目次

改訂情報

変更年月日	変更内容
2016-08-01	初版
2018-12-01	第2版 下記を追加・変更しました。 ▪ 表記のゆれを訂正しました。
2019-04-01	第3版 下記を追加・変更しました。 ▪ トラブルシューティングを本書から独立させました。
2020-04-01	第4版 下記を追加・変更しました。 ▪ Windows 7 / Windows Server 2008 の記述を削除しました。 ▪ 「 *はじめに* 」の トラブルシューティングに関する記載を削除しました。
2020-08-01	第5版 下記を追加・変更しました。 ▪ 「 *サポート* 」の内容を見直しました。 ▪ 「 *ステータスコード* 」の記載を追加・変更しました。 ▪ 見出しを「エラーコード」から「ステータスコード」へ変更しました。 ▪ 目次を「エラーコード一覧」から「ステータスコード一覧」へ変更しました。 ▪ ステータスコード一覧にステータスコード -125 の記載を追加しました。
2021-08-01	第6版 下記を追加・変更しました。 ▪ 「 *ステータスコード一覧* 」へ注意を追加しました。
2023-10-01	第7版 下記を追加・変更しました。 ▪ 「 *API リストについて* 」のAPIリストの所在を変更 ▪ 「 *前提条件* 」のチュートリアルについての記述、および、前提条件についての記述を変更 ▪ 「 環境 」を削除 ▪ 「 *準備* 」を追加 ▪ 「 サンプルプログラムの場所 」を削除 ▪ 「 *JSP プログラムの作成* 」のサンプルプログラムを変更、作成手順の記述を変更、指定する情報についてコラムを変更、および、保存時の文字コードについて注意を追加 ▪ 「 *サンプルプログラム* 」のサンプルプログラムを変更
2024-10-01	第8版 下記を追加・変更しました。 ▪ 「 *JSP プログラムの作成* 」のサンプルプログラムを変更、および、タイムスタンプ局のポリシーに関するコラムを追加 ▪ 「 *サンプルプログラム* 」のサンプルプログラムを変更 ▪ 「 *ステータスコード一覧* 」を見直し、記述を変更

はじめに

目次

本書の目的

本書では、 IM-PDFTimeStamper for Accel Platform を利用する場合の基本的な方法や注意点等について説明します。

対象読者

本書は、開発をスムーズに開始するための手引書です。

したがって、実際に IM-PDFTimeStamper for Accel Platform を利用したアプリケーションを開発するプログラマの方が対象です。

- 以下のいずれかを理解していることが必須です。
  - JavaEE開発モデル （ Java ）
  - スクリプト開発モデル （ サーバサイドJavaScript ）

また、本書は、以下に列挙する技術に関する知識を有することを前提として構成されています。

これらの技術に関して不明な点がある場合、本ドキュメントの内容を正しく理解することが困難になることがありますので、予めご了承ください。

なお、前提知識となる技術に関しては、一般の専門書籍等を参照してください。

- Javaプログラミング言語
- Java Servlet および JSP
- オペレーティングシステム
- ネットワーク

本書の構成

- *API リスト*

  利用できるAPIについて説明します。

- *プログラミング*

  プログラム開発の際の注意点や、プログラムの方法などを説明します。

- *チュートリアル*

  本製品のAPI を利用して実際にプログラムを作成する過程を学びます。

- *ステータスコード*

エラー発生時に返されるエラーコードを説明します。

- *サポート*

  製品サポートおよび技術情報の公開について説明します。

APIリスト

目次

APIリストについて

IM-PDFTimeStamper for Accel Platform には、 JavaEE開発モデル 用のAPI が用意されています。

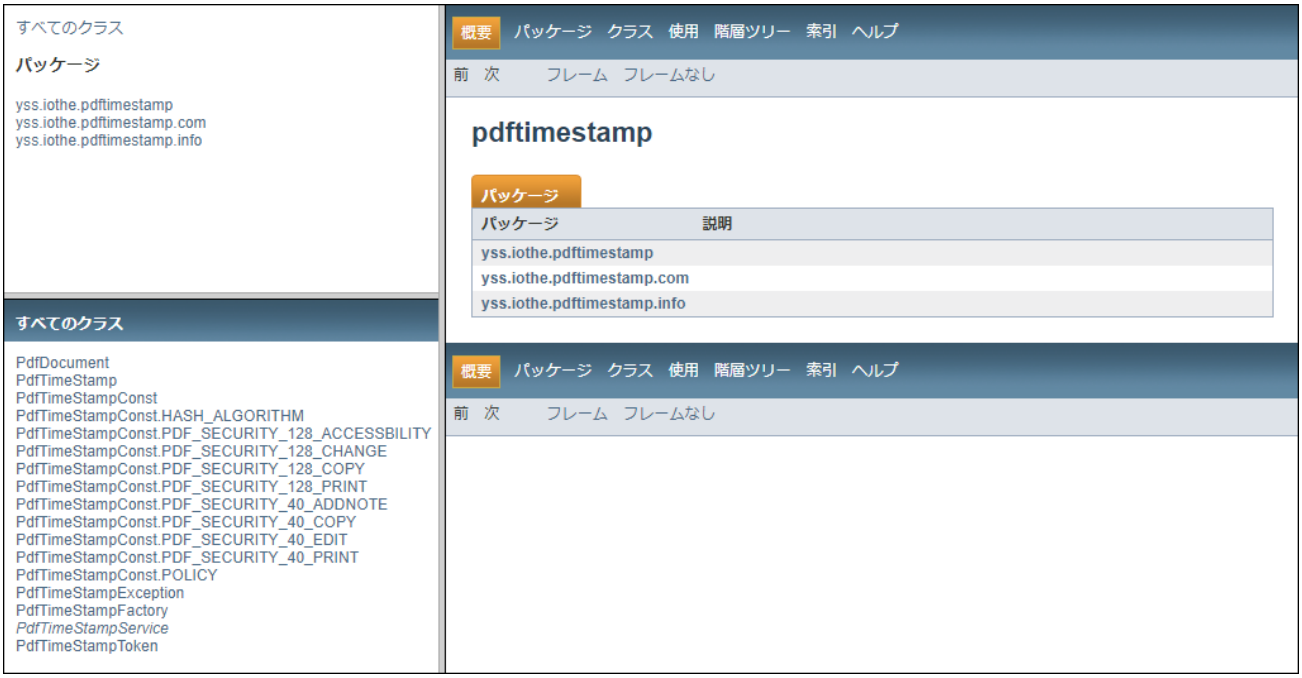
スクリプト開発モデル で開発をする場合は、スクリプト開発モデル のソースコード内でJavaのクラスを呼び出してください。

