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

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。

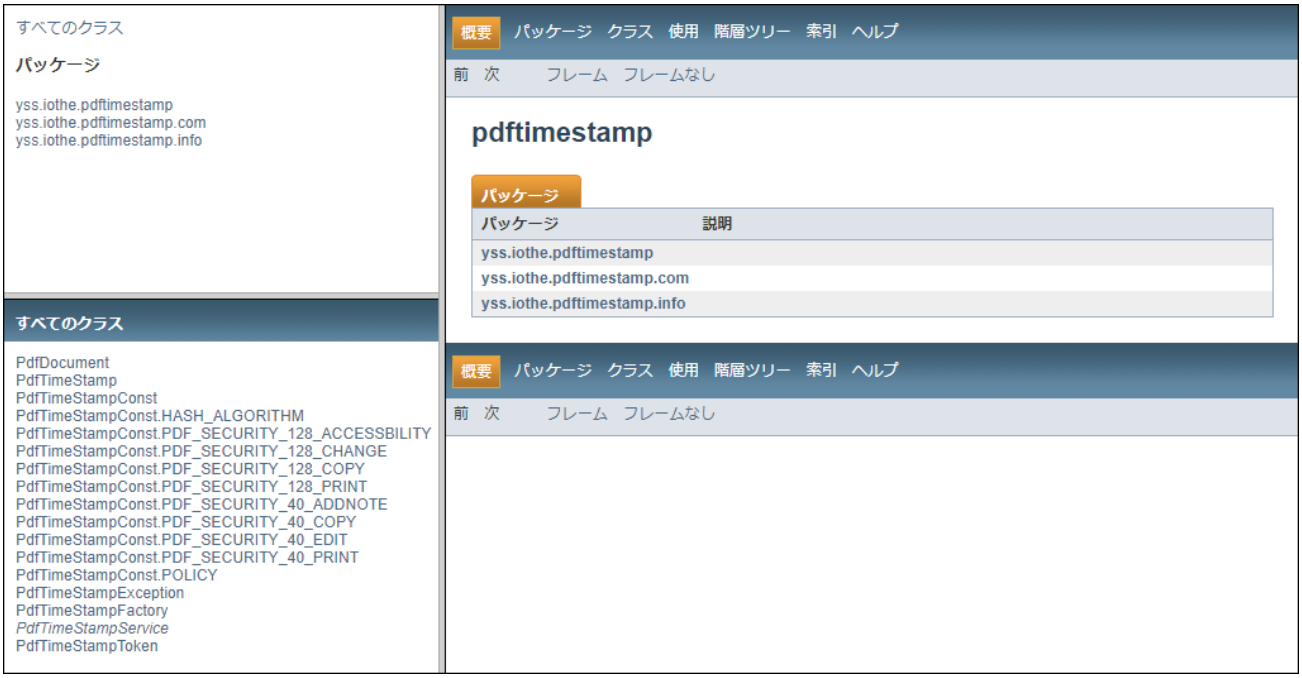
スクリプト開発モデル で開発をする場合は、スクリプト開発モデル のソースコード内でJavaのクラスを呼び出してください。

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.TYPEA2;

    /****************************************
     * タイムスタンプ局の設定
     ****************************************/
    /* タイムスタンプ局の接続先のURL */
    private static final String tsaurl = "https://timestamp.seiko-cybertime.jp/basic/Timestamp?
type=AccreditedA2";

    /* 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>
```

```
<%!
/**
 * 文書タイムスタンプの付与
 */
String addTimestamp() {

 StringBuffer resultBuffer = new StringBuffer(200);

 /* PDF情報 */
 PdfDocument docinfo;

 try {
  /*****************************************
   * 実行準備
   *****************************************/
  /* 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 */
  service.setInputPdf(inputpdfpath, inputpdfpasswd);

  /* 処理結果PDF出力先パスを設定 */
  service.setOutputPdf(outpdfpath1);

  /* 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 */
  service.setHashAlgorithm(alg);

