Re:FreeXAdES から始める長期署名

～オープンソースでタイムスタンプを使おう編～

2018-05-23
Naoto Miyachi（miyachi@langedge.jp）

Q:何故「Re:」が付いているの?

A:2年前の春祭りでFreeXAdESを
公開したのに放置していたものを
再開したから…です…(^^;

まず前回(2年前)のおさらい

「第1回 ～XAdES-BES編～」

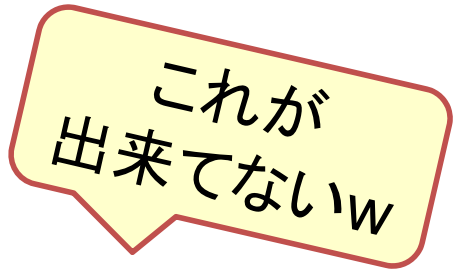
FreeXAdES

入門/勉強用にシンプルなXAdESを実装

※ 高度な機能が必要なら弊社製品版のご検討を…(^^;

Java標準機能で実装（他に依存しない）

- 簡単でシンプルに**使える**こと。
- ただしJava8以降の機能が必要。
  Base64(Java8), XMLSignature(Java6)



~~**XAdESレベル毎に勉強会で説明して行く**~~

MPL v2.0 (Mozilla Public License)で公開

http://mozilla.org/MPL/2.0/

- 商用利用も可能です。

MPL v2 ライセンス

ソース公開義務	GPL	MPL	BSD
OSS本体への修正/追加分	○ 公開必須	○ 公開必須	× 公開不要
OSSを利用したプログラム	○ 公開必須	× 公開不要	× 公開不要

私の理解：間違っていたらご指摘ご指導をm(_ _)m

- FreeXAdESを使うプログラム/システムはソース公開義務無し
- FreeXAdES自体を修正したら**修正部はソース公開義務あり**

※ GitHubで修正分を**プルリクエスト**してください!

XML署名と長期署名XAdES

注:XML署名済みファイルをXAdES化はできない！

機能	XML署名	XAdESレベル
デジタル署名	○	○ XAdES-B (XAdES-BES)
署名証明書保護	▲ ※1	
署名時刻証明	× TS使えない	○ XAdES-T
検証情報保持	▲ ※2	○ XAdES-LT (XAdES-X Long)
長期保管 (長期署名)	× TS使えない	○ XAdES-LTA **(XAdES-A)**

※1 KeyInfo を参照（Reference）追加すれば可能。
※2 証明書認証パスの証明書群は KeyInfo の下に格納可能。

XAdESのXML構造例

XML署名要素に
XAdES要素を追加
することでXAdES化

Signature（XmlDsig：ルート）
- **SignedInfo**（XmlDsig：署名情報）
  - **Reference** URI=“#Sign-Target”（XmlDsig：署名対象参照）
  - **Reference** URI=“#XAdES-Sign-Atrb”（XmlDsig：**XAdES参照**）
- **SignatureValue**（XmlDsig：**署名値**）
- **KeyInfo**（XmlDsig：鍵/証明書情報）
- **Object**（XmlDsig：**XAdESオブジェクト**）
  - **QualifyingProperties**（XAdES：属性情報）
    - **SignedProperties** Id=“XAdES-Sign-Atrb”（XAdES：**署名属性領域**）
    - **UnsignedProperties**（XAdES：**非署名属性領域**）
      - **UnsignedSignatureProperties**（XAdES）
        - **SignatureTimeStamp**（XAdES：**署名タイムスタンプ**）
- **Object** Id=“Sign-Target”（XmlDsig：署名対象）

ハッシュ値

今回の追加要素

以上で「おさらい」は終了。

XML署名/XAdESの基本部であるXAdES-Bについては前回の資料を参照してください。

http://eswg.jnsa.org/matsuri/201605/20160523-S4-miyachi.pdf

さて今回のテーマは、

オープンソースで
タイムスタンプを使おう編

長期署名とタイムスタンプ(RFC 3161)

XAdESレベル	機能（長期署名の内容）
XAdES-B	署名のみ（前回の FreeXAdES β1版）
XAdES-T	**署名タイムスタンプ**追加（β2版） ※ 署名値の保護と署名時刻の証明
XAdES-LT	検証情報（証明書と失効情報）追加
XAdES-LTA （XAdES-A）	**アーカイブタイムスタンプ**追加 ※ 長期保管：直前までの内容の保護

長期署名のタイムスタンプは目的別に**2種類**ある。

- 署名タイムスタンプとアーカイブタイムスタンプ（対象が異なる）
- どちらもタイムスタンプ自体は RFC 3161 に準拠

※ 今回は**XAdES-T**の生成までを実装。

署名タイムスタンプのXML表現

SignatureTimeStamp要素（XAdES v1.3）
署名タイムスタンプ用の主要素

EncapsulatedTimeStamp要素（XAdES v1.3）
タイムスタンプトークンをBase64で指定する要素
ArchiveTimeStamp要素等でも利用される形式

