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。

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

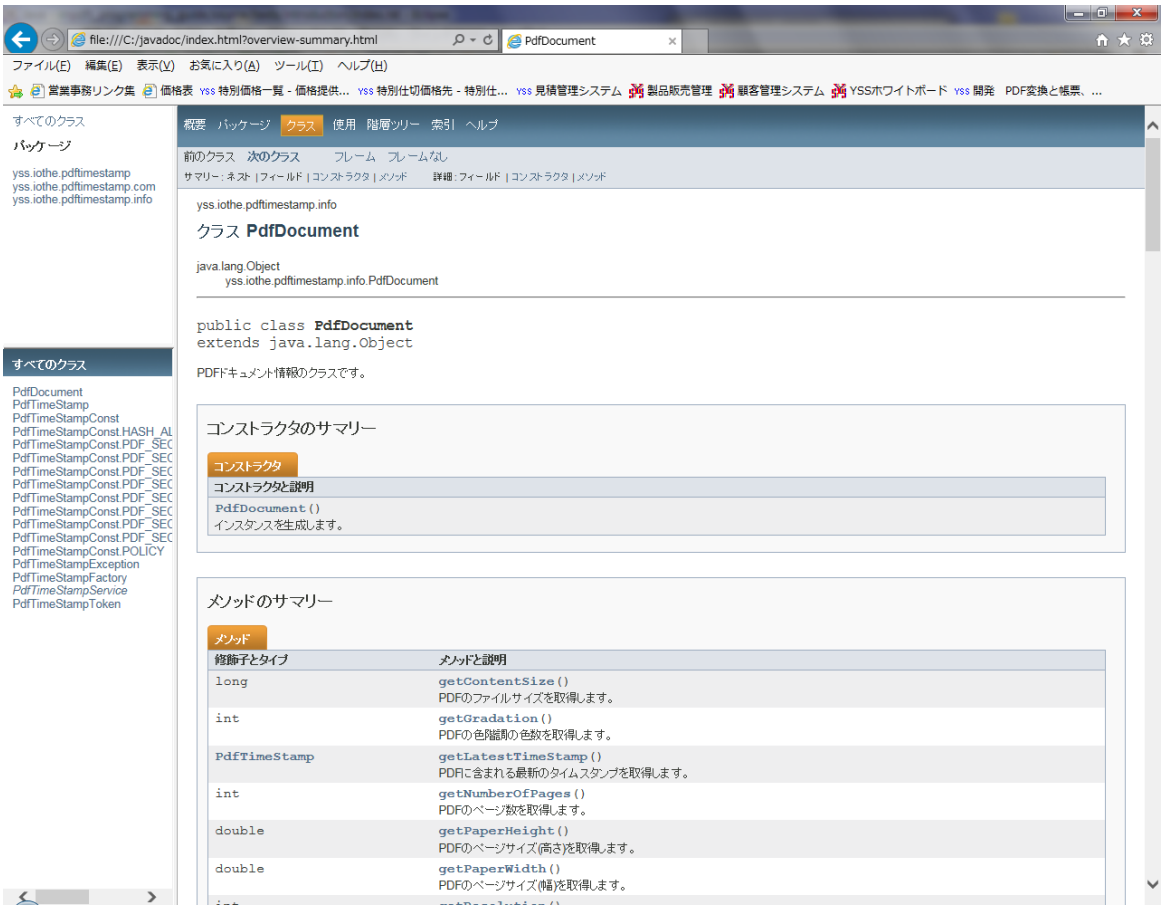
目次

APIリストについて

本製品には、IM-PDFTimeStamper for Accel Platform 専用のAPI リストが付属します。
API リストは、document/apilist.zip にあります。このファイルは、ZIP で圧縮されていますので、任意のZIP解凍ツールで解凍してください。 解凍するときは、ディレクトリ付きで解凍してください。
アーカイブファイルを解凍後 apilist/index.html をブラウザで開くと、API リストを閲覧できます。
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。

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

このチュートリアルでは、 JavaEE開発モデル におけるプログラミングの方法について説明します。
このチュートリアルでは、タイムスタンプ付与処理のサンプルを作成します。

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。s

環境

チュートリアルを学ぶための環境です。
このドキュメント内では、ここで示す環境を前提として解説しています。

サーバ環境

intra-mart Accel Platform と IM-PDFTimeStamper for Accel Platform が、正常にインストールされていることを前提とします。

タイムスタンプ処理サーバ 環境

サーバには IM-PDFTimeStamper for Accel Platform が正しくインストールされ、APIが正常に動作している状態であることが前提です。
サーバは Windows Server 2012 で動作しているものとして説明をします。

また、Java はOracle JDK がインストールされているものとします。

準備

このドキュメントではC:¥temp をプログラム作成の作業領域として説明しています。
このフォルダが存在しない場合には、予め作成しておいてください。
別なフォルダで作業をする場合には、その環境に合わせてドキュメントを読みすすめてください。
プログラム作成には、テキストエディタが必要です。プログラム作成のできるテキストエディタをご用意ください。

サンプルプログラムの場所

document/tutorial/PdfTimeStampSample.jsp
にサンプルプログラムを用意しておりますのでご覧ください。

サンプルデータの用意

C:\temp\in.pdf ファイルを用意ください。

JSPプログラムの作成

このサンプルでは、JSPから IM-PDFTimeStamper for Accel Platform の タイムスタンプ付与等の各機能を呼び出します。

テキストエディタを起動して、以下のプログラムを記述します。
この時、ファイル名の大文字・小文字は厳密な意味を持ちますので、注意してください。

作成した JSPファイルは、以下のフォルダに保存してください。

ファイル名	保存場所
PdfTimeStampSample.jsp	%RESIN_HOME%/webapps/{アプリケーション名}/PdfTimeStampSample.jsp

```
<%@ 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" %>
<%!
```

```
 /**
  * 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;
%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="X-UA-Compatible" content="IE=edge">
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>IM-PDFTimeStamperサンプル</title>
</head>
<body>

 <DIV align="center" style="center; padding-top: 25px;">
 <TABLE border="1">
  <TR>
   <TH align="center" style="padding: 5px 10px;" nowrap>文書タイムスタンプ付与</TH>
  </TR>
  <TR>
   <TD align="left" style="padding: 5px 10px;" nowrap><%= addTimestamp() %></TD>
  </TR>
 </TABLE>
 </DIV>

 <DIV align="center" style="center; padding-top: 25px;">
 <TABLE border="1">
  <TR>
   <TH align="center" style="padding: 5px 10px;" nowrap>延長タイムスタンプ付与</TH>
  </TR>
  <TR>
   <TD align="left" style="padding: 5px 10px;" nowrap><%= extendTimestamp() %></TD>
  </TR>
 </TABLE>
```

```
</DIV>

<DIV align="center" style="center; padding-top: 25px;">
<TABLE border="1">
 <TR>
  <TH align="center" style="padding: 5px 10px;" nowrap>PDF文書のタイムスタンプ検証(validate)</TH>
 </TR>
 <TR>
  <TD align="left" style="padding: 5px 10px;" nowrap><%= validateTimestamp() %></TD>
 </TR>
</TABLE>
</DIV>

<DIV align="center" style="center; padding-top: 25px;">
<TABLE border="1">
 <TR>
  <TH align="center" style="padding: 5px 10px;" nowrap>PDF文書のタイムスタンプ検証(validateTs)</TH>
 </TR>
 <TR>
  <TD align="left" style="padding: 5px 10px;" nowrap><%= validateTsTimestamp() %></TD>
 </TR>
</TABLE>
</DIV>

<DIV align="center" style="center; padding-top: 25px;">
<TABLE border="1">
 <TR>
  <TH align="center" style="padding: 5px 10px;" nowrap>PDF文書の情報取得</TH>
 </TR>
 <TR>
  <TD align="left" style="padding: 5px 10px;" nowrap><%= getInfo() %></TD>
 </TR>
</TABLE>
</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("addTimestamp : 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("File Size: " + fileSize + " Bytes<br>");

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

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

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

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

/* 解像度取得 */
int gradation = docinfo.getGradation();
resultBuffer.append("Gradation: " + 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("TimeStamp Length: " + 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 TimeStamp Length: " + 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("NotVRI TimeStamp Length: " + 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("HashAlgorithm: " + hashAlgorithm + "<br>");

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

 /* タイムスタンプ生成日時の取得 */
 Date createDate = timeStampList[i].getCreationDate();
 resultBuffer.append("CreateDate: " + createDate + "<br>");

 /* 検証結果の取得 */
 int validateResult = timeStampList[i].getValidateResult();
 resultBuffer.append("validateResult: " + validateResult + "<br>");

 /* 署名Vriが付加されているかの取得 */
 boolean vriFlg = timeStampList[i].getVriFlg();
 resultBuffer.append("vriFlg: " + vriFlg + "<br>");

 /* タイムスタンプ情報の取得 */
 PdfTimeStampToken token = timeStampList[i].getTimeStampToken();

 /****************************************
  * 取得情報出力：タイムスタンプトークン
  ****************************************/
 /* タイムスタンプデータ値の取得 */
 byte[] tokenData = token.getData();
 resultBuffer.append("Token Data: " + tokenData + "<br>");

 /* 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("Token RegisterDigest: " + 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("Token Digest: " + digest + "<br>");
```

```
 }
 } catch (PdfTimeStampException e) {
  resultBuffer.append("addTimestamp : ERROR <br>");
  resultBuffer.append("status : " + e.getCode() + ", message : "+ e.getMessage() + "<br>");
 }
 finally {

 }

 return resultBuffer.toString();
}

/**
 * 延長タイムスタンプの付与
 */
String extendTimestamp() {

 StringBuffer resultBuffer = new StringBuffer(200);

 /* PDF情報 */
 PdfDocument docinfo;

 try {
  /*****************************************
   * 実行準備
   *****************************************/
  /* 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 */
  service.setInputPdf(outpdfpath1, inputpdfpasswd);

  /* 処理結果PDF出力先パスを設定 */
  service.setOutputPdf(outpdfpath2);

  /* 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);

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

  /*****************************************
   * 実行結果出力
   *****************************************/
  resultBuffer.append("extendTimestamp : 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("File Size: " + fileSize + " Bytes<br>");

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

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

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

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

/* 解像度取得 */
int gradation = docinfo.getGradation();
resultBuffer.append("Gradation: " + 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("TimeStamp Length: " + 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 TimeStamp Length: " + 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("NotVRI TimeStamp Length: " + 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("HashAlgorithm: " + hashAlgorithm + "<br>");

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