  /* タイムスタンプトークン取得用のポリシーを設定 */
  service.setPolicy(policy);

  /* タイムスタンプ局の接続先のURLを設定 */
  service.setTsaUrl(tsaurl);

  /* タイムスタンプ局接続時の接続ID、パスワードを設定 */
  service.setTsaUser(tsauser, tsapasswd);

  /* 出力PDFのセキュリティ設定(40/128のどちらかを指定) */
  /*
  service.setSecurity40(openpasswd, secpasswd,
     security40_print, security40_edit,
     security40_copy, security40_addnote);
  */
  service.setSecurity128(openpasswd, secpasswd,
     security128_print, security128_access,
     security128_copy, security128_change);

  /*****************************************
   * 実行
   *****************************************/
  /* PDFに対し文書タイムスタンプを付与 */
  service.generate();

  resultBuffer.append("文書タイムスタンプ付与：SUCCESS<br>");

  /*****************************************
   * 実行
   *****************************************/
  /* PDFおよびタイムスタンプ情報の取得 */
  docinfo = service.getPdfDocument();

  /*****************************************
   * 取得情報出力：PDFドキュメント情報
   *****************************************/
  /* ファイルサイズの取得 */
```

```
    long fileSize = docinfo.getContentSize();
    resultBuffer.append("ファイルサイズ：" + fileSize + " Bytes<br>");

    /* ページ数の取得 */
    int pageNum = docinfo.getNumberOfPages();
    resultBuffer.append("ページ数：" + pageNum + " Pages<br>");

    /* ページサイズ(幅)の取得 */
    double paperWidth = docinfo.getPaperWidth();
    resultBuffer.append("ページ幅：" + paperWidth + " pts<br>");

    /* ページサイズ(高さ)の取得 */
    double paperHeight = docinfo.getPaperHeight();
    resultBuffer.append("ページ高：" + paperHeight + " pts<br>");

    /* 解像度取得 */
    int resolution = docinfo.getResolution();
    resultBuffer.append("解像度：" + resolution + " dpi<br>");

    /* 階調取得 */
    int gradation = docinfo.getGradation();
    resultBuffer.append("階調：" + gradation + "<br>");

    /* 最新のタイムスタンプ情報の取得 */
    // PdfTimeStamp timeStamp = docinfo.getLatestTimeStamp();

    /* タイムスタンプリストの取得 */
    PdfTimeStamp[] timeStampList = docinfo.getTimeStamps();
    resultBuffer.append("タイムスタンプリスト：" + timeStampList.length + "<br>");

    /* VRI付きタイムスタンプリストの取得 */
    PdfTimeStamp[] vriTimeStampList = docinfo.getTimeStampsWithVRI();
    if (vriTimeStampList != null) {
     resultBuffer.append("VRI付きタイムスタンプリスト：" + vriTimeStampList.length + "<br>");
    }

    /* VRI付きでないタイムスタンプリストの取得 */
    PdfTimeStamp[] notVriTimeStampList = docinfo.getTimeStampsWithNotVRI();
    if (notVriTimeStampList != null) {
     resultBuffer.append("VRI付きでないタイムスタンプリスト：" + notVriTimeStampList.length + "
<br>");
    }

    /*****************************************
     * 取得情報出力：タイムスタンプ情報
     *****************************************/
    for (int i = 0; i < timeStampList.length; i++) {
     resultBuffer.append("【TimeStamp[" + i + "]】<br>");

     /* ハッシュアルゴリズムの取得 */
     HASH_ALGORITHM hashAlgorithm = timeStampList[i]
        .getHashAlgorithm();
     resultBuffer.append("　ハッシュアルゴリズム：" + hashAlgorithm + "<br>");

     /* 有効期限の取得 */
     Date expirationDate = timeStampList[i]
        .getTimeStampExpirationDate();
     resultBuffer.append("　有効期限：" + expirationDate + "<br>");

     /* タイムスタンプ生成日時の取得 */
     Date createDate = timeStampList[i].getCreationDate();
```

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

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

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

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

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

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

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