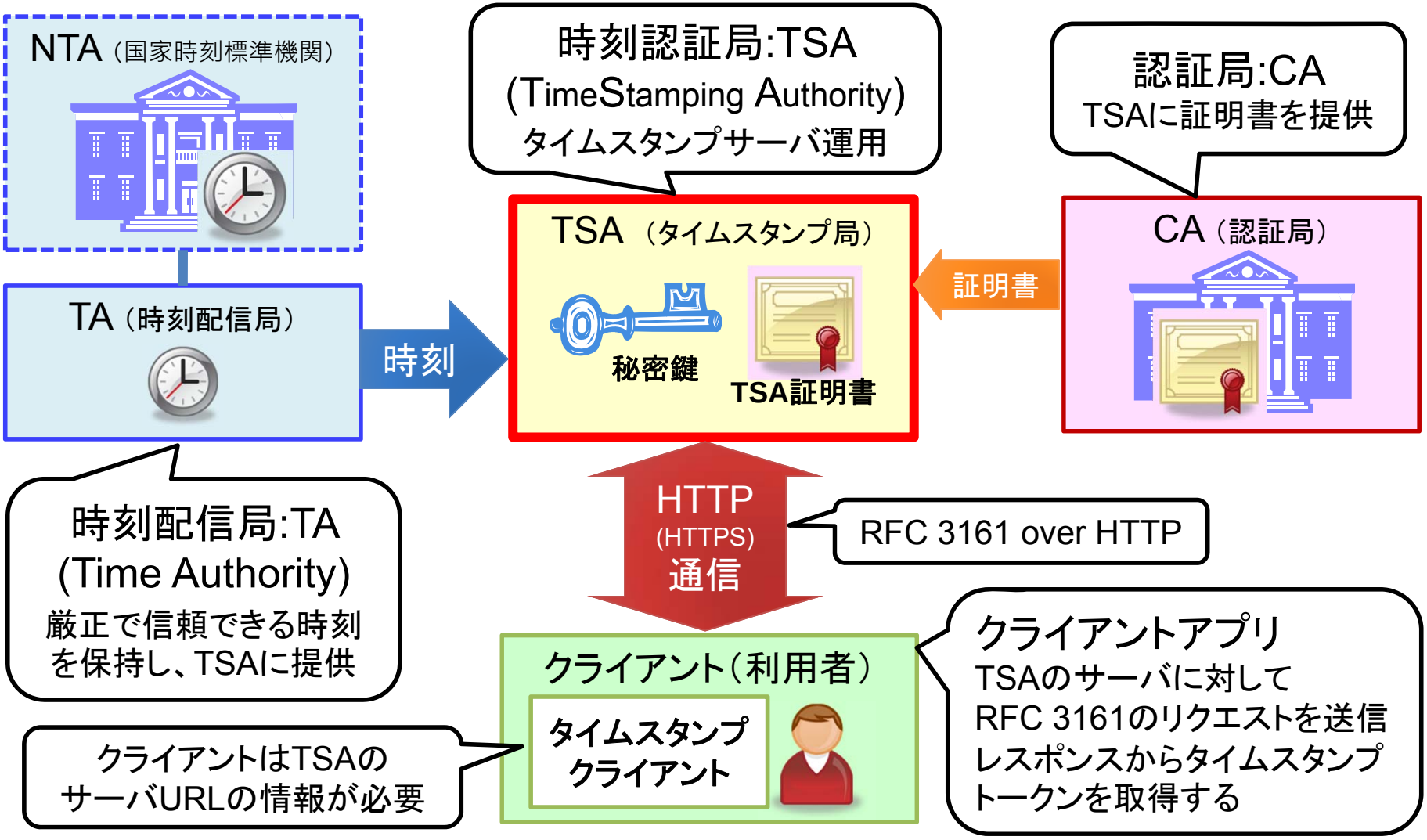
```
<xades:SignatureTimeStamp>
  <xades:EncapsulatedTimeStamp>
    MIILMAYJKoZIhvcNAQcCoIILITCCCxOCAQMxCzAJBgUrDgMCGgUAMIHZBgsqhkiG9w0BCRABBKCB
    yQSBxjCBwwIBAQYEVR0gADAvMAsGCWCGSAFlAwQCAQQgQuDEa6O4gDa6Mn6aFVrrykYTZDxzc59F
    eNhORleCiaACAhFYGA8yMDE4MDUxMTAwNTUwNlowCgIBAYACAfSBAWQBAf8CCIoF/NG9J+ZRoFmk
        (略)
    rVNQtOBx3YkMKQa3pOV3zuSS4m/KjYfMA/Tm1J5//ZGYkXVMGQ8zrOFDsU8MIGmOZQFEpcD7/xkj
    4HS6KEEKEQZYS89FnSD/nnNmt/sDGwk4iU/4wTdvDaKdh6rXuKnxEHYm1Mna0/rM7In04UNvCfUX
    /O33OHgJWoOn/X29JSikvFZk
  </xades:EncapsulatedTimeStamp>
</xades:SignatureTimeStamp>
```

Base64化されたタイムスタンプトークン
RFC 3161準拠のTimeStampToken

※ RFC 3161準拠タイムスタンプトークンが必要!

参考:タイムスタンプサービスの構成

※ タイムスタンプはサーバ署名（署名対象に時刻を含む）である！



TSクライアントアプリに必要な機能

1. タイムスタンプトークン取得

① ハッシュ値からタイムスタンプリクエストの生成
② タイムスタンプリクエストをサーバへ送信
③ タイムスタンプレスポンスの確認とトークン取得

2. タイムスタンプトークン解析

① ASN.1/DERの解析機能
② タイムスタンプ時刻の取り出し
③ タイムスタンプトークンの検証（ハッシュ値取り出し）

どちらもJava標準機能/クラスには無い!

FreeXAdESは、Java標準機能だけで実装すると宣言しているので…

フルスクラッチで実装するしかない！

（良い子はマネをしないよーにw）

（素直にBouncyCastle使えよ…）

タイムスタンプトークンの取得

実は2015年に実施した、
「電子署名（PKI）ハンズオン」
にて、タイムスタンプの解説と取得の為の
Java実装は公開済み。

資料の43ページから62ページを参照！
タイムスタンプの仕組みも分かります。
http://eswg.jnsa.org/sandbox/handson/ESig-PKI-handson-doc-v100.pdf

FreeTimeStampクラスの
getFromServerメソッドとして**再実装**した。

タイムスタンプトークンの取得API

```
/** タイムスタンプをサーバ（TSA）から取得する */
public int getFromServer (
        byte[] hash,        // タイムスタンプ対象のハッシュ値をバイナリで指定
        String url,         // タイムスタンプサーバのURLを指定
        String userid,      // Basic認証が必要な場合にユーザIDを指定
        String passwd       // Basic認証が必要な場合にパスワードを指定
);
/** 取得済みタイムスタンプから「タイムスタンプトークン」を取得する */
public byte[] getToken ();
```

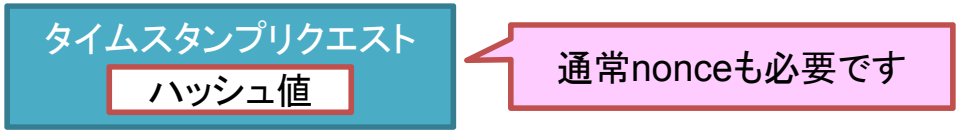
利用例：

```
// FreeTimeStampインスタンス生成
FreeTimeStamp timestamp = new FreeTimeStamp();
// ハッシュ対象の用意（FreePKI.getHashによりハッシュ計算可能）
byte[] hash = FreePKI.getHash(target.getBytes(), "SHA-256");
// タイムスタンプのサーバからの取得（ネットワーク接続/HTTP通信）
rc = timestamp.getFromServer(hash, tsUrl, tsUserid, tsPasswd);
// タイムスタンプトークンの取得
byte[] token = timestamp.getToken();
```

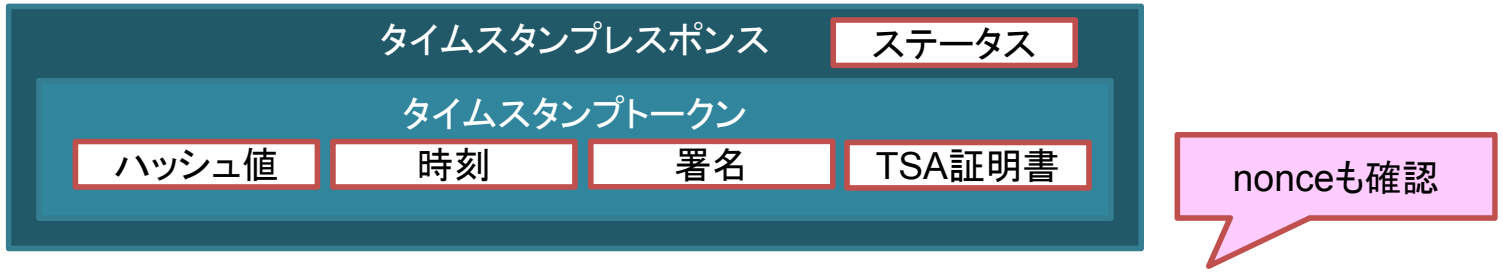
参考:タイムスタンプ取得手順

データは全て
ASN.1/DER 形式

1. 対象データのハッシュ値を計算
2. ハッシュ値を埋め込んだタイムスタンプリクエスト生成



3. タイムスタンプリクエストをサーバに送信
4. サーバからタイムスタンプレスポンスを取得



5. タイムスタンプレスポンスのステータスを確認
6. レスポンスからタイムスタンプトークンを取り出し利用

参考:タイムスタンプ リクエスト

※ FreeTimeStampでは SHA-256/512のみ対応

```
// SHA-1タイムスタンプリクエスト情報BER定義
byte[] sha1req = {
 0x30, 0x31,                                    // Request SEQUENCE (49バイト)
 0x02, 0x01, 0x01,                              // Version INTEGER (1バイト) value:1
 0x30, 0x1f,                                    // MessageImprint SEQUENCE (31バイト)
 0x30, 0x07,                                    // AlgorithmOID SEQUENCE (7バイト)
 0x06, 0x05,                                    // OID (5バイト)
 0x2b, 0x0e, 0x03, 0x02, 0x1a,                  // OIDSHA1 value: 1.3.14.3.2.26
 0x04, 0x14,                                    // Hash OCTET STRING (20バイト)
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x02, 0x08,                                    // Nonce INTEGER (8バイト)
 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
 0x01, 0x01, 0xff                               // RequestCertificate BOOLEAN (1バイト) value:true
};
```

全て固定長なのでバイナリで準備をしておけば良い。Javaソース参照。

SHA-1なら20バイトのハッシュ値をセットすれば良い

Nonceは8バイト(任意)固定

```
TimeStampReq ::= SEQUENCE  {
 version                  INTEGER  { v1(1) },
 messageImprint           MessageImprint,
 reqPolicy                TSAPolicyId OPTIONAL,
 nonce                    INTEGER OPTIONAL,
 certReq                  BOOLEAN DEFAULT FALSE,
 extensions               [0] IMPLICIT Extensions OPTIONAL
}
```

```
MessageImprint ::= SEQUENCE  {
 hashAlgorithm   AlgorithmIdentifier,
 hashedMessage   OCTET STRING
}
```

参考:タイムスタンプ レスポンス

```
TimeStampResp ::= SEQUENCE  {
 status          PKIStatusInfo,
 timeStampToken TimeStampToken OPTIONAL
}
PKIStatusInfo ::= SEQUENCE {
 status        PKIStatus,
 statusString PKIFreeText OPTIONAL,
 failInfo     PKIFailureInfo OPTIONAL
}
```

成功(granted)ならタイムスタンプトークンが含まれる

statusをチェックする

```
// レスポンスのステータス
PKIStatus ::= INTEGER {
 granted                (0),    // TSTを含む
 grantedWithMods        (1),    // TSTを含み、プライベート拡張を含む
 rejection              (2),    // TSTを含まず、拒否された
 waiting                (3),    // TSTを含まず、レシートのみ含む
 revocationWarning      (4),    // TSTを含まず、TSA証明書の失効が近い
 revocationNotification (5)     // TSTを含まず、TSA証明書が失効している
}
```

OK

- 簡単な構造だけどここでもASN.1/DERの解析が必要になる。
- タイムスタンプレスポンスの解析にも後述のFreePKIを利用。

タイムスタンプトークンの解析

ASN.1/DER解析の為に新たに、
FreePKIクラスをフルスクラッチで実装し、
FreeTimeStampクラスの
setTokenメソッドにトークン解析を実装。

```
byte[] getToken();                  // タイムスタンプトークンの取得
String getTimeStampDate();          // タイムスタンプ時刻の取得
byte[] getSerial();                 // タイムスタンプシリアル番号の取得
byte[] getNonce();                  // タイムスタンプナンスの取得
String getMsgImprintAlg();          // タイムスタンプ対象ハッシュアルゴリズムの取得
byte[] getMsgImprint();             // タイムスタンプ対象ハッシュ値の取得
X509Certificate getSignerCert();    // タイムスタンプ(TSA)証明書の取得
X509Certificate[] getAllCerts();    // タイムスタンプに含まれる全証明書の取得
```

※ ASN.1/DERについてはPKIハンズオン資料の18ページから22ページで解説。

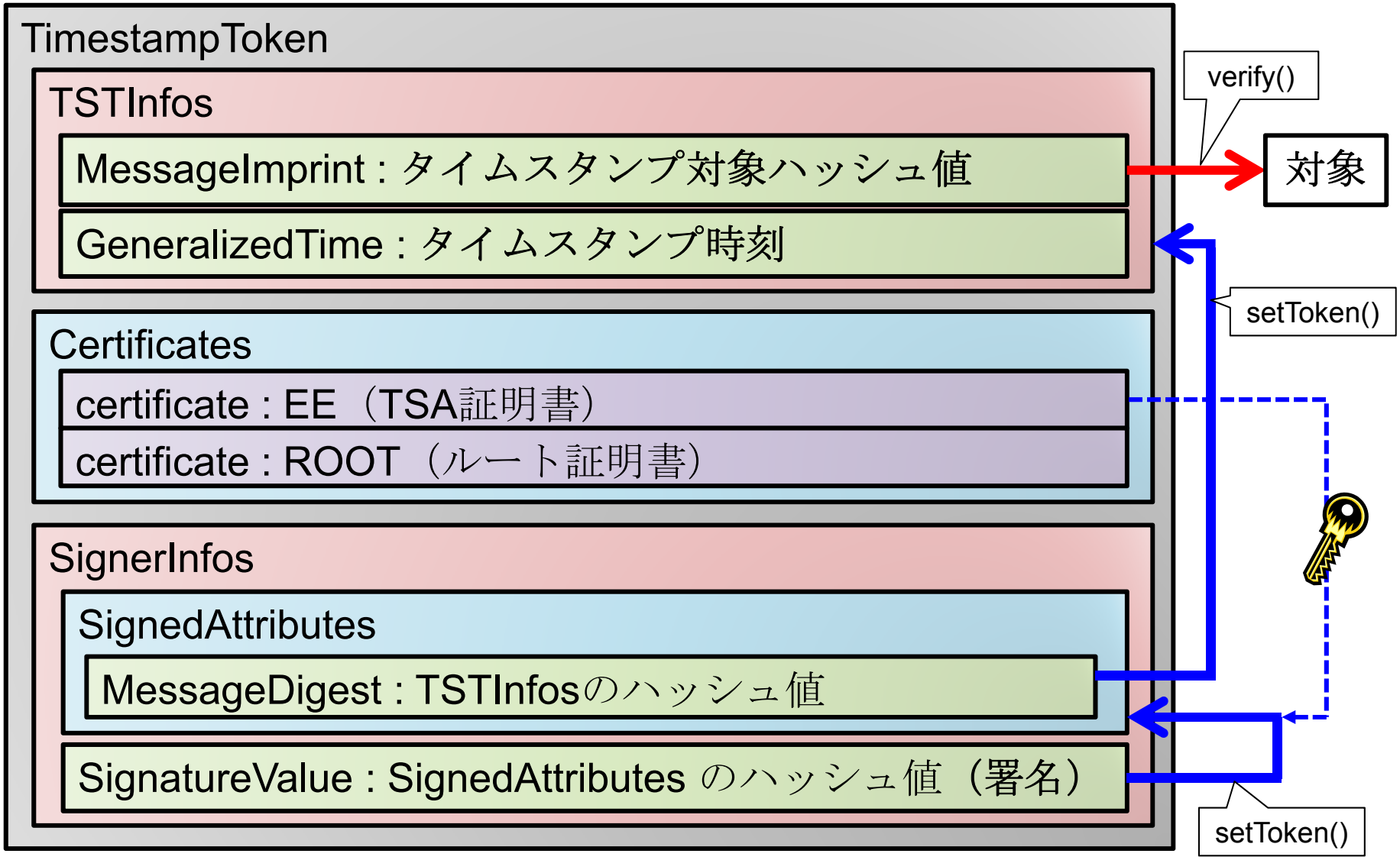
タイムスタンプトークンの解析API

```
/** 取得済みのタイムスタンプトークンのバイナリをセットして検証する
      ※ getFromServerの実行済みならsetTokenは不要 */
public int setToken (
          byte[] token        // 取得済みタイムスタンプトークンをバイナリで指定
);
/** タイムスタンプトークンを検証 (TSA証明書検証はsetTokenで実行済み) */
public int verify (
          byte[] hash         // タイムスタンプ対象のハッシュ値をバイナリで指定
);
```

