PLATAFORMA DE DESENVOLVIMENTO PINHÃO PARANÁ

TABELIÃO – MÓDULO PARA CERTIFICAÇÃO DIGITAL

MANUAL DE ACOPLAMENTO E DESENVOLVIMENTO

Setembro – 2008

Sumário de Informações do Documento

Tipo do Documento: Manual

Título do Documento: Manual de Acoplagem da Solução Tabelião.

Estado do Documento: Elaborado

Responsáveis: Emerson Sachio Saito, Thiago Meneghello (Revisado por Cîntia Evangelista)

Palavras-Chaves: Certificação Digital, Tabelião, Acoplamento, Acoplagem, Desenvolvimento.

Resumo: Esse documento contêm as instruções para acoplagem da solução TABELIÃO em sistemas hospedeiros.

Número de páginas: 60

Software utilizados: BrOffice Writer

Versão	Data	Mudanças
1.0	22/12/2007	Elaboração/Criação do Manual
1.1	03/03/2008	Alteração da relação de funcionamento com o Sentinela e informações de Desenvolvimento.
1.2	07/07/2008	Inclusão de facilidades para geração e validação de assinatura no formato EnvelopeED-XML
1.3	01/08/2008	Implementada funcionalidade para efetuar Contra-Assinatura em formato EnvelopED-XML, e os atributos de SignaturePolicyIdentifier.
1.4	20/09/2008	Novas funcionalidades para geração e validação de assinaturas em formato EnvelopING -XML, alteradas as assinaturas dos métodos para gerarEnvelopedXML.
1.5	17/11/2008	Mudanças nas TagLibs de Assinaturas XML em ambiente WEB (usando base64), mudanças nas classes básicas de XML para instanciar a partir de: InputStream, InputSource, Document e String **.**

SUMÁRIO

1 INTRODUÇÃO

Este manual destina-se a fornecer informações aos responsáveis pelo desenvolvimento de sistemas, baseados na Plataforma de Desenvolvimento Pinhão Paraná, de como configurar e utilizar as facilidades providas pela solução denominada TABELIÃO-PINHÃO que responderá pelos requisitos de CERTIFICAÇÃO DIGITAL.

Na versão atual os serviços providos são:

- Autenticação: Login/Validação com certificado;
- Assinatura Digital (PKCS#7/CADES): Geração e Validação;
- Atualização de Lista de Certificados Revogados (LCR), através da aplicação administrativa;
- Manutenção de base de Cadeias de Certificados e repositório de chaves públicas, através da aplicação administrativa;
- Assinatura Digital em formato EnvelopED-XML: Geração e Validação;
- Assinatura Digital em formato EnvelopING-XML: Geração e Validação;

Nos capítulos subseqüentes, serão apresentados os procedimentos de acoplagem do módulo cliente do TABELIÃO ao sistema hospedeiro.

Informações mais detalhadas das bibliotecas do TABELIÃO podem ser obtidas acessando o ***JavaDoc*** que está disponível no site da Plataforma Pinhão.

2 PRÉ-REQUISITOS OBRIGATÓRIOS

Como a solução do TABELIÃO é de uma natureza especial e específica, alguns pré-requisitos são obrigatórios para melhor utilização da solução:

- Para todos os usuários envolvidos na solução técnica é necessária uma capacitação específica, através do curso de Nivelamento Teórico em Certificação Digital que é ministrado internamente pela CELEPAR através da GIC, pois não serão abordados em profundidade nenhuma teoria ou termo relacionado à certificação digital.
- Todo desenvolvedor deverá ter completado o treinamento na plataforma PINHÃO no perfil de Desenvolvedor, pois nenhum detalhe de desenvolvimento será abordado no documento.
- Preferencialmente o uso de um Certificado Digital, válido, e emitido por autoridade certificadora credenciada junto à ICP-BRASIL, caso opte-se por usar certificado de testes, é necessário que este siga as mesmas políticas de requisitos utilizadas pela ICP-BRASIL, lