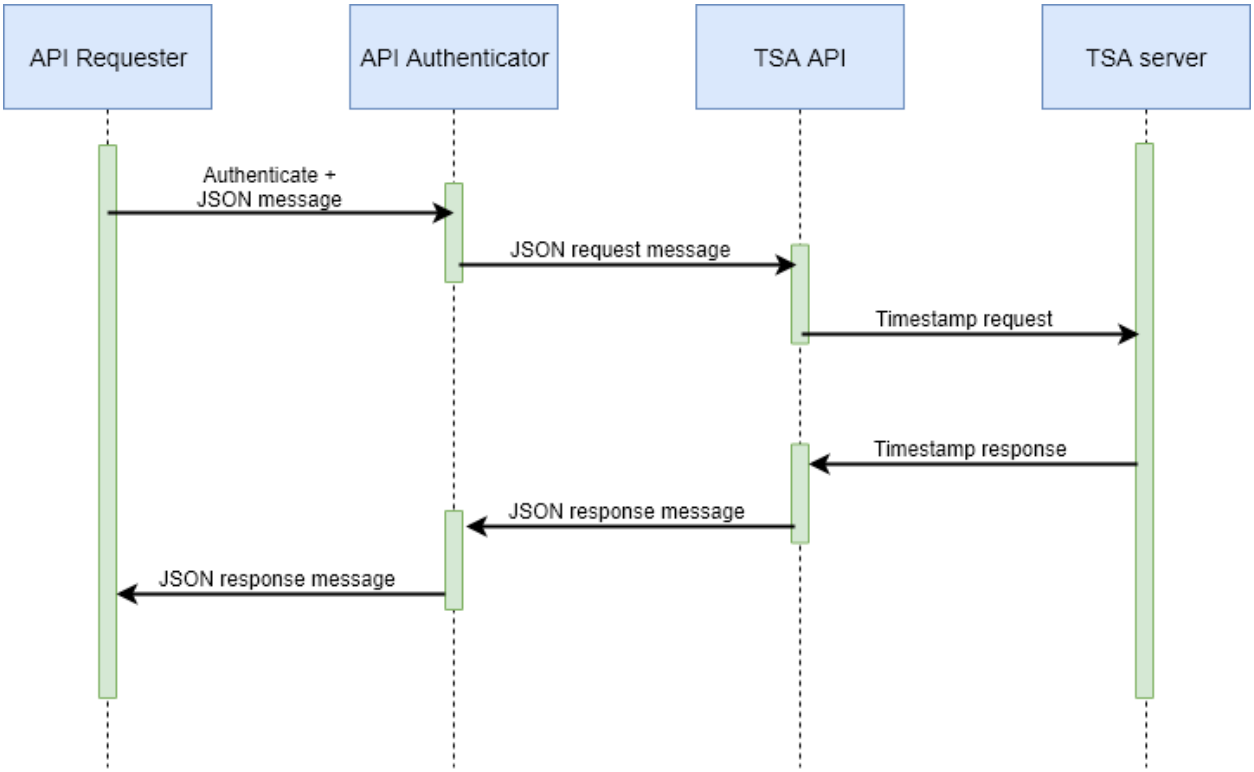
Input
""
"https://time-uat.teda.th/signserver/process?workerName=teda"
"resources/Napon.p12"
"Napon"
"PKCS12"
"2"

Output: PDFTimestamp_log.txt
2018/12/07 10:20:35
inputFile: , tsaUrl: https://time-uat.teda.th/signserver/process?workerName=teda, keystoreFile: resources/Napon.p12
java.lang.StringIndexOutOfBoundsException: String index out of range: -1
at java.lang.String.substring(Unknown Source)
at main.main(main.java:48)

บทที่ 6 Timestamp API Specification

Timestamp API เป็นบริการที่สร้างมาเพื่ออำนวยความสะดวกในการเรียก TSA server สำหรับสร้าง Timestamp Token ตามมาตรฐาน RFC3161 โดยฝั่ง API Requester จะส่ง HTTPS request เข้ามาในระบบ พร้อมกับ parameter ที่จำเป็น ได้แก่ค่า hash ของสิ่งที่ต้องการทำ timestamp และ hash algorithm จากนั้น ระบบจะทำการตอบกลับเป็น Timestamp Token ในรูปแบบ Base64String พร้อมกับค่าสถานะของการทำ timestamp สามารถอธิบายลำดับการทำงานได้ตามรูปด้านล่าง

** HTTPS request จำเป็นต้องแนบ API key สำหรับการ authentication (โดยสามารถทำตามขั้นตอน “การยืนยันตัวตนผู้ขอใช้บริการด้วยรูปแบบ API key” ในบทก่อนหน้า



HTTP Request : Timestamp request

Request HTTP Type:

HTTP Type	Description
POST	POST method สำหรับการส่ง hash ของสิ่งที่ต้องการทำ timestamp

Request URL:

URL
https://[URL]/timestamp/[version]/[endpoint]

Request HTTP Header:

HTTP Header	Value
Content-Type	application/json
apikey	[ค่า API key ที่ได้รับมาจาก ETDA]

Request HTTP Body:

Parameter	Type	Description	M/O
messageDigest	Text (Base64)	hash ของสิ่งที่ต้องการทำ timestamp โดยจะต้องสร้างด้วย hash algorithm เดียวกันกับ parameter hashAlgo	Mandatory
hashAlgo	Text	ค่า hash algorithm โดยสามารถใส่ค่าได้ดังนี้ 0 : “SHA-1” 1 : “SHA-256” 2 : “SHA-512”	Mandatory

HTTP Status Codes:

Code	Description
200 OK	กรณีที่ใส่ค่ามาครบถูกต้อง
400 Bad Request	กรณีที่ใส่ค่า parameter ผิด หรือไม่ครบ เช่นการ request body มาไม่ตรง format ที่กำหนด
401 Unauthorized	กรณีไม่ได้ระบุ certificate authentication
403 Forbidden	กรณีที่ไม่ได้ลงทะเบียนการใช้งานระบบ API
404 Not Found	กรณีที่ใส่ URL ผิด
405 Method Not Allowed	กรณีที่ใส่ HTTP Method ผิด
500 Internal Server Error	กรณีข้อผิดพลาดอื่นๆ

Example Request:

URL
https://api-test.teda.th/timestamp/v1/basic

BODY

{
"messageDigest":"+FDEb/1m2AgiqosN9Oyuz0PQt16C5ppqOkiGGok410Q=",
"hashAlgo":"1"
}

HTTP Response : Timestamp response

Response HTTP Header:

HTTP Header	Value
Content-Type	application/json;charset=UTF-8

Response HTTP Body:

Parameter	Type	Description	M/O
resultCode	Text	ค่าสถานะการทำ Timestamp สามารถดูได้จากตาราง Result code and Result Message	Mandatory
token	Text (Base64)	Timestamp token ตามมาตรฐาน RFC3161 ถ้าหาก resultCode แสดงผลว่าพบ error token จะมีค่าเป็น null	Optional
resultMessage	Text	Return message ของค่า resultCode ถ้าหาก result code แสดงผล เป็นค่า P1004 – P1007 API จะ return ค่า ID มาในฟิลด์นี้ เพื่ออ้างอิงการตรวจสอบ transaction นั้นในระบบ	Mandatory

Result code and Result message

resultCode	resultMessage	Description
P1000	Success	สามารถสร้าง Token ได้สำเร็จ
P1001	Request hash is null	parameter messageDigest มีค่าเท่ากับ null หรือ parameter name ไม่ถูกต้อง
P1002	Request hash algorithm is null	parameter hashAlgo มีค่าเท่ากับ null หรือ parameter name ไม่ถูกต้อง
P1003	Input hash algorithm support only 0,1,2 (for SHA-1,SHA-256,SHA-512)	ใส่ค่า hashAlgo อื่นๆ นอกเหนือจากค่า 0,1,2
P1004	Input hash is not in Base64 format	Input messageDigest ไม่อยู่ในรูปแบบของ Base64String
P1005	An error occurred in TSA API.Please contact admin with transaction ID	พบ error ในส่วนของ TSA API เช่น ตั้งค่า URL ของ TSA server ไว้ผิดรูปแบบ
P1006	Response timestamp token is null	TSA server มี response message ตอบข้อความกลับมาที่ TSA API แต่ timestamp token ภายใน response message จะมีค่าเป็น null เมื่อ TSA API return resultCode เป็น P1006 TSA API จะแนบ message ที่ถูกส่งมาจาก TSA server เพิ่มเติมด้วย
P1007	TSA server connection timeout	ไม่สามารถติดต่อ TSA server ได้ ภายในเวลาที่กำหนดไว้ (ปัจจุบันตั้งค่าไว้ที่ 10 วินาที)
P9999	Unknown error	พบข้อผิดพลาดอื่นๆ เมื่อ TSA API return resultCode เป็น P9999 TSA API จะแนบ message ที่พบจาก exception เพิ่มเติมด้วย

Example Request and Response:

Example 1 : Valid request

Request Body: P1000


```
{
        "messageDigest":"+FDEb/1m2AgiqosN9Oyuz0PQt16C5ppqOkiGGok410Q=",
        "hashAlgo":"1"
}
```


Response Body: P1000


```
{
    "resultCode": "P1000",
    "token": "MIAGCSqGSIb3DQEHAqCAMIIQ9QIBA.....bKOQLFtY6N77xmRt+lUZPlSQAAAAA=",
    "resultMessage": "Success"
}
```


Example 2: Request hash is null

Request Body: P1001


```
{
        "messageDigest":null,
        "hashAlgo":"1"
}
```


Response Body: P1001


```
{
    "resultCode": "P1001",
    "token": null,
    "resultMessage": "Request hash is null"
}
```

Example 3: Request hash algorithm is null

Request Body: P1002

{
"messageDigest":"+FDEb/1m2AgiqosN9Oyuz0PQt16C5ppqOkiGGok410Q=",
"hashAlgo":null
}

Response Body: P1002

{
"resultCode": "P1002",
"token": null,
"resultMessage": "Request hash algorithm is null"
}

Example 4 : Invalid hash algorithm

Request Body: P1003

{
"messageDigest":"+FDEb/1m2AgiqosN9Oyuz0PQt16C5ppqOkiGGok410Q=",
"hashAlgo":"99"
}

Response Body: P1003

{
"resultCode": "P1003",
"token": null,
"resultMessage": "ID:3d074e0900cefd77b21546f7898e405e1541576287382,Message:Input hash algorithm support only 0,1,2 (for SHA-1,SHA-256,SHA-512)"
}

Example 5 : Input hash is not Base64String format

Request Body: P1004


```
{
        "messageDigest":"aaa",
        "hashAlgo":"1"
}
```


Response Body: P1004


```
{
    "resultCode": "P1004",
    "token": null,
    "resultMessage": "ID:714d1e81f5df788a000af4a11a7f4ece1541495691180,Message:Input hash is not in Base64 format"
}
```


Example 6 : Internal TSA API error

เช่น config ให้ TSA API เรียก TSA Server ด้วย URL ที่ผิด format

Request Body: P1005


```
{
        "messageDigest":"+FDEb/1m2AgiqosN9Oyuz0PQt16C5ppqOkiGGok410Q=",
        "hashAlgo":"1"
}
```


Response Body: P1005


```
{
    "resultCode": "P1005",
    "token": null,
    "resultMessage": "ID:64d8170196c1ac29a7d7309a04b561f61541496549424,Message:An error occurred in TSA API.Please contact admin with transaction ID"
}
```

Example 7: Response from TSA server is null

Request Body: P1006

{
"messageDigest":"+FDEb/1m2AgiqosN9Oyuz0PQt16C5ppqOkiGGok410Q=",
"hashAlgo":"0"
}

Response Body: P1006

{
"resultCode": "P1006",
"token": null,
"resultMessage":
"ID:41c82709e34ca448d2e546bb274ebd2d1541496303142,Message:Response timestamp token is null,Cause:null,Exception:[Error-45008] Failed to validate request - hash length in TSA request does not match with hash algorithm"
}

Example 8: TSA server connection timeout

เช่น config ให้ TSA API เรียก TSA Server ด้วย URL ที่ไม่ถูกต้อง ทำให้เกิด TSA Timeout ที่ 10 วินาที

Request Body: P1007

{
"messageDigest":"+FDEb/1m2AgiqosN9Oyuz0PQt16C5ppqOkiGGok410Q=",
"hashAlgo":"1"
}

Response Body: P1007

{
"resultCode": "P1007",
"token": null,

```
    "resultMessage": "ID:ad1ba9f841aeefe7b04053c708f40e091541496943758,Message:TSA
     server connection timeout"
}
```


Example 9: Unknown error

Request Body: P9999

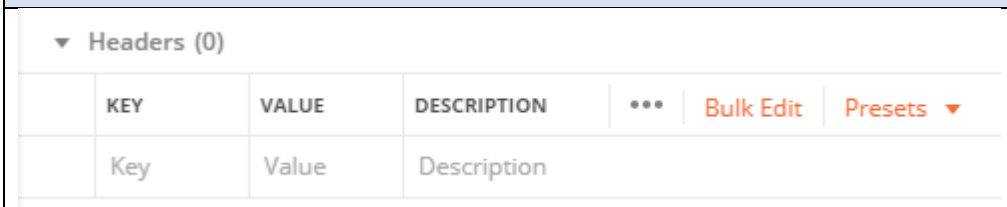

```
{
        "messageDigest":"+FDEb/1m2AgiqosN9Oyuz0PQt16C5ppqOkiGGok410Q=",
        "hashAlgo":"1"
}
```


Response Body: P9999


```
{
    "resultCode": "P9999",
    "token": null,
    "resultMessage": "ID:93d0d8525249fc3cf24952b7be6e8d2f1541680732734,Message:Unknown
error,Cause:null,Exception:null"
}
```


Example 10: No API key

Request Header: No API key



Response Body: No API key


```
{
    "message": "No API key found in request"
}
```

Example 11: Invalid API key

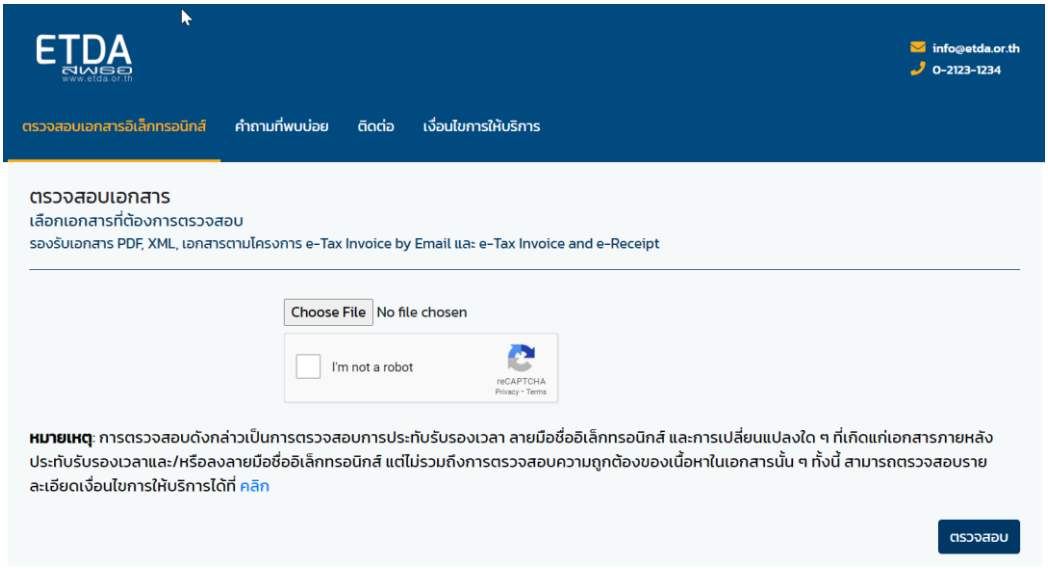
Request Header: Invalid API key

Response Body: Invalid API key

```
{
    "message": "Invalid authentication credentials"
}
```