利用例:

```
// FreeTimeStampインスタンス生成
FreeTimeStamp timestamp = new FreeTimeStamp();
// 取得済みタイムスタンプのセット（XAdESの署名タイムスタンプ等）
rc = timestamp.setToken(token);
// タイムスタンプの情報取得（他getTimeStampDate等APIあり 前ページ参照）
String info = timestamp.getInfo();
// タイムスタンプの検証（MessageImprintハッシュ値の確認等）
rc = timestamp.verify(hash);
```

参考:タイムスタンプトークンの検証



FreeXAdES XAdES-Tの生成API

```
/** 署名タイムスタンプを追加する */
public int addEsT (
        String url,         // タイムスタンプサーバのURLを指定
        String userid,      // Basic認証が必要な場合にユーザIDを指定
        String passwd,      // Basic認証が必要な場合にパスワードを指定
        String id,          // SignatureTimeStamp要素に付けるIdの指定
        String xpath        // 対象となるSignature要素をXPathで指定
);
```

利用例：

```
// FreeXAdESインスタンス生成
FreeXAdES xades = new FreeXAdES();
// XAdES-Bファイルの読み込み（XAdES-B署名済みのファイルを用意）
rc = xades.loadXml("xades-b.xml", IFreeXAdES.FXAT_FILE_PATH);
// 署名タイムスタンプの追加
rc = xades.addEsT(tsUrl, tsUserid, tsPasswd, "ES-T-test", xpath);
// XAdES-Tの検証（仮：XAdES-Tのタイムスタンプ検証も行う）
rc = xades.verifySign(IFreeXAdES.FXVF_NONE, xpath);
// XAdES-Tファイルの保存
rc = xades.saveXml("xades-t.xml");
```

FreeXAdES β2版のまとめ

新たに2つのクラスを実装した。

FreeTimeStamp クラス

→ タイムスタンプの取得と解析の実装

FreePKI クラス

→ ASN.1/DER解析やハッシュ値取得等の実装

FreeXAdESクラスに**addEsT**メソッドを追加。

FreeXAdESで**XAdES-T**が使えるようになった！

タイムスタンプ利用のすゝめ

タイムスタンプは**REST API的に使える。**
異なる点はJSON/XMLでは無くASN.1/DERを使うところ。
→ FreeTimeStampを使えばASN.1/DERは使える。

システムログの保護であったり研究資料の保護にも使えるタイムスタンプ！

FreeTimeStampクラスは
FreePKIクラスにのみに依存！
単独でも気軽に使えます！

次回予告

XAdES-LT（XAdES-X Long）への対応。

…いよいよ**検証**に着手します…

検証はPKIの基本であり最も重要な機能です!

果たして秋祭りで発表できるのか?
それとも来年の春祭りなのかっ?
「Re:Re:」にならないように…
ご期待ください！（誰も待って無いってw

おまけ：FreeXAdESを試す

公開リポジトリ

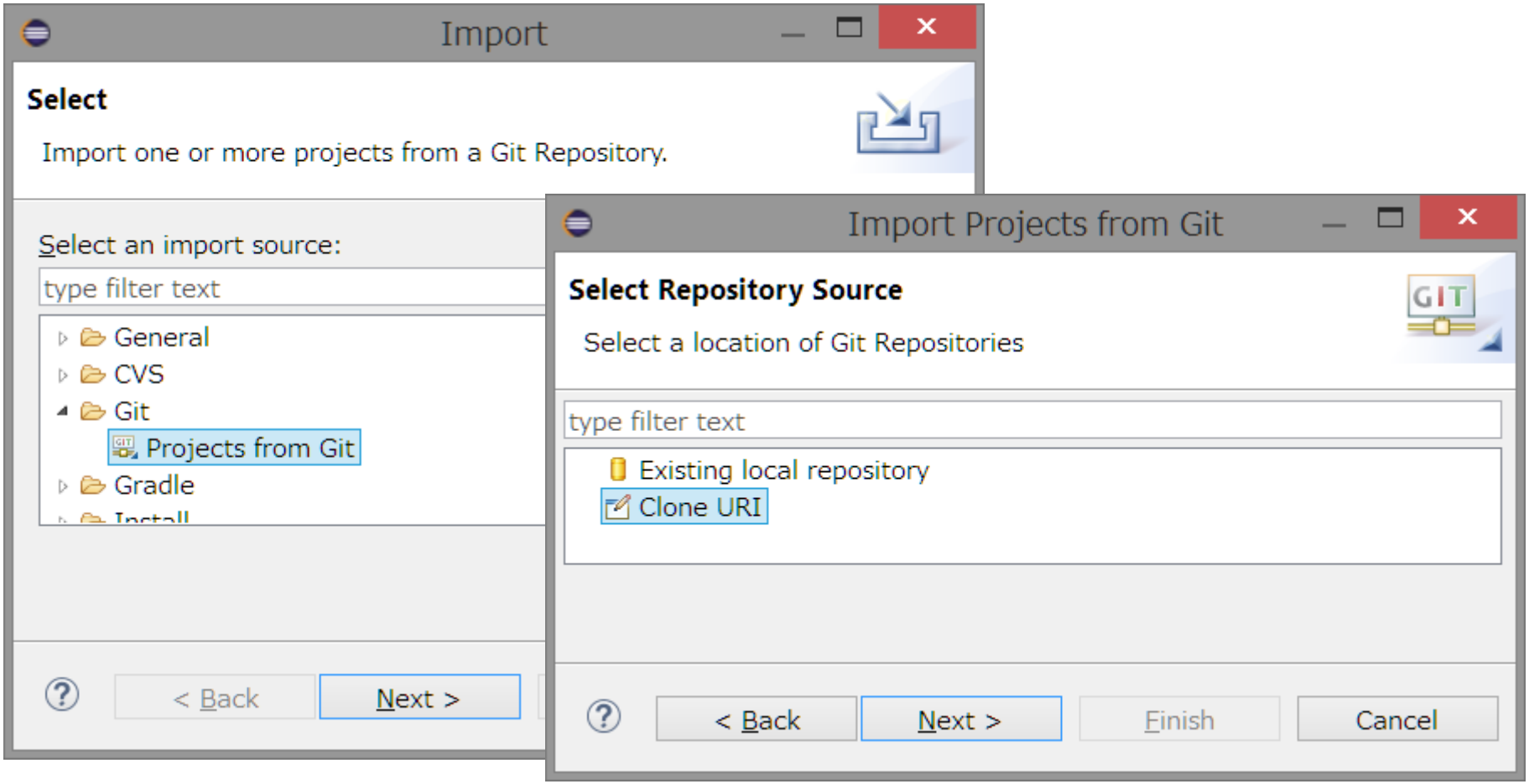
https://github.com/miyachi/FreeXAdES

開発環境

Eclipse IDE for Java Developers
Version：Mars or after (ex:Oxygen)
Java：**Java8** or after
Other：JUnit4（eclipse標準）