IM-PDFTimeStamper for Accel Platform のAPIリストは、次の通りです。

- IM-PDFTimeStamper for Accel Platform API ドキュメント

JavaEE開発モデル

IM-PDFTimeStamper for Accel Platform は、JavaEE開発モデル で利用可能なJava-API（クラス）を用意しています。



スクリプト開発モデル

IM-PDFTimeStamper for Accel Platform は、JavaEE開発モデル で利用可能なJava-API（クラス）を用意しています。
そのため、 スクリプト開発モデル で IM-PDFTimeStamper for Accel Platform を利用する場合は、スクリプト開発モデル のソースコード内でJavaのクラスを呼んでください。
スクリプト開発モデル 内でのJavaのクラスの呼び出し方法については、intra-mart 付属のマニュアルを参照ください。

プログラミング

目次

動作概念

通常の JavaEE開発モデル スクリプト開発モデル プログラムは、ApplicationRuntime で実行されます。
IM-PDFTimeStamper for Accel Platform で提供されるAPI も、そのほとんどはApplicationRuntime で動作します。
以下の方法でタイムスタンプ処理が実行できます。詳しくは、APIリストを参照してください。

No.	メソッド	説明
1	void generate ()	PDFに対して文書タイムスタンプを付与します。
2	void generateLtv()	PDFに対して延長タイムスタンプを付与します。
3	void getPdfDocument()	PDFの情報を取得します。
4	int validate()	PDFのタイムスタンプを検証します。

エラー処理について

各タイムスタンプ処理でエラーが発生した場合、例外がスローされます。
例外からは下記の情報が取得可能です。詳しくは、APIリストを参照してください。

- エラーコード
- エラーメッセージ

APIの種類と性質

IM-PDFTimeStamper for Accel Platform は、JavaEE開発モデル で利用可能なJava-API（クラス）を用意しています。
そのため、 スクリプト開発モデル で IM-PDFTimeStamper for Accel Platform を利用する場合は、スクリプト開発モデル のソースコード内でJavaのクラスを呼んでください。
スクリプト開発モデル 内でのJavaのクラスの呼び出し方法については、intra-mart 付属のマニュアルを参照してください。

プログラム開発における注意点

IM-PDFTimeStamper for Accel Platform が提供するAPIでファイルのパスを指定する際には、AppRuntimeからアクセス可能なパスを指定してください。
処理するPDFファイルのサイズによっては、ネットワーク、APIのレスポンス、PDFファイルの処理が完全に終了するタイミングが大きく異なる場合があります。

特にサイズの大きいPDFファイルを処理する場合は、十分な時間が経過した後にPDF ファイルにアクセスするようにしてください。

体験版ライセンスにおける注意点

試用版ライセンスでご利用のお客様は、60日間の試用期間が終了するとAPIが自動的に利用できない状態となります。
その場合は、正規の製品ライセンスを購入いただき、アンインストール後に再インストールしてください。
アンインストール・再インストールの方法は、インストールマニュアルをご確認ください。

チュートリアル

目次

前提条件

本項では、 IM-PDFTimeStamper for Accel Platform での開発の導入として、APIを利用したPDFファイルへのタイムスタンプ付与処理を作成することによって、 IM-PDFTimeStamper for Accel Platform での開発の流れを体験します。

本項のチュートリアルを開始するにあたっての前提条件は、次の通りです。

- intra-mart Accel Platform 、および、 IM-PDFTimeStamper for Accel Platform が正しくセットアップされていること
- セイコーソリューションズ株式会社のセイコータイムスタンプサービスの契約が完了していること

ここでは、 JavaEE開発モデル の開発の流れを説明します。

用語解説

- Resin をインストールしたディレクトリを %RESIN_HOME% と略します。
- Apache HTTP Server をインストールしたディレクトリを %APACHE_HOME% と略します。
- Storage として使用するディレクトリを %PUBLIC_STORAGE_PATH% と略します。
- Webサーバ利用時の静的コンテンツを配置するディレクトリを %WEB_PATH% と略します。

準備

タイムスタンプを付与するPDFファイルを「 in.pdf 」のファイル名で作成し、 intra-mart Accel Platform サーバの < C:/temp >ディレクトリに配置してください。