บทที่ 7 การตรวจสอบความถูกต้องของการประทับรับรองเวลา ด้วย TEDA Web Validation Portal

1. เปิดลิงก์ต่อไปนี้ในเบราว์เซอร์ของคุณ https://validation.teda.th
2. เลือกไฟล์เอกสารที่จะทำการตรวจสอบ และทำการติ๊กที่ช่องสี่เหลี่ยม ทำ process reCAPTCHA และกดปุ่มตรวจสอบ



3. เลื่อนลงมาที่หัวข้อ “E TIMESTAMP” เพื่อดูรายละเอียดในของการทำ timestamp *ด้วยใบรับรองอิเล็กทรอนิกส์ที่ระบบบรองรับ* บนไฟล์เอกสารที่ทำการตรวจสอบ

PDF E-Timestamp ✔

ผลการตรวจสอบการประทับรับรองเวลา (Timestamp Validation Result)	✔ น่าเชื่อถือ
หน่วยงานผู้ประทับรับรองเวลา (Organization Name)	Electronic Transactions Development Agency (Public Organization)
ผู้ประทับรับรองเวลา (Timestamping Authority)	TeDA Time-Stamping Service G3
ผู้ออกใบรับรอง (Certification Authority)	DigiCert SHA2 Assured ID Timestamping CA
วันที่ประทับรับรองเวลา (Timestamping Date)	18 ส.ค. 2564 15:49:16 น. (เวลาประเทศไทย)
วันออกใบรับรอง (Certificate Create Date)	12 มิ.ย. 2563 07:00:00 น. (เวลาประเทศไทย)
วันหมดอายุใบรับรอง (Certificate Expiration Date)	11 มิ.ย. 2566 19:00:00 น. (เวลาประเทศไทย)
สถานะ (Status)	การประทับรับรองเวลามีความน่าเชื่อถือ
ผลการตรวจสอบ LTV (LTV Result)	-
ผลการตรวจสอบ LTA (LTA Result)	-

ภาคผนวก

ตัวอย่างหนังสือขอใช้บริการ

เรื่อง ขอความอนุเคราะห์ใช้งานระบบ e-Timestamping
เรียน ผู้อำนวยการสำนักงานพัฒนาธุรกรรมทางอิเล็กทรอนิกส์
สิ่งที่แนบมาด้วย แบบฟอร์มการสมัครขอใช้บริการประทับรับรองเวลา

ด้วย....ชื่อหน่วยงาน..... มีนโยบายในการนำระบบ Electronic Document มาใช้ให้บริการ เพื่อเพิ่มความสะดวกวดเร็ว ในการรับรองเอกสาร สัญญา หรือ ธุรกรรมอื่นใด ที่จะต้องมีการลงประทับรับรองเวลา เพื่อเป็นหลักฐานในการรับรองเอกสารว่าน่าเชื่อถือได้

ในการนี้ เพื่อให้การบริการเป็นไปด้วยความเรียบร้อย จึงขอความอนุเคราะห์การใช้งานระบบ e-Timestamp บนเอกสาร เพื่อสร้างความมั่นใจให้กับผู้ใช้เอกสาร รวมถึงหน่วยงานที่เกี่ยวข้อง สามารถตรวจสอบการเปลี่ยนแปลงใดๆ และความถูกต้องของเอกสารได้

จึงเรียนมาเพื่อโปรดพิจารณาให้ความอนุเคราะห์ด้วย จะเป็นพระคุณยิ่ง

End of document