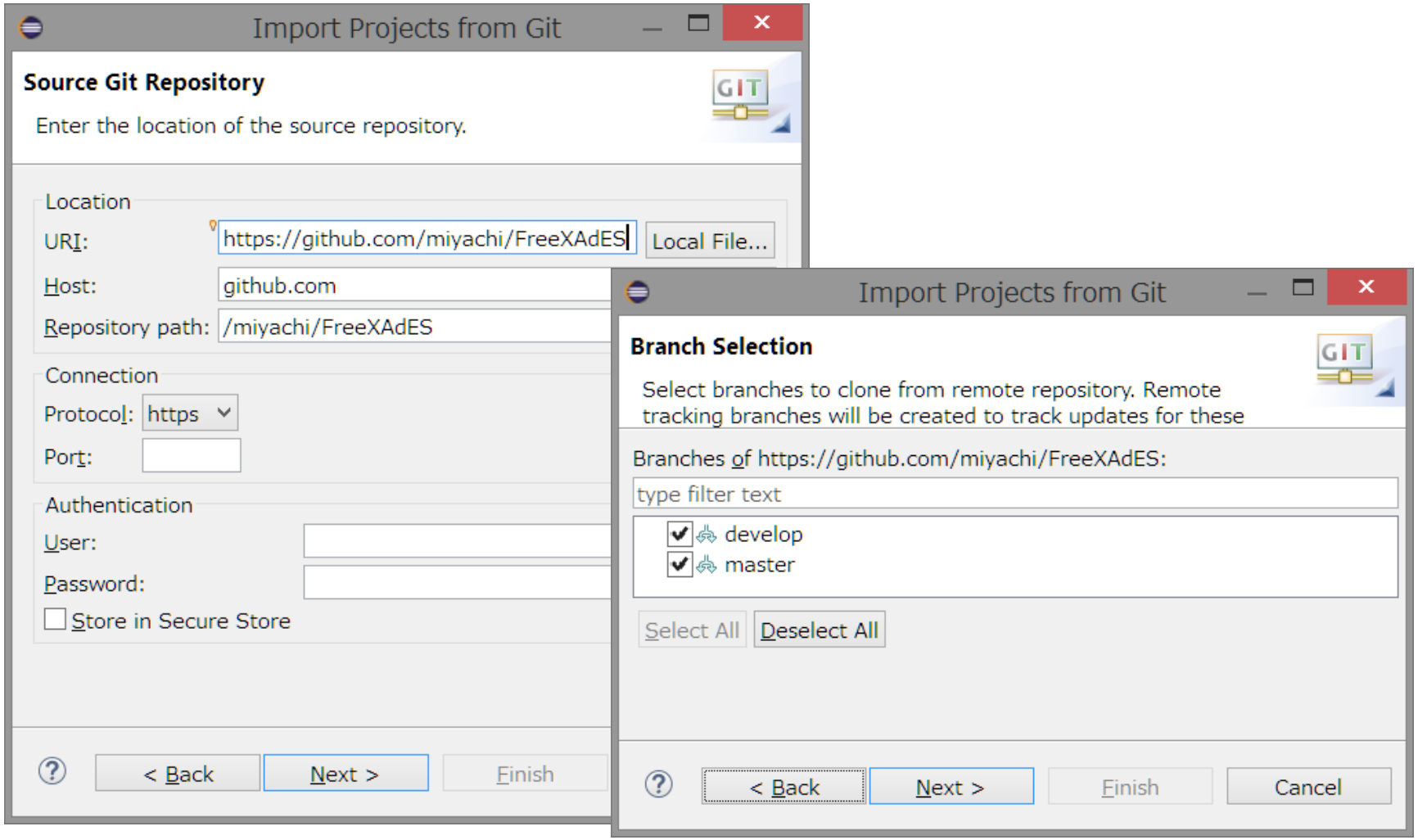
おまけ：FreeXAdESを試す手順1

Menu: [File]-[Import…]
Select: [Git]-[Projects from Git] click [Next>]
Select: [Clone URI] click [Next>]



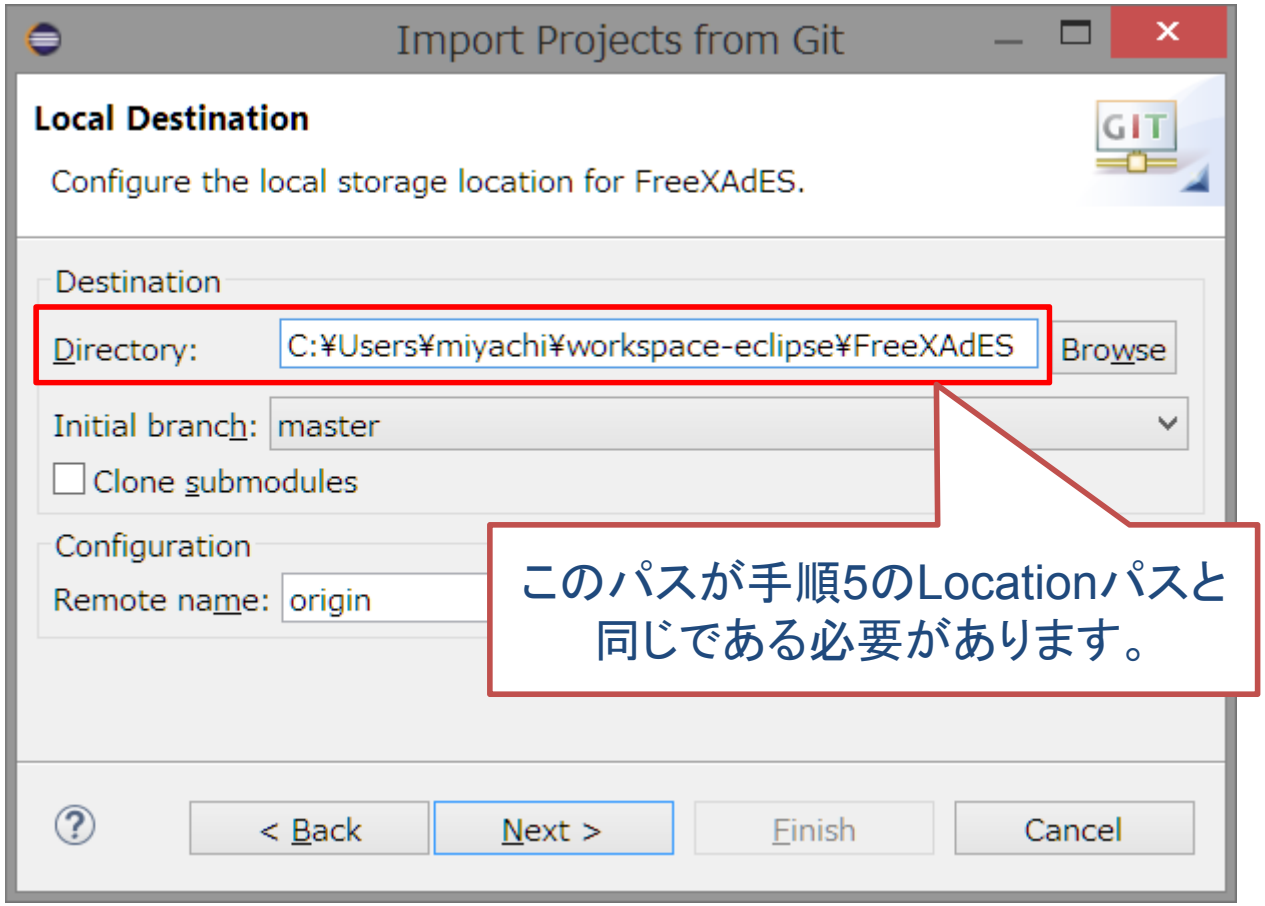
おまけ：FreeXAdESを試す手順2

URI: “https://github.com/miyachi/FreeXAdES” click [Next>]
Branch Selection: click [Next>]