JSPプログラムの作成

テキストエディタを使用してJSPファイルを作成します。

Resin の場合、< %RESIN_HOME%/webapps/warファイルと同名のディレクトリ/ >の配下に、「 PdfTimeStampSample.jsp 」の名前でファイルを作成し、次のソースを実装します。

```
<%@ page language="java" contentType="text/html; charset=UTF-8" %>
<%@ page import="yss.iothe.pdftimestamp.PdfTimeStampException" %>
<%@ page import="yss.iothe.pdftimestamp.PdfTimeStampFactory" %>
<%@ page import="yss.iothe.pdftimestamp.PdfTimeStampService" %>
<%@ page import="yss.iothe.pdftimestamp.com.PdfTimeStampConst.HASH_ALGORITHM" %>
<%@ page
import="yss.iothe.pdftimestamp.com.PdfTimeStampConst.PDF_SECURITY_128_ACCESSBILITY" %>
<%@ page
import="yss.iothe.pdftimestamp.com.PdfTimeStampConst.PDF_SECURITY_128_CHANGE" %>
<%@ page import="yss.iothe.pdftimestamp.com.PdfTimeStampConst.PDF_SECURITY_128_COPY"
%>
<%@ page import="yss.iothe.pdftimestamp.com.PdfTimeStampConst.PDF_SECURITY_128_PRINT"
%>
<%@ page
import="yss.iothe.pdftimestamp.com.PdfTimeStampConst.PDF_SECURITY_40_ADDNOTE" %>
<%@ page import="yss.iothe.pdftimestamp.com.PdfTimeStampConst.PDF_SECURITY_40_COPY"
%>
<%@ page import="yss.iothe.pdftimestamp.com.PdfTimeStampConst.PDF_SECURITY_40_EDIT"
%>
<%@ page import="yss.iothe.pdftimestamp.com.PdfTimeStampConst.PDF_SECURITY_40_PRINT"
%>
<%@ page import="yss.iothe.pdftimestamp.com.PdfTimeStampConst.POLICY" %>
<%@ page import="yss.iothe.pdftimestamp.info.PdfDocument" %>
<%@ page import="yss.iothe.pdftimestamp.info.PdfTimeStamp" %>
<%@ page import="yss.iothe.pdftimestamp.info.PdfTimeStampToken" %>
<%@ page import="java.util.Date" %>
<%@ taglib prefix="imui" uri="http://www.intra-mart.co.jp/taglib/imui" %>
<%!

 /**
  * PDFタイムスタンプインスタンス生成
  *   URLが未指定であればスタンドアロン環境、
  *   URLが指定されていれば分散環境として処理を行う。
  *   URLは下記の形式で指定する。
  *     「http://{IPアドレスおよびポート番号}/pdftimestamp/webapi/timestamp」
  */
 private static PdfTimeStampService service =
PdfTimeStampFactory.createPdfTimeStampService();
 /*
 private static PdfTimeStampService service = PdfTimeStampFactory.createPdfTimeStampService(
    "http://xxxxxxxx:xxxx/pdftimestamp/webapi/timestamp", 30, 600);
 */

 /****************************************
  * 処理ファイル設定
  ****************************************/
 /* 処理対象PDFファイルパス */
 private static final String inputpdfpath = "C:/temp/in.pdf";

 /* 処理対象PDF権限パスワード */
 private static final String inputpdfpasswd = "security";

 /* 処理結果PDF出力先ファイルパス1 */
 private static final String outpdfpath1 = "C:/temp/out.1.pdf";

 /* 処理結果PDF出力先ファイルパス2 */
 private static final String outpdfpath2 = "C:/temp/out.2.pdf";

 /****************************************
  * タイムスタンプトークン取得設定
  ****************************************/
```

```
     ****************************************/
    /* ハッシュアルゴリズム */
    private static final HASH_ALGORITHM alg = HASH_ALGORITHM.SHA512;

    /* ポリシー */
    private static final POLICY policy = POLICY.TYPEA3;

    /*****************************************
     * タイムスタンプ局の設定
     *****************************************/
    /* タイムスタンプ局のURL */
    private static final String tsaurl = "https://pades01.seiko-cybertime.jp/basic/Timestamp";

    /* タイムスタンプ局接続時の接続ID */
    private static final String tsauser = "xxxxxxxx@xxxx.xx.xx";

    /* タイムスタンプ局接続時のパスワード */
    private static final String tsapasswd = "xxxxxxxx";

    /*****************************************
     * セキュリティ設定
     *****************************************/
    /* 処理結果PDFファイルに付与する参照パスワード */
    private static final String openpasswd = "open";

    /* 処理結果PDFファイルに付与するセキュリティパスワード */
    private static final String secpasswd = "security";

    /*****************************************
     * 処理結果PDFファイルに付与する
     * RC4 40bitセキュリティ設定
     *****************************************/
    /* 印刷許可 */
    private static final PDF_SECURITY_40_PRINT security40_print =
    PDF_SECURITY_40_PRINT.DISABLE;

    /* 編集許可 */
    private static final PDF_SECURITY_40_EDIT security40_edit = PDF_SECURITY_40_EDIT.ENABLE;

    /* コピー許可 */
    private static final PDF_SECURITY_40_COPY security40_copy =
    PDF_SECURITY_40_COPY.DISABLE;

    /* 注釈追記許可 */
    private static final PDF_SECURITY_40_ADDNOTE security40_addnote =
    PDF_SECURITY_40_ADDNOTE.DISABLE;

    /*****************************************
     * 処理結果PDFファイルに付与する
     * RC4 1280bitセキュリティ設定
     *****************************************/
    /* 印刷許可 */
    private static final PDF_SECURITY_128_PRINT security128_print =
    PDF_SECURITY_128_PRINT.ENABLE;

    /* アクセス許可 */
    private static final PDF_SECURITY_128_ACCESSBILITY security128_access =
    PDF_SECURITY_128_ACCESSBILITY.ENABLE;

    /* 印刷許可 */
    private static final PDF_SECURITY_128_COPY security128_copy =
```

```
PDF_SECURITY_128_COPY.ENABLE;

/* 文書変更許可 */
 private static final PDF_SECURITY_128_CHANGE security128_change =
PDF_SECURITY_128_CHANGE.ENABLE;
%>
<imui:head>
 <title>IM-PDFTimeStamper-チュートリアル-JavaEE開発モデル-TimeStamp</title>
</imui:head>

<div class="imui-title">
 <h1>IM-PDFTimeStamper チュートリアル JavaEE開発モデル TimeStamp</h1>
</div>

<div class="imui-form-container">
 <div class="imui-chapter-title"><h2>実行結果</h2></div>
  <form>
   <table class="imui-table">
    <tbody>
     <tr>
      <th class="wd-225px">文書タイムスタンプ付与</th>
      <td><%= addTimestamp() %></td>
     </tr>
    </tbody>
   </table>
   <table class="imui-table">
    <tbody>
     <tr>
      <th class="wd-225px">延長タイムスタンプ付与</th>
      <td><%= extendTimestamp() %></td>
     </tr>
    </tbody>
   </table>
   <table class="imui-table">
    <tbody>
     <tr>
      <th class="wd-225px">PDF文書のタイムスタンプ検証(validate)</th>
      <td><%= validateTimestamp() %></td>
     </tr>
    </tbody>
   </table>
   <table class="imui-table">
    <tbody>
     <tr>
      <th class="wd-225px">PDF文書のタイムスタンプ検証(validateTs)</th>
      <td><%= validateTsTimestamp() %></td>
     </tr>
    </tbody>
   </table>
   <table class="imui-table">
    <tbody>
     <tr>
      <th class="wd-225px">PDF文書の情報取得</th>
      <td><%= getInfo() %></td>
     </tr>
    </tbody>
   </table>
  </form>
</div>

<%!
```

```
183     /**
184      * 文書タイムスタンプの付与
185      */
186     String addTimestamp() {
187
188      StringBuffer resultBuffer = new StringBuffer(200);
189
190      /* PDF情報 */
191      PdfDocument docinfo;
192
193      try {
194       /*****************************************
195        * 実行準備
196        *****************************************/
197       /* 文書タイムスタンプを付与するPDFファイルのパス、及び権限パスワードを設定 */
198       service.setInputPdf(inputpdfpath, inputpdfpasswd);
199
200       /* 処理結果PDF出力先パスを設定 */
201       service.setOutputPdf(outpdfpath1);
202
203       /* タイムスタンプトークン取得用のハッシュアルゴリズムを設定 */
204       service.setHashAlgorithm(alg);
205
206       /* タイムスタンプトークン取得用のポリシーを設定 */
207       service.setPolicy(policy);
208
209       /* タイムスタンプ局の接続先のURLを設定 */
210       service.setTsaUrl(tsaurl);
211
212       /* タイムスタンプ局接続時の接続ID、パスワードを設定 */
213       service.setTsaUser(tsauser, tsapasswd);
214
215       /* 出力PDFのセキュリティ設定(40/128のどちらかを指定) */
216       /*
217       service.setSecurity40(openpasswd, secpasswd,
218          security40_print, security40_edit,
219          security40_copy, security40_addnote);
220       */
221       service.setSecurity128(openpasswd, secpasswd,
222          security128_print, security128_access,
223          security128_copy, security128_change);
224
225       /*****************************************
226        * 実行
227        *****************************************/
228       /* PDFに対し文書タイムスタンプを付与 */
229       service.generate();
230
231       resultBuffer.append("文書タイムスタンプ付与：SUCCESS<br>");
232
233       /*****************************************
234        * 実行
235        *****************************************/
236       /* PDFおよびタイムスタンプ情報の取得 */
237       docinfo = service.getPdfDocument();
238
239       /*****************************************
240        * 取得情報出力：PDFドキュメント情報
241        *****************************************/
242       /* ファイルサイズの取得 */
243       long fileSize = docinfo.getContentSize();
```

```
244     resultBuffer.append("ファイルサイズ：" + fileSize + " Bytes<br>");
245
246     /* ページ数の取得 */
247     int pageNum = docinfo.getNumberOfPages();
248     resultBuffer.append("ページ数：" + pageNum + " Pages<br>");
249
250     /* ページサイズ(幅)の取得 */
251     double paperWidth = docinfo.getPaperWidth();
252     resultBuffer.append("ページ幅：" + paperWidth + " pts<br>");
253
254     /* ページサイズ(高さ)の取得 */
255     double paperHeight = docinfo.getPaperHeight();
256     resultBuffer.append("ページ高：" + paperHeight + " pts<br>");
257
258     /* 解像度取得 */
259     int resolution = docinfo.getResolution();
260     resultBuffer.append("解像度：" + resolution + " dpi<br>");
261
262     /* 階調取得 */
263     int gradation = docinfo.getGradation();
264     resultBuffer.append("階調：" + gradation + "<br>");
265
266     /* 最新のタイムスタンプ情報の取得 */
267     // PdfTimeStamp timeStamp = docinfo.getLatestTimeStamp();
268
269     /* タイムスタンプリストの取得 */
270     PdfTimeStamp[] timeStampList = docinfo.getTimeStamps();
271     resultBuffer.append("タイムスタンプリスト：" + timeStampList.length + "<br>");
272
273     /* VRI付きタイムスタンプリストの取得 */
274     PdfTimeStamp[] vriTimeStampList = docinfo.getTimeStampsWithVRI();
275     if (vriTimeStampList != null) {
276      resultBuffer.append("VRI付きタイムスタンプリスト：" + vriTimeStampList.length + "<br>");
277     }
278
279     /* VRI付きでないタイムスタンプリストの取得 */
280     PdfTimeStamp[] notVriTimeStampList = docinfo.getTimeStampsWithNotVRI();
281     if (notVriTimeStampList != null) {
282      resultBuffer.append("VRI付きでないタイムスタンプリスト：" + notVriTimeStampList.length + "
283 <br>");
284     }
285
286     /*****************************************
287      * 取得情報出力：タイムスタンプ情報
288      *****************************************/
289     for (int i = 0; i < timeStampList.length; i++) {
290      resultBuffer.append("【TimeStamp[" + i + "]】<br>");
291
292      /* ハッシュアルゴリズムの取得 */
293      HASH_ALGORITHM hashAlgorithm = timeStampList[i]
294          .getHashAlgorithm();
295      resultBuffer.append("　ハッシュアルゴリズム：" + hashAlgorithm + "<br>");
296
297      /* 有効期限の取得 */
298      Date expirationDate = timeStampList[i]
299          .getTimeStampExpirationDate();
300      resultBuffer.append("　有効期限：" + expirationDate + "<br>");
301
302      /* タイムスタンプ生成日時の取得 */
303      Date createDate = timeStampList[i].getCreationDate();
304      resultBuffer.append("　タイムスタンプ生成日時：" + createDate + "<br>");
```

```
305
306     /* 検証結果の取得 */
307     int validateResult = timeStampList[i].getValidateResult();
308     resultBuffer.append("   検証結果：" + validateResult + "<br>");
309
310     /* 署名Vriが付加されているかの取得 */
311     boolean vriFlg = timeStampList[i].getVriFlg();
312     resultBuffer.append("   署名VRIの付加：" + vriFlg + "<br>");
313
314     /* タイムスタンプ情報の取得 */
315     PdfTimeStampToken token = timeStampList[i].getTimeStampToken();
316
317     /***************************************
318      * 取得情報出力：タイムスタンプトークン
319      ***************************************/
320     /* タイムスタンプデータ値の取得 */
321     byte[] tokenData = token.getData();
322     resultBuffer.append("   タイムスタンプデータ値：" + tokenData + "<br>");
323
324     /* 登録時のタイムスタンプハッシュ値の取得 */
325     byte[] registerDigest = token.getRegisterDigest();
326     resultBuffer.append("   登録時のタイムスタンプハッシュ値：" + registerDigest + "<br>");
327
328     /* 検証時のタイムスタンプハッシュ値の取得 */
329     byte[] digest = token.getDigest();
330     resultBuffer.append("   検証時のタイムスタンプハッシュ値：" + digest + "<br>");
331    }
332   } catch (PdfTimeStampException e) {
333    resultBuffer.append("文書タイムスタンプ付与：ERROR <br>");
334    resultBuffer.append("ステータス：" + e.getCode() + "<br>メッセージ："+ e.getMessage() + "
335 <br>");
336   }
337   finally {
338
339   }
340
341   return resultBuffer.toString();
342  }
343
344  /**
345   * 延長タイムスタンプの付与
346   */
347  String extendTimestamp() {
348
349   StringBuffer resultBuffer = new StringBuffer(200);
350
351   /* PDF情報 */
352   PdfDocument docinfo;
353
354   try {
355    /***************************************
356     * 実行準備
357     ***************************************/
358    /* 延長タイムスタンプを付与するPDFファイルのパス、及び権限パスワードを設定 */
359    service.setInputPdf(outpdfpath1, inputpdfpasswd);
360
361    /* 処理結果PDF出力先パスを設定 */
362    service.setOutputPdf(outpdfpath2);
363
364    /* タイムスタンプトークン取得用のハッシュアルゴリズムを設定 */
365    service.setHashAlgorithm(alg);
```

```
366
367     /* タイムスタンプトークン取得用のポリシーを設定 */
368     service.setPolicy(policy);
369
370     /* タイムスタンプ局の接続先のURLを設定 */
371     service.setTsaUrl(tsaurl);
372
373     /* タイムスタンプ局接続時の接続ID、パスワードを設定 */
374     service.setTsaUser(tsauser, tsapasswd);
375
376     /*****************************************
377      * 実行
378      *****************************************/
379     /* PDFに対し文書タイムスタンプを付与 */
380     service.generateLtv();
381
382     /*****************************************
383      * 実行結果出力
384      *****************************************/
385     resultBuffer.append("延長タイムスタンプ付与：SUCCESS<br>");
386
387     /*****************************************
388      * 実行
389      *****************************************/
390     /* PDFおよびタイムスタンプ情報の取得 */
391     docinfo = service.getPdfDocument();
392
393     /*****************************************
394      * 取得情報出力：PDF ドキュメント情報
395      *****************************************/
396     /* ファイルサイズの取得 */
397     long fileSize = docinfo.getContentSize();
398     resultBuffer.append("ファイルサイズ：" + fileSize + " Bytes<br>");
399
400     /* ページ数の取得 */
401     int pageNum = docinfo.getNumberOfPages();
402     resultBuffer.append("ページ数：" + pageNum + " Pages<br>");
403
404     /* ページサイズ(幅)の取得 */
405     double paperWidth = docinfo.getPaperWidth();
406     resultBuffer.append("ページ幅：" + paperWidth + " pts<br>");
407
408     /* ページサイズ(高さ)の取得 */
409     double paperHeight = docinfo.getPaperHeight();
410     resultBuffer.append("ページ高：" + paperHeight + " pts<br>");
411
412     /* 解像度取得 */
413     int resolution = docinfo.getResolution();
414     resultBuffer.append("解像度：" + resolution + " dpi<br>");
415
416     /* 階調取得 */
417     int gradation = docinfo.getGradation();
418     resultBuffer.append("階調：" + gradation + "<br>");
419
420     /* 最新のタイムスタンプ情報の取得 */
421     // PdfTimeStamp timeStamp = docinfo.getLatestTimeStamp();
422
423     /* タイムスタンプリストの取得 */
424     PdfTimeStamp[] timeStampList = docinfo.getTimeStamps();
425     resultBuffer.append("タイムスタンプリスト：" + timeStampList.length + "<br>");
426
```

```
427     /* VRI付きタイムスタンプリストの取得 */
428     PdfTimeStamp[] vriTimeStampList = docinfo.getTimeStampsWithVRI();
429     if (vriTimeStampList != null) {
430      resultBuffer.append("VRI付きタイムスタンプリスト：" + vriTimeStampList.length + "<br>");
431     }
432
433     /* VRI付きでないタイムスタンプリストの取得 */
434     PdfTimeStamp[] notVriTimeStampList = docinfo.getTimeStampsWithNotVRI();
435     if (notVriTimeStampList != null) {
436      resultBuffer.append("VRI付きでないタイムスタンプリスト：" + notVriTimeStampList.length + "
437    <br>");
438     }
439
440     /****************************************
441      * 取得情報出力：タイムスタンプ情報
442      ****************************************/
443     for (int i = 0; i < timeStampList.length; i++) {
444      resultBuffer.append("【TimeStamp[" + i + "]】<br>");
445
446      /* ハッシュアルゴリズムの取得 */
447      HASH_ALGORITHM hashAlgorithm = timeStampList[i]
448          .getHashAlgorithm();
449      resultBuffer.append("　ハッシュアルゴリズム：" + hashAlgorithm + "<br>");
450
451      /* 有効期限の取得 */
452      Date expirationDate = timeStampList[i]
453          .getTimeStampExpirationDate();
454      resultBuffer.append("　有効期限：" + expirationDate + "<br>");
455
456      /* タイムスタンプ生成日時の取得 */
457      Date createDate = timeStampList[i].getCreationDate();
458      resultBuffer.append("　タイムスタンプ生成日時：" + createDate + "<br>");
459
460      /* 検証結果の取得 */
461      int validateResult = timeStampList[i].getValidateResult();
462      resultBuffer.append("　検証結果：" + validateResult + "<br>");
463
464      /* 署名Vriが付加されているかの取得 */
465      boolean vriFlg = timeStampList[i].getVriFlg();
466      resultBuffer.append("　署名VRIの付加：" + vriFlg + "<br>");
467
468      /* タイムスタンプ情報の取得 */
469      PdfTimeStampToken token = timeStampList[i].getTimeStampToken();
470
471      /****************************************
472       * 取得情報出力：タイムスタンプトークン
473       ****************************************/
474      /* タイムスタンプデータ値の取得 */
475      byte[] tokenData = token.getData();
476      resultBuffer.append("　タイムスタンプデータ値：" + tokenData + "<br>");
477
478      /* 登録時のタイムスタンプハッシュ値の取得 */
479      byte[] registerDigest = token.getRegisterDigest();
480      resultBuffer.append("　登録時のタイムスタンプハッシュ値：" + registerDigest + "<br>");
481
482      /* 検証時のタイムスタンプハッシュ値の取得 */
483      byte[] digest = token.getDigest();
484      resultBuffer.append("　検証時のタイムスタンプハッシュ値：" + digest + "<br>");
485     }
486    } catch (PdfTimeStampException e) {
487     resultBuffer.append("延長タイムスタンプ付与：ERROR<br>");
```

```
488     resultBuffer.append("ステータス：" + e.getCode() + "<br>メッセージ：" + e.getMessage() + "
489 <br>");
490     }
491     finally{
492 
493     }
494 
495     return resultBuffer.toString();
496     }
497 
498     /**
499     * タイムスタンプの検証
500     */
501     String validateTimestamp() {
502 
503     StringBuffer resultBuffer = new StringBuffer(200);
504 
505     /* 検証結果 */
506     int res;
507 
508     try {
509     /*****************************************
510      * 実行準備
511      *****************************************/
512     /* タイムスタンプを検証するPDFファイルのパス、及び権限パスワードを設定 */
513     service.setInputPdf(outpdfpath2, inputpdfpasswd);
514 
515     /*****************************************
516      * 実行
517      *****************************************/
518     /* PDFのタイムスタンプを検証 */
519     res = service.validate();
520 
521     /*****************************************
522      * 実行結果出力
523      *****************************************/
524     resultBuffer.append("PDF文書のタイムスタンプ検証(validate)：SUCCESS：[" + res + "]<br>");
525     } catch (PdfTimeStampException e) {
526     resultBuffer.append("PDF文書のタイムスタンプ検証(validate)：ERROR<br>");
527     resultBuffer.append("ステータス：" + e.getCode() + "<br>メッセージ：" + e.getMessage() + "
528 <br>");
529     }
530     finally{
531 
532     }
533 
534     return resultBuffer.toString();
535     }
536 
537     String validateTsTimestamp() {
538 
539     StringBuffer resultBuffer = new StringBuffer(200);
540 
541     /* 検証結果 */
542     PdfTimeStamp[] timeStampList;
543 
544     try {
545     /*****************************************
546      * 実行準備
547      *****************************************/
548     /* タイムスタンプを検証するPDFファイルのパス、及び権限パスワードを設定 */
```

```
549     service.setInputPdf(outpdfpath2, inputpdfpasswd);
550
551     /*****************************************
552      * 実行
553      *****************************************/
554     /* PDFのタイムスタンプを検証*/
555     timeStampList = service.validateTs();
556
557     /*****************************************
558      * 取得情報出力：タイムスタンプ情報
559      *****************************************/
560     for (int i = 0; i < timeStampList.length; i++) {
561      resultBuffer.append("【TimeStamp[" + i + "]】<br>");
562
563      /* ハッシュアルゴリズムの取得*/
564      HASH_ALGORITHM hashAlgorithm = timeStampList[i]
565         .getHashAlgorithm();
566      resultBuffer.append("　ハッシュアルゴリズム：" + hashAlgorithm + "<br>");
567
568      /* 有効期限の取得*/
569      Date expirationDate = timeStampList[i]
570         .getTimeStampExpirationDate();
571      resultBuffer.append("　有効期限：" + expirationDate + "<br>");
572
573      /* タイムスタンプ生成日時の取得*/