```
  resultBuffer.append("ExpirationDate: " + expirationDate + "<br>");

  /* タイムスタンプ生成日時の取得 */
  Date createDate = timeStampList[i].getCreationDate();
  resultBuffer.append("CreateDate: " + createDate + "<br>");

  /* 検証結果の取得 */
  int validateResult = timeStampList[i].getValidateResult();
  resultBuffer.append("validateResult: " + validateResult + "<br>");

  /* 署名Vriが付加されているかの取得 */
  boolean vriFlg = timeStampList[i].getVriFlg();
  resultBuffer.append("vriFlg: " + vriFlg + "<br>");

  /* タイムスタンプ情報の取得 */
  PdfTimeStampToken token = timeStampList[i].getTimeStampToken();

  /***************************************
   * 取得情報出力：タイムスタンプトークン
   ***************************************/
  /* タイムスタンプデータ値の取得 */
  byte[] tokenData = token.getData();
  resultBuffer.append("Token Data: " + tokenData + "<br>");

  /* 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("Token RegisterDigest: " + 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("Token Digest: " + digest + "<br>");
 }
 } catch (PdfTimeStampException e) {
  resultBuffer.append("extendTimestamp : ERROR<br>");
  resultBuffer.append("status : " + e.getCode() + ", message : " + e.getMessage() + "<br>");
 }
 finally{

 }

 return resultBuffer.toString();
}

/**
 * タイムスタンプの検証
 */
String validateTimestamp() {

 StringBuffer resultBuffer = new StringBuffer(200);

 /* 検証結果 */
 int res;

 try {
  /***************************************
   * 実行準備
   ***************************************/
  /* 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 */
  service.setInputPdf(outpdfpath2, inputpdfpasswd);

  /***************************************
   * 実行
```

```
 ********************************************/
 /* PDFのタイムスタンプを検証 */
 res = service.validate();