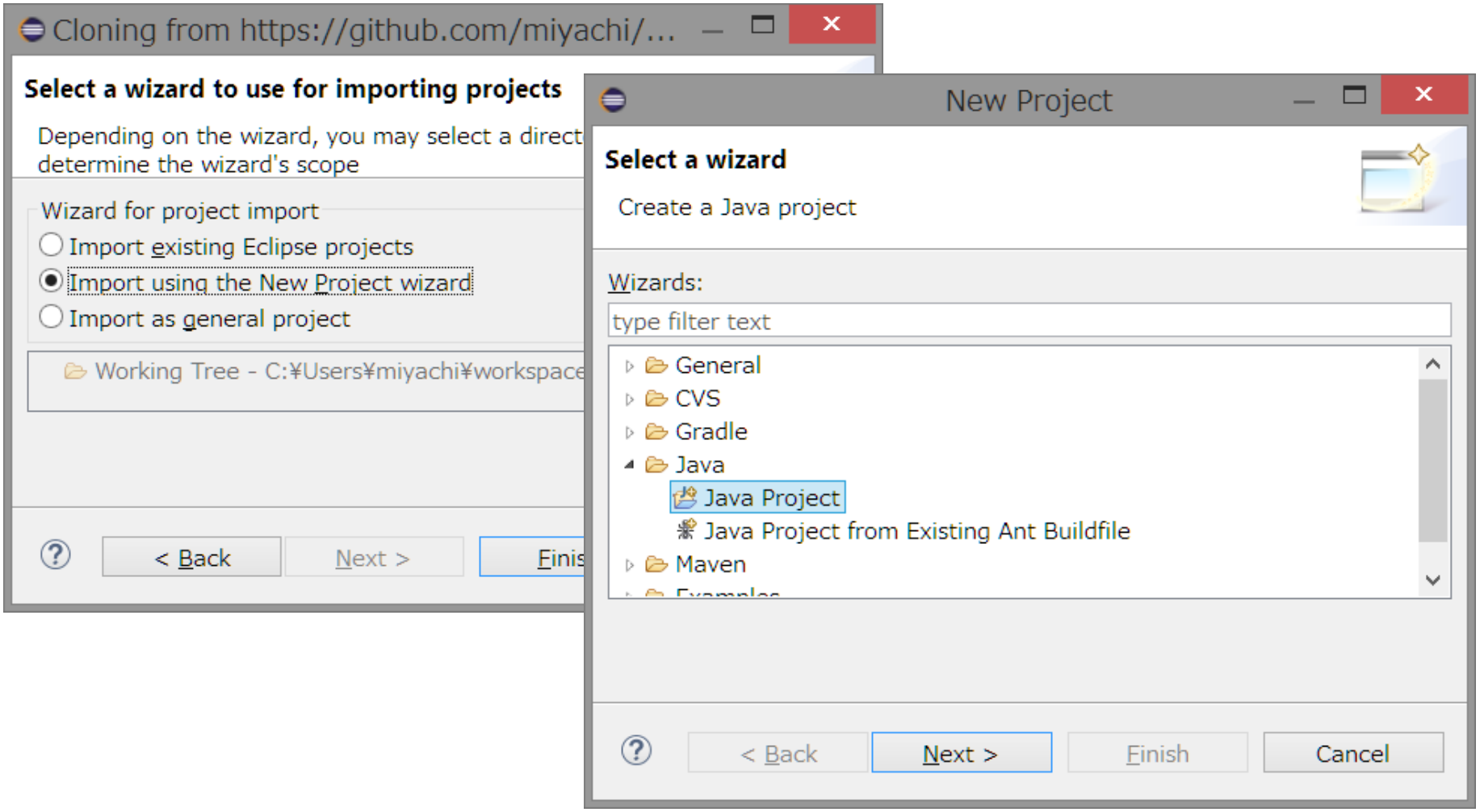
おまけ：FreeXAdESを試す手順3

Directory: “C:¥Users¥XXX¥workspace¥FreeXAdES” click [Next>]
注：Directoryの設定パスはワークスペース下のFreeXAdESにすること。