```
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。

注意

文字コードを UTF-8 にして保存してください。

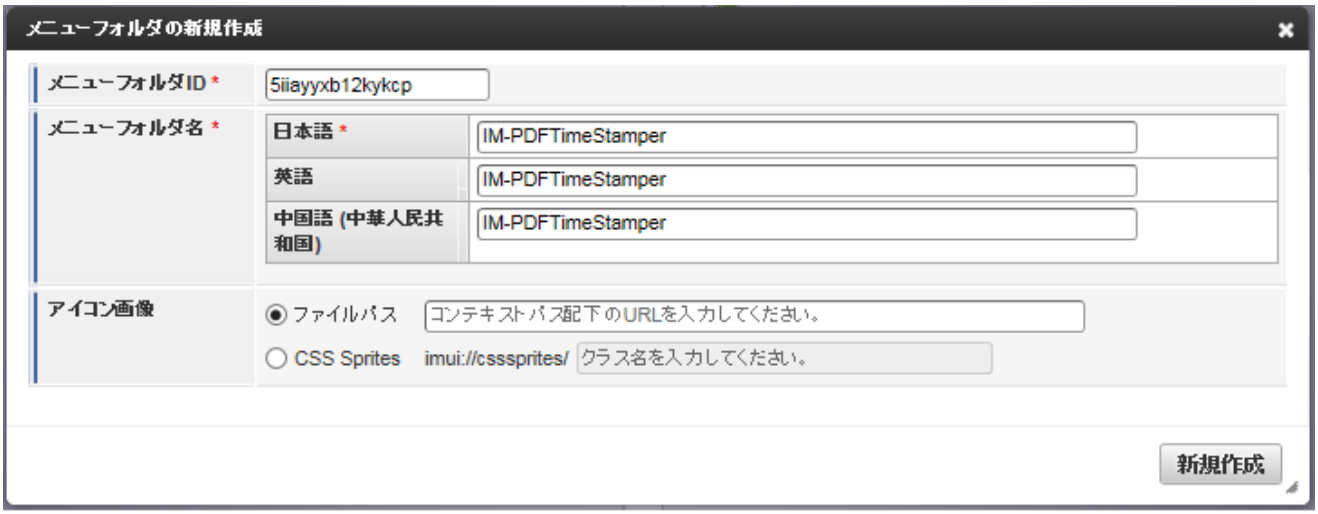
プログラム実行

準備

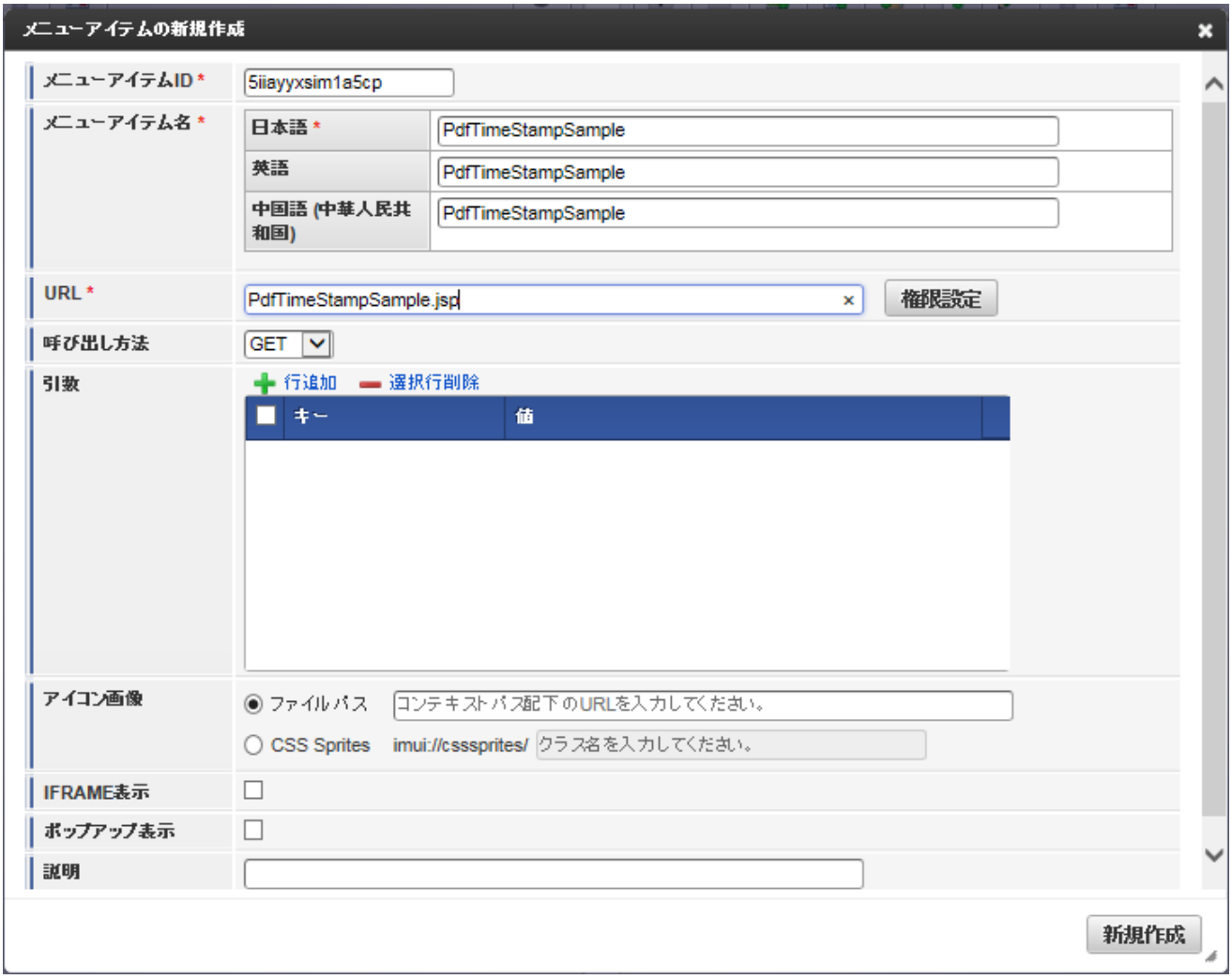
実行させるための準備の手順を説明します。

メニュー設定

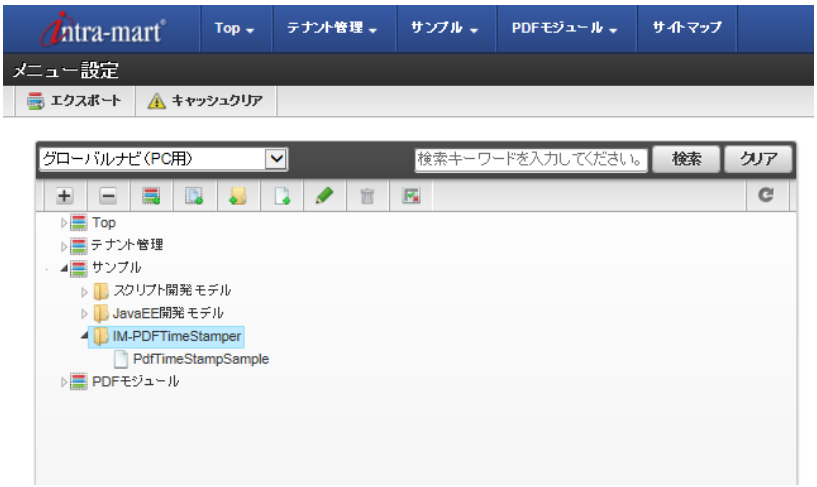
1. テナント管理者でログインし、以下のメニューを設定します。
2. [テナント管理]-[メニュー]画面を開きます。
3. フォルダを作成します。



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。
-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。
-119	タイムスタンプの検証に失敗に失敗しました。
-120	必須パラメータが設定されていません。
-125	画像ファイルのPDF変換に失敗しました。
-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。