 /*******************************************
  * 実行結果出力
  *******************************************/
 resultBuffer.append("validateTimestamp : SUCCESS : [" + res + "]<br>");
 } catch (PdfTimeStampException e) {
 resultBuffer.append("validateTimestamp : ERROR<br>");
 resultBuffer.append("status : " + e.getCode() + ", message : " + e.getMessage() + "<br>");
 }
 finally{

 }

 return resultBuffer.toString();
}

String validateTsTimestamp() {

 StringBuffer resultBuffer = new StringBuffer(200);

 /* 検証結果 */
 PdfTimeStamp[] timeStampList;

 try {
  /*******************************************
   * 実行準備
   *******************************************/
  /* 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 */
  service.setInputPdf(outpdfpath2, inputpdfpasswd);

  /*******************************************
   * 実行
   *******************************************/
  /* PDFのタイムスタンプを検証 */
  timeStampList = service.validateTs();

  /*******************************************
   * 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("HashAlgorithm: " + hashAlgorithm + "<br>");

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

   /* タイムスタンプ生成日時の取得 */
   Date createDate = timeStampList[i].getCreationDate();
   resultBuffer.append("CreateDate: " + createDate + "<br>");

   /* 検証結果の取得 */
   int validateResult = timeStampList[i].getValidateResult();
```

```
  int validateResult = timeStampList[i].getValidateResult();
  resultBuffer.append("validateResult: " + validateResult + "<br>");

  /* 署名Vriが付加されているかの取得 */
  boolean vriFlg = timeStampList[i].getVriFlg();
  resultBuffer.append("vriFlg: " + vriFlg + "<br>");

  /* タイムスタンプ情報の取得 */
  PdfTimeStampToken token = timeStampList[i].getTimeStampToken();

  /*****************************************
   * 取得情報出力：タイムスタンプトークン
   *****************************************/
  /* タイムスタンプデータ値の取得 */
  byte[] tokenData = token.getData();
  resultBuffer.append("Token Data: " + tokenData + "<br>");

  /* 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("Token RegisterDigest: " + 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("Token Digest: " + digest + "<br>");
 }
 } catch (PdfTimeStampException e) {
  resultBuffer.append("validateTsTimestamp : ERROR<br>");
  resultBuffer.append("status : " + e.getCode() + ", message : " + e.getMessage() + "<br>");
 }
 finally{

 }

 return resultBuffer.toString();
}

/**
 * PDFドキュメント、及びタイムスタンプ情報の取得
 */
String getInfo() {

 StringBuffer resultBuffer = new StringBuffer(200);

 /* PDF情報 */
 PdfDocument docinfo;

 try {
  /*****************************************
   * 実行準備
   *****************************************/
  /* 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 */
  service.setInputPdf(outpdfpath2, inputpdfpasswd);

  /*****************************************
   * 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("File Size: " + fileSize + " Bytes<br>");

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

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

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

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

/* 解像度取得 */
int gradation = docinfo.getGradation();
resultBuffer.append("Gradation: " + 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();

/* タイムスタンプが付与されていない場合は処理終了 */
if (timeStampList == null) {
 resultBuffer.append("タイムスタンプが付与されていません。<br>");
 resultBuffer.append("getInfo : SUCCESS<br>");
 return resultBuffer.toString();
}

/* 付与されているタイムスタンプを出力 */
resultBuffer.append("TimeStamp Length: " + 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 TimeStamp Length: " + 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("NotVRI TimeStamp Length: " + 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();
```

```
    .getHashAlgorithm();
  resultBuffer.append("HashAlgorithm: " + hashAlgorithm + "<br>");

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

  /* タイムスタンプ生成日時の取得 */
  Date createDate = timeStampList[i].getCreationDate();
  resultBuffer.append("CreateDate: " + createDate + "<br>");

  /* 検証結果の取得 */
  int validateResult = timeStampList[i].getValidateResult();
  resultBuffer.append("validateResult: " + validateResult + "<br>");

  /* 署名Vriが付加されているかの取得 */
  boolean vriFlg = timeStampList[i].getVriFlg();
  resultBuffer.append("vriFlg: " + vriFlg + "<br>");

  /* タイムスタンプ情報の取得 */
  PdfTimeStampToken token = timeStampList[i].getTimeStampToken();

  /*****************************************
   * 取得情報出力：タイムスタンプトークン
   *****************************************/
  /* タイムスタンプデータ値の取得 */
  byte[] tokenData = token.getData();
  resultBuffer.append("Token Data: " + tokenData + "<br>");