574      Date createDate = timeStampList[i].getCreationDate();
575      resultBuffer.append("　タイムスタンプ生成日時：" + createDate + "<br>");
576
577      /* 検証結果の取得*/
578      int validateResult = timeStampList[i].getValidateResult();
579      resultBuffer.append("　検証結果：" + validateResult + "<br>");
580
581      /* 署名Vriが付加されているかの取得*/
582      boolean vriFlg = timeStampList[i].getVriFlg();
583      resultBuffer.append("　署名VRIの付加：" + vriFlg + "<br>");
584
585      /* タイムスタンプ情報の取得*/
586      PdfTimeStampToken token = timeStampList[i].getTimeStampToken();
587
588      /*****************************************
589       * 取得情報出力：タイムスタンプトークン
590       *****************************************/
591      /* タイムスタンプデータ値の取得*/
592      byte[] tokenData = token.getData();
593      resultBuffer.append("　タイムスタンプデータ値：" + tokenData + "<br>");
594
595      /* 登録時のタイムスタンプハッシュ値の取得*/
596      byte[] registerDigest = token.getRegisterDigest();
597      resultBuffer.append("　登録時のタイムスタンプハッシュ値：" + registerDigest + "<br>");
598
599      /* 検証時のタイムスタンプハッシュ値の取得*/
600      byte[] digest = token.getDigest();
601      resultBuffer.append("　検証時のタイムスタンプハッシュ値：" + digest + "<br>");
602     }
603     } catch (PdfTimeStampException e) {
604      resultBuffer.append("PDF文書のタイムスタンプ検証(validateTs)：ERROR<br>");
605      resultBuffer.append("ステータス：" + e.getCode() + "<br>メッセージ：" + e.getMessage() + "
606    <br>");
607     }
608     finally{
609
```

```
610     }
611
612     return resultBuffer.toString();
613     }
614
615     /**
616     * PDF ドキュメント、及びタイムスタンプ情報の取得
617     */
618     String getInfo() {
619
620     StringBuffer resultBuffer = new StringBuffer(200);
621
622     /* PDF情報 */
623     PdfDocument docinfo;
624
625     try {
626     /*****************************************
627      * 実行準備
628      *****************************************/
629     /* 情報を取得するPDFファイルのパス、及び権限パスワードを設定 */
630     service.setInputPdf(outpdfpath2, inputpdfpasswd);
631
632     /*****************************************
633      * 実行
634      *****************************************/
635     /* PDFおよびタイムスタンプ情報の取得 */
636     docinfo = service.getPdfDocument();
637
638     /*****************************************
639      * 取得情報出力：PDF ドキュメント情報
640      *****************************************/
641     /* ファイルサイズの取得 */
642     long fileSize = docinfo.getContentSize();
643     resultBuffer.append("ファイルサイズ：" + fileSize + " Bytes<br>");
644
645     /* ページ数の取得 */
646     int pageNum = docinfo.getNumberOfPages();
647     resultBuffer.append("ページ数：" + pageNum + " Pages<br>");
648
649     /* ページサイズ(幅)の取得 */
650     double paperWidth = docinfo.getPaperWidth();
651     resultBuffer.append("ページ幅：" + paperWidth + " pts<br>");
652
653     /* ページサイズ(高さ)の取得 */
654     double paperHeight = docinfo.getPaperHeight();
655     resultBuffer.append("ページ高：" + paperHeight + " pts<br>");
656
657     /* 解像度取得 */
658     int resolution = docinfo.getResolution();
659     resultBuffer.append("解像度：" + resolution + " dpi<br>");
660
661     /* 階調取得 */
662     int gradation = docinfo.getGradation();
663     resultBuffer.append("階調：" + gradation + "<br>");
664
665     /* 最新のタイムスタンプ情報の取得 */
666     //PdfTimeStamp timeStamp = docinfo.getLatestTimeStamp();
667
668     /* タイムスタンプリストの取得 */
669     PdfTimeStamp[] timeStampList = docinfo.getTimeStamps();
670
```

```
671     /* タイムスタンプが付与されていない場合は処理終了 */
672     if (timeStampList == null) {
673      resultBuffer.append("タイムスタンプが付与されていません。<br>");
674      resultBuffer.append("PDF文書の情報取得：SUCCESS<br>");
675      return resultBuffer.toString();
676     }
677
678     /* 付与されているタイムスタンプを出力 */
679     resultBuffer.append("タイムスタンプリスト：" + timeStampList.length + "<br>");
680
681     /* VRI付きタイムスタンプリストの取得 */
682     PdfTimeStamp[] vriTimeStampList = docinfo.getTimeStampsWithVRI();
683     if (vriTimeStampList != null) {
684      resultBuffer.append("VRI付きタイムスタンプリスト：" + vriTimeStampList.length + "<br>");
685     }
686
687     /* VRI付きでないタイムスタンプリストの取得 */
688     PdfTimeStamp[] notVriTimeStampList = docinfo.getTimeStampsWithNotVRI();
689     if (notVriTimeStampList != null) {
690      resultBuffer.append("VRI付きでないタイムスタンプリスト：" + notVriTimeStampList.length + "
691 <br>");
692     }
693
694     /****************************************
695      * 取得情報出力：タイムスタンプ情報
696      ****************************************/
697     for (int i = 0; i < timeStampList.length; i++) {
698      resultBuffer.append("【TimeStamp[" + i + "]】<br>");
699
700      /* ハッシュアルゴリズムの取得 */
701      HASH_ALGORITHM hashAlgorithm = timeStampList[i]
702         .getHashAlgorithm();
703      resultBuffer.append("　ハッシュアルゴリズム：" + hashAlgorithm + "<br>");
704
705      /* 有効期限の取得 */
706      Date expirationDate = timeStampList[i]
707         .getTimeStampExpirationDate();
708      resultBuffer.append("　有効期限：" + expirationDate + "<br>");
709
710      /* タイムスタンプ生成日時の取得 */
711      Date createDate = timeStampList[i].getCreationDate();
712      resultBuffer.append("　タイムスタンプ生成日時：" + createDate + "<br>");
713
714      /* 検証結果の取得 */
715      int validateResult = timeStampList[i].getValidateResult();
716      resultBuffer.append("　検証結果：" + validateResult + "<br>");
717
718      /* 署名Vriが付加されているかの取得 */
719      boolean vriFlg = timeStampList[i].getVriFlg();
720      resultBuffer.append("　署名VRIの付加：" + vriFlg + "<br>");
721
722      /* タイムスタンプ情報の取得 */
723      PdfTimeStampToken token = timeStampList[i].getTimeStampToken();
724
725      /****************************************
726       * 取得情報出力：タイムスタンプトークン
727       ****************************************/
728      /* タイムスタンプデータ値の取得 */
729      byte[] tokenData = token.getData();
730      resultBuffer.append("　タイムスタンプデータ値：" + tokenData + "<br>");
731
```

```
732     /* 登録時のタイムスタンプハッシュ値の取得 */
        byte[] registerDigest = token.getRegisterDigest();
        resultBuffer.append("　登録時のタイムスタンプハッシュ値：" + registerDigest + "<br>");

        /* 検証時のタイムスタンプハッシュ値の取得 */
        byte[] digest = token.getDigest();
        resultBuffer.append("　検証時のタイムスタンプハッシュ値：" + digest + "<br>");
       }

       /*****************************************
        * 実行結果出力
        *****************************************/
       resultBuffer.append("PDF文書の情報取得：SUCCESS<br>");
      } catch (PdfTimeStampException e) {
       resultBuffer.append("PDF文書の情報取得：ERROR<br>");
       resultBuffer.append("ステータス：" + e.getCode() + "<br>メッセージ：" + e.getMessage() + "
<br>");
      }
      finally{

      }

      return resultBuffer.toString();
     }
    %>
```