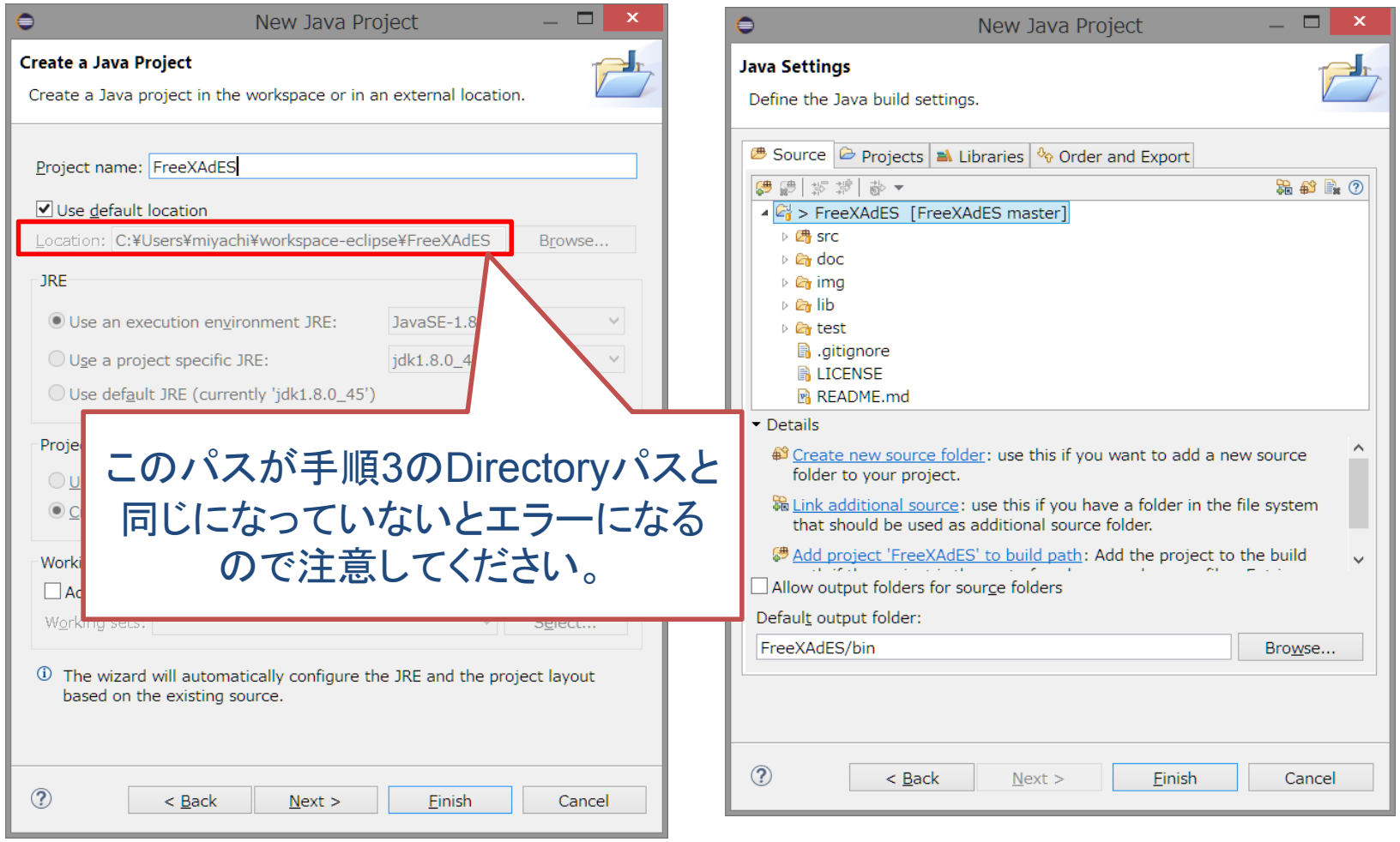
おまけ：FreeXAdESを試す手順4

Select: [Import using the New Project wizard] click [Next>]
Wizards: [Java]-[Java Project] click [Next>]



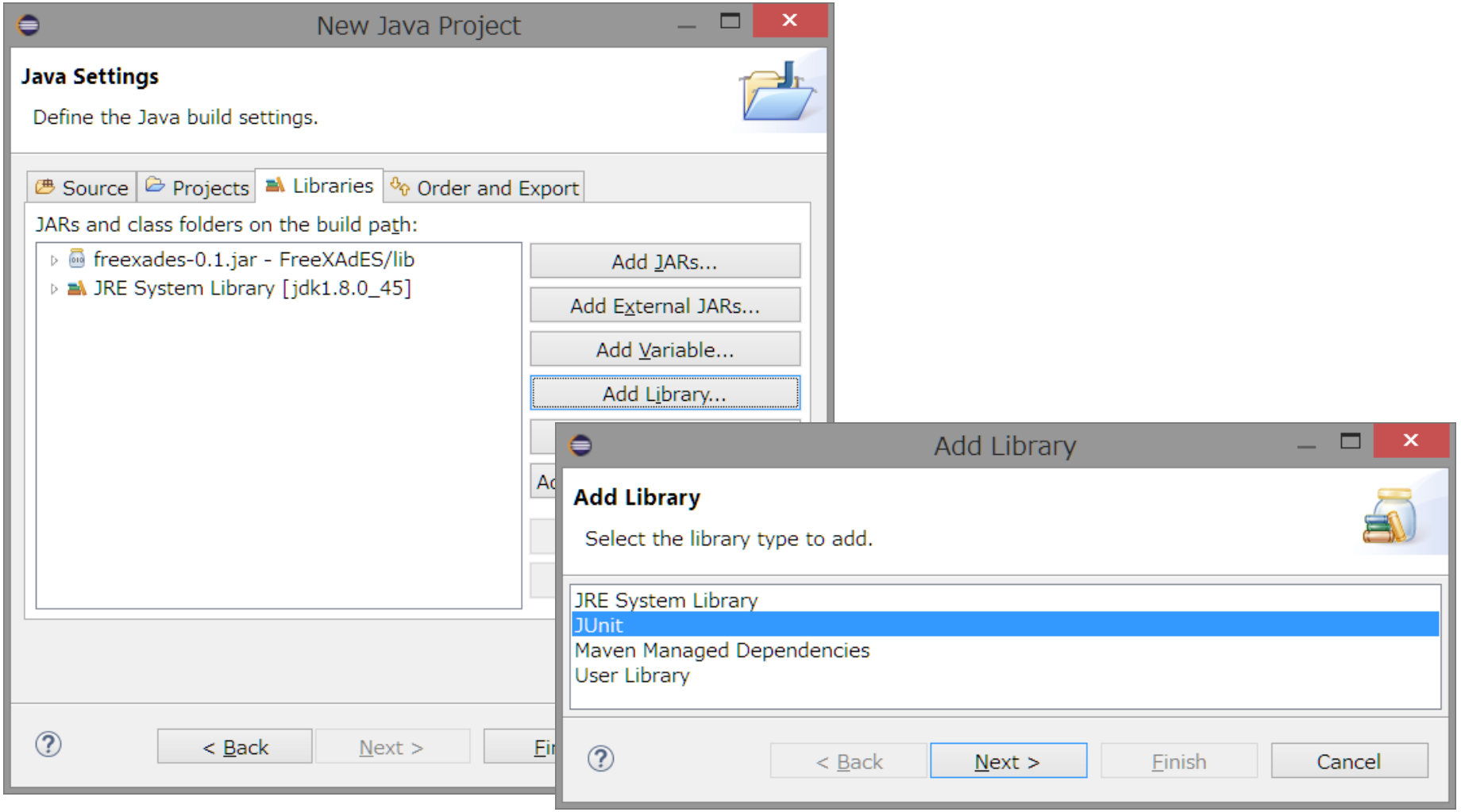
おまけ：FreeXAdESを試す手順5

Poject name: “FreeXAdES” click [Next>]
注：Locationが “C:¥Users¥XXX¥workspace¥FreeXAdES” になること。