  /* 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("Token RegisterDigest: " + 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("Token Digest: " + digest + "<br>");
 }

 /*****************************************
  * 実行結果出力
  *****************************************/
 resultBuffer.append("getInfo : SUCCESS<br>");
} catch (PdfTimeStampException e) {
 resultBuffer.append("getInfo : ERROR<br>");
 resultBuffer.append("status : " + e.getCode() + ", message : " + e.getMessage() + "<br>");
}
finally{

}

 return resultBuffer.toString();
}
%>
</body>
</html>
```

> **コラム**
> タイムスタンプ局のユーザID、パスワード情報は環境に合わせてご指定ください。

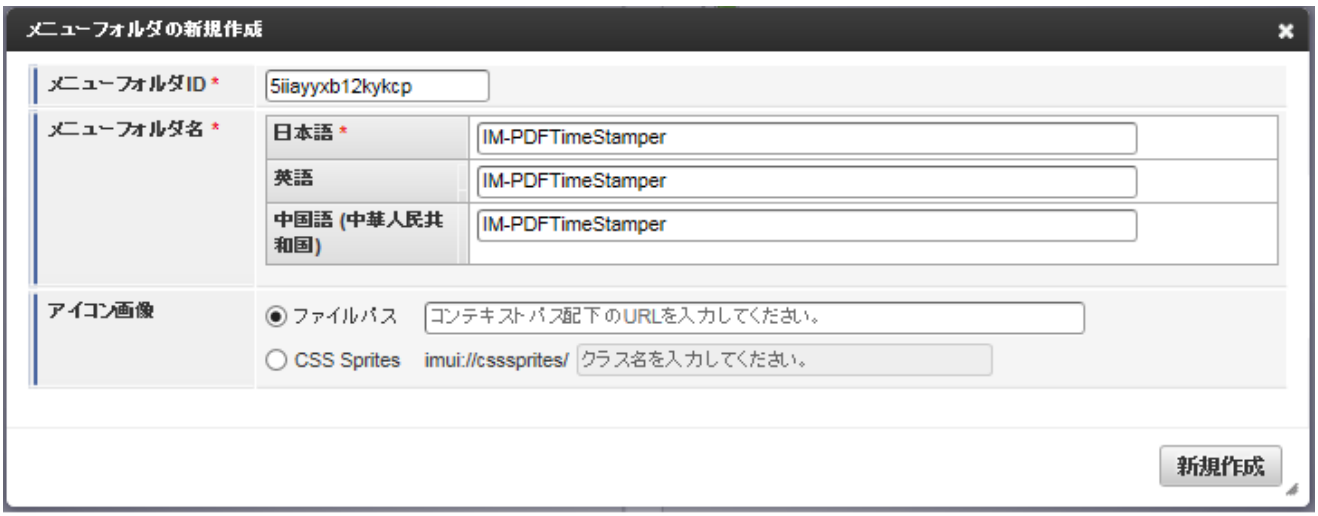
プログラム実行

準備

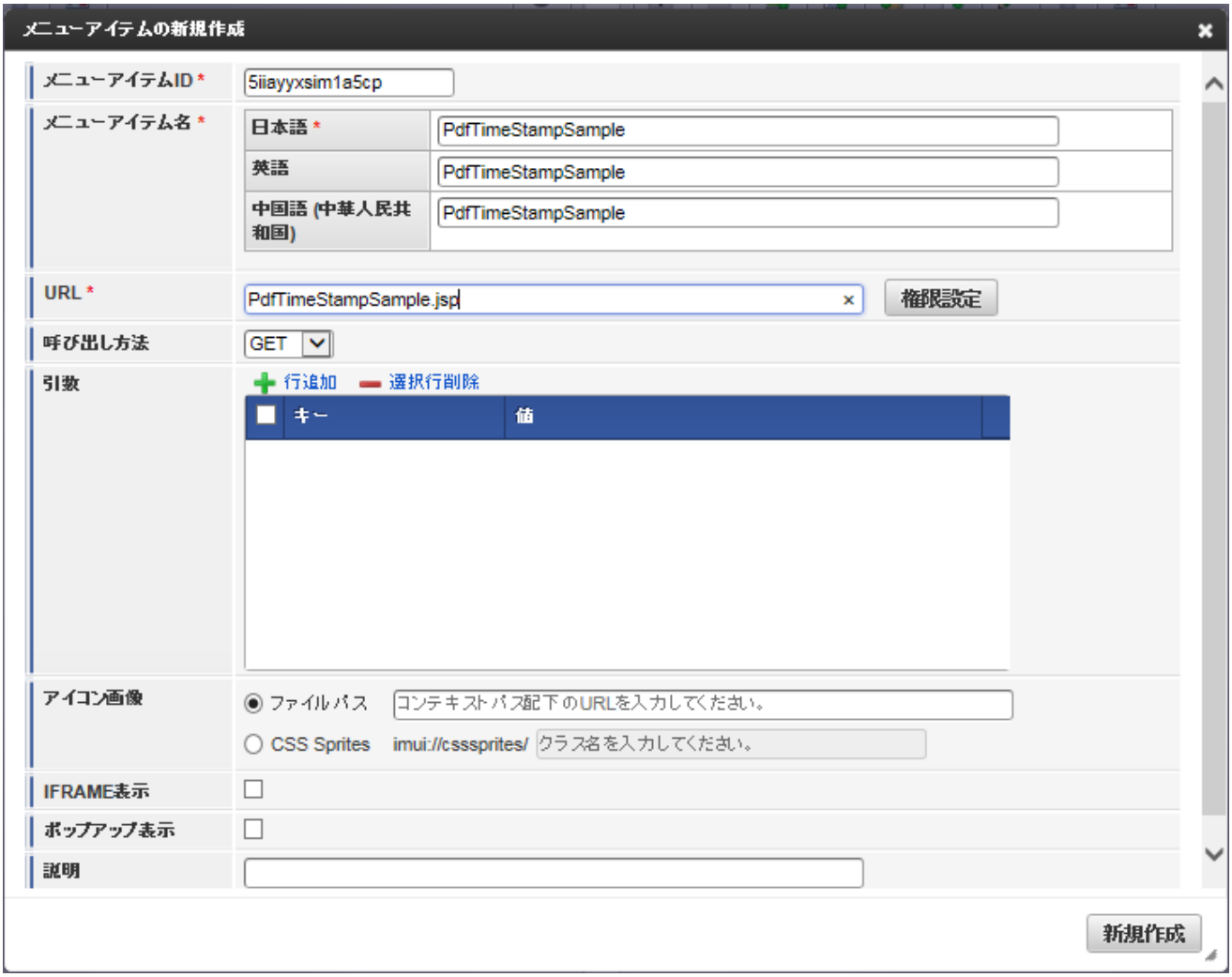
実行させるための準備の手順を説明します。

メニュー設定

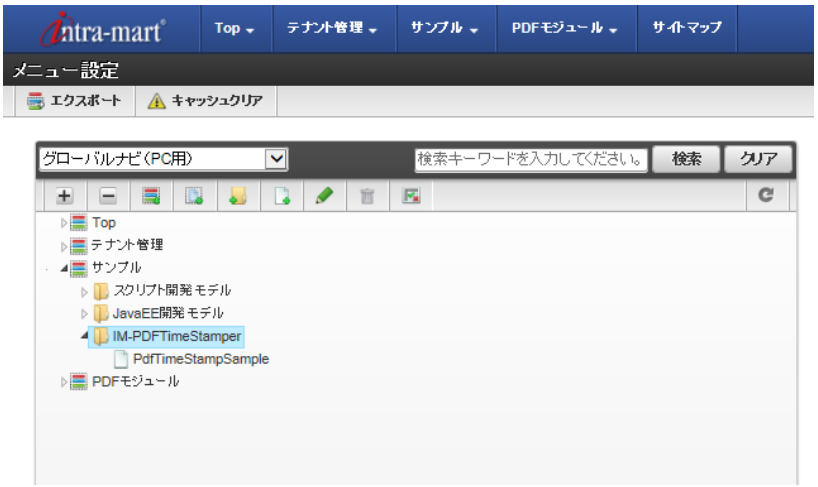
1. テナント管理者でログインし、以下のメニューを設定します。
2. [テナント管理]-[メニュー]画面を開きます。
3. フォルダを作成します。



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。