> **コラム**
> タイムスタンプ局のURL、ユーザID、および、パスワード情報は環境に合わせて指定してください。

> **コラム**
>
> タイムスタンプ局のポリシーは、タイムスタンプ局のURLのクエリパラメータで指定することも可能です。
>
> - TypeA3（”1.3.6.1.4.1.955.1.10.1.5.3”）の場合
>   - https://pades01.seiko-cybertime.jp/basic/Timestamp?type=AccreditedA3

> **注意**
> 文字コードを UTF-8 にして保存してください。

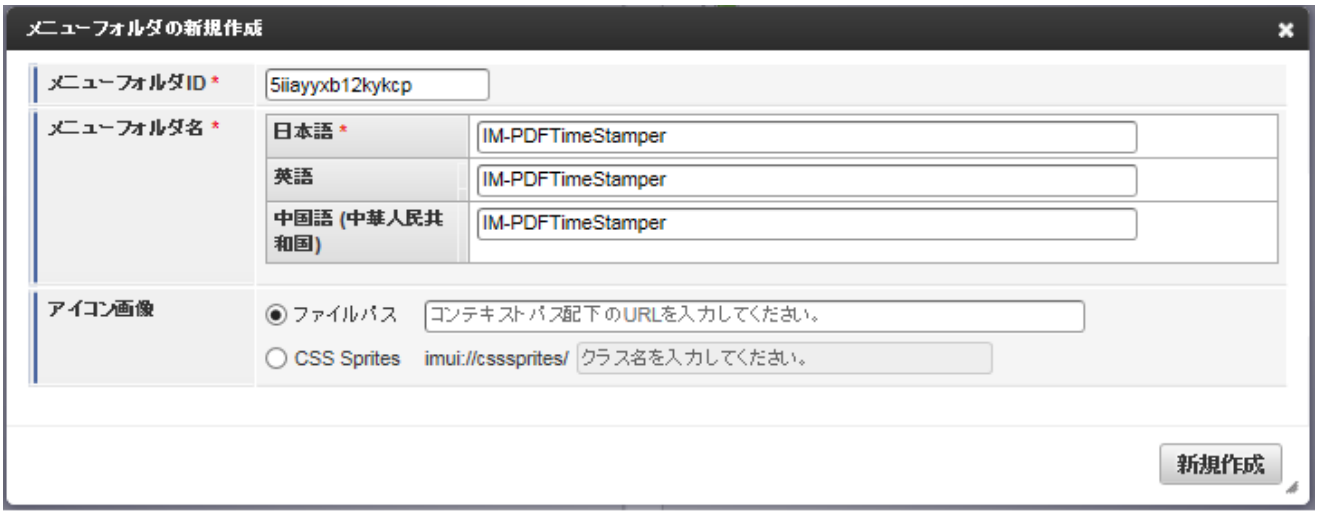
プログラム実行

準備

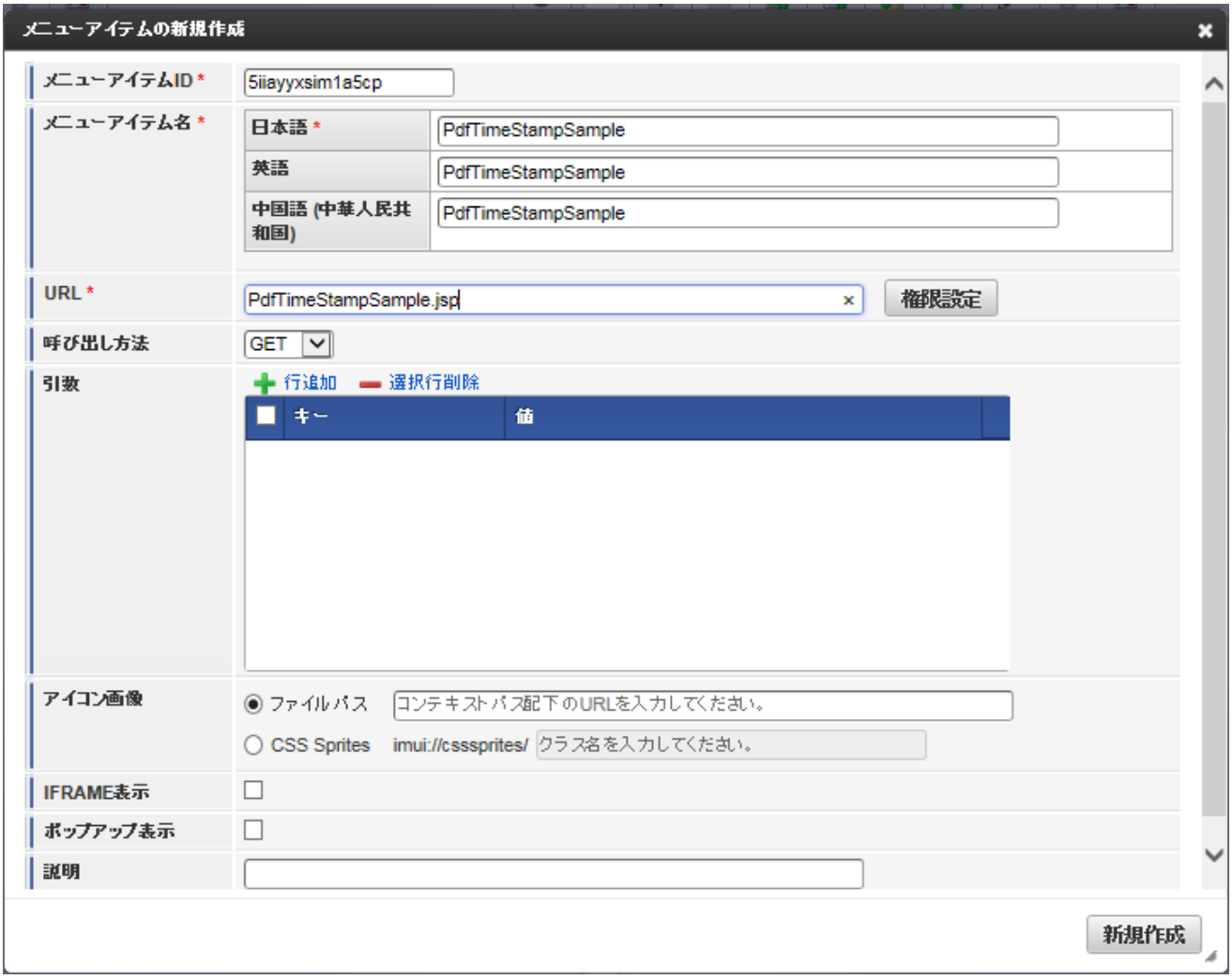
実行させるための準備の手順を説明します。

メニュー設定

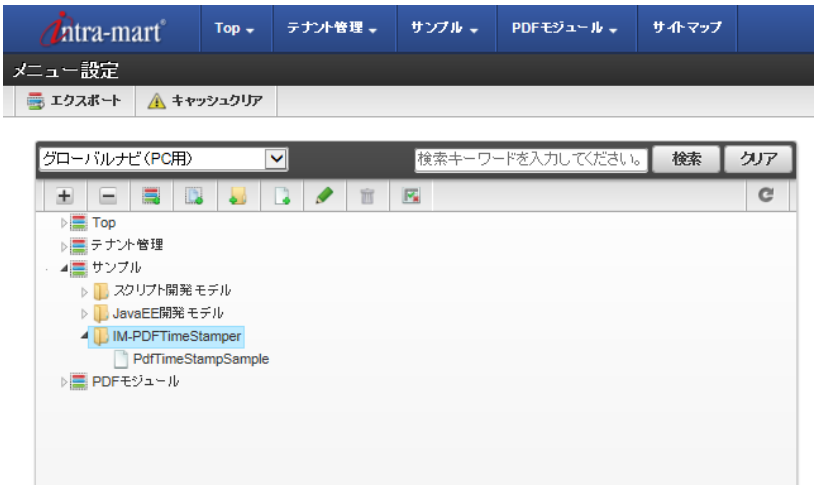
1. テナント管理者でログインし、以下のメニューを設定します。
2. [テナント管理]-[メニュー]画面を開きます。
3. フォルダを作成します。



4. URLに、PdfTimeStampSample.jsp を設定し、メニューを追加します。



5. メニュー設定は完了です。



プログラム実行

メニューで『PdfTimeStampSample』を選択してください。作成したJSPファイルが実行されます。

JSPの実行エラー（コンパイルエラー）になってしまった場合には、エラーメッセージの内容に従いJSPプログラムを修正してください。

JSPプログラムが正しく動作しているにも関わらず実行時エラーになってしまう場合は、エラーの内容にしたがって環境を正しく構築してください（環境を変更した場合は、サーバの再起動が必要になる場合があります）。

確認

プログラムが正しく実行されると、タイムスタンプ情報が画面に表示され、 C:/temp ディレクトリに out1.pdf 、out2.pdf というPDFファイルが作成されます。

このファイルにタイムスタンプ情報が付与されており、PDFビューア（Adobe AcrobatReader など）で正しく表示できればすべての処理が正しく行われたことになります。

サンプルプログラム

- **PdfTimeStampSample.jsp**

ステータスコード

目次

ステータスコード一覧

ステータスコード	内容
0	付与されているタイムスタンプは有効です。
1	付与されているタイムスタンプの有効期限が切れています。
2	付与されているタイムスタンプのデータが改竄されています。
3	付与されているタイムスタンプが失効しています。
-1	付与されているタイムスタンプは未検証です。