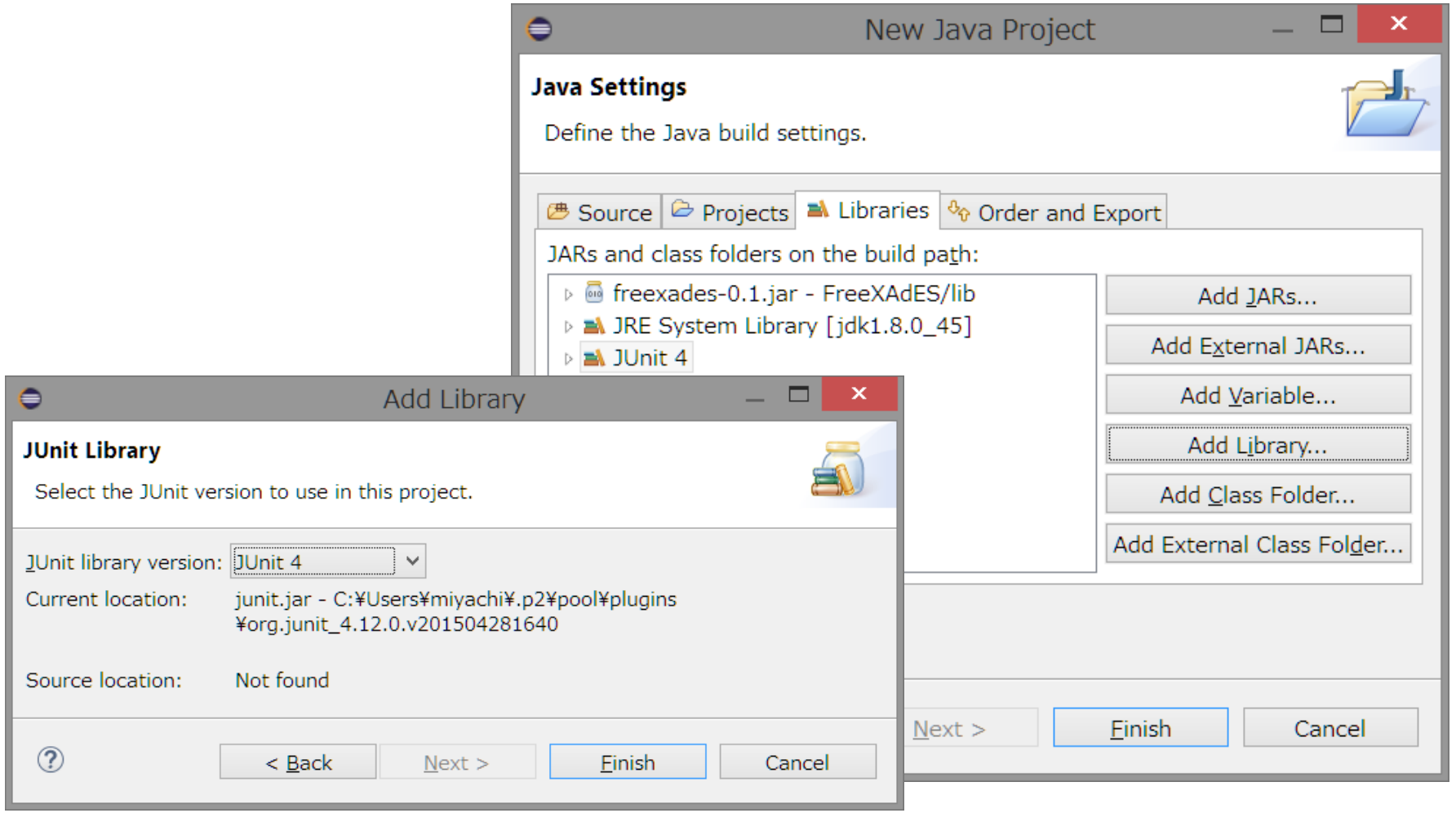
おまけ：FreeXAdESを試す手順6

Libraries: click [Add Library]
Add Library: select [JUnit] click [Next>]



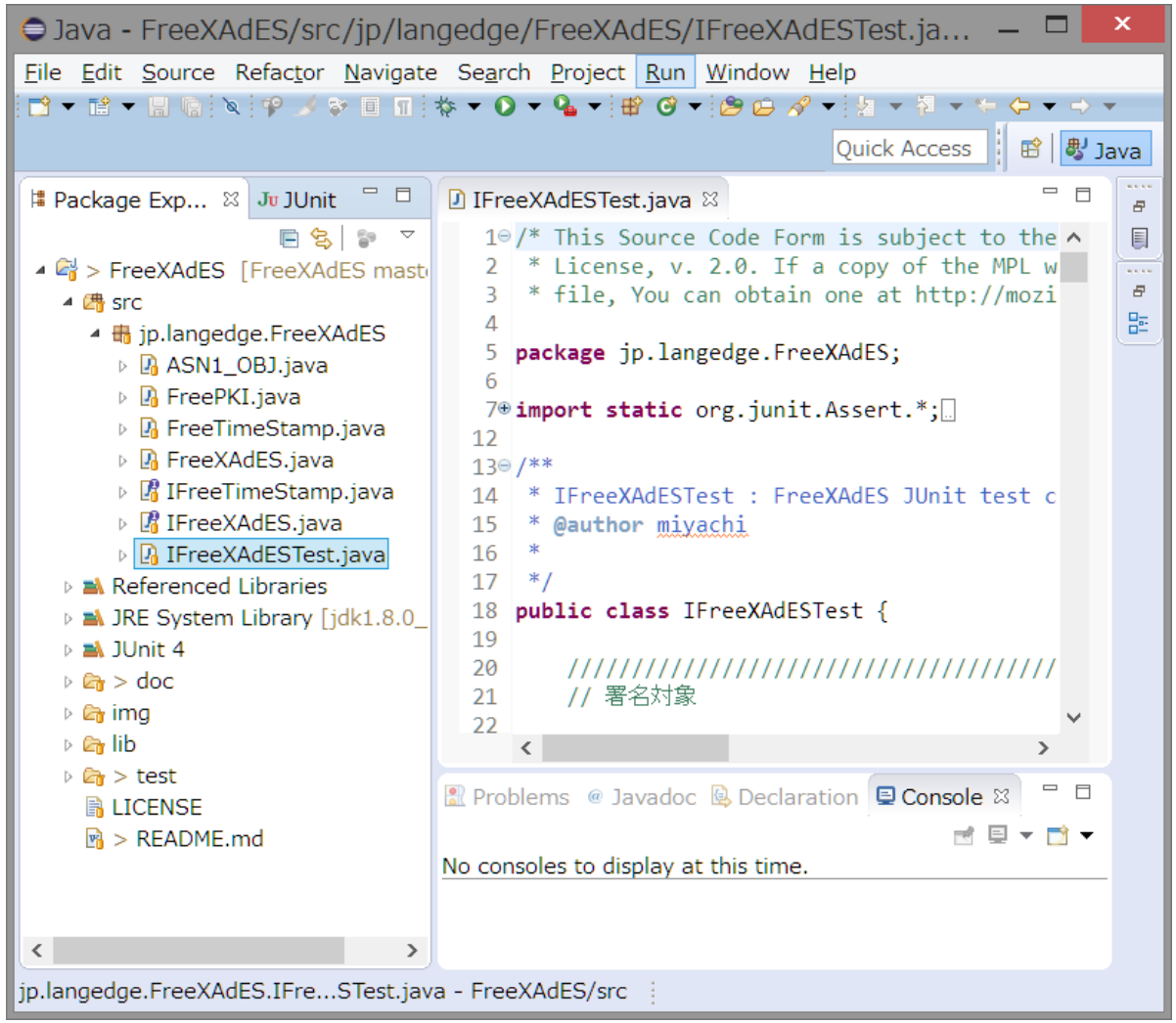
おまけ：FreeXAdESを試す手順7

JUnit Library version: select [JUnit 4] click [Finish]
Java Setting: click [Finish]



おまけ：FreeXAdESのJUnit実行

Open: FreeXAdeS/src/jp/langedge/FreeXAdES/IFreeXAdESTest.java
Menu: [Run]-[Run]



おまけ：FreeXAdESのSample実行

Windows:

```
> cd test
> FxSample.bat
Sample Compile.
Sample Execute.
 - create.
 - add Detached.
 - exec Sign.
 - add ES-T.
 - save XAdES-T.
 - exec Verify = [OK]
 - done.
Sample Finished.
>
```

Eclipse/JUnit不要

Linux:

```
$ cd test
$ chmod +x *.sh
$ ./FxSample.sh
Sample Compile.
Sample Execute.
 - create.
 - add Detached.
 - exec Sign.
 - add ES-T.
 - save XAdES-T.
 - exec Verify = [OK]
 - done.
Sample Finished.
$
```