-101	タイムスタンプを付与するPDFが存在しません。
-102	タイムスタンプを付与するPDFの読み込みに失敗しました。
-103	タイムスタンプを付与したPDFの出力に失敗しました。
-104	タイムスタンプ局への接続に失敗しました。
-105	タイムスタンプトークンの取得に失敗しました。
-106	タイムスタンプトークンの埋め込みに失敗しました。
-107	PDFにタイムスタンプが付与されていません。
-108	タイムスタンプの取得に失敗しました。
-109	LTVの生成に失敗しました。
-110	CRL配布ポイントへの接続に失敗しました。
-111	CRLの取得に失敗しました。
-112	付与されているタイムスタンプが失効しています。
-113	PDF情報の取得に失敗しました。
-114	処理対象外の形式のタイムスタンプが付与されています。
-115	PDFタイムスタンプサービスのリモートサーバへの接続に失敗しました。
-116	一時ディレクトリの作成に失敗しました。
-117	一時ファイルの作成に失敗しました。
-118	PDFセキュリティの付与に失敗しました。
-119	タイムスタンプの検証に失敗しました。
-120	必須パラメータが設定されていません。
-121	変換元の画像ファイルの拡張子が「.jpg」か「.jpeg」ではありません。
-122	変換元の画像ファイルが存在しません。

ステータスコード	内容
-123	変換元の画像ファイルの読み込みに失敗しました。
-124	解像度に0以下の値が指定されています。
-125	画像ファイルのPDF変換に失敗しました。
-126	画像ファイルを変換したPDFの出力に失敗しました。
-127	オブジェクト識別子(OID)に不正な値が指定されています。
-999	予期しないエラーが発生しました。

注意

次のエラーが発生した場合、「Password error」と表示されますが、パスワードに起因するエラーではないケースがあります。

```
yss.iothe.pdftimestamp.PdfTimeStampException: 処理対象PDFの読み込みに失敗しました。
「Password error」
```

上記のエラーが発生する主なPDFファイルの形式やケースは、次の通りです。

- パスワードが付与され、暗号化されているPDFファイル
- 電子署名やタイムスタンプ等が付与されているPDFファイル
- Adobe Acrobat の拡張機能等が使用されているPDFファイル
- 内部構造が一部破損しているPDFファイル
- PDFの規格に準拠していないPDFファイル

サポート

弊社では、Web にて弊社製品に対するサポートおよび技術情報の公開を行っております。

当製品に関して不明な点などがございましたら、情報検索または弊社サポート窓口までご相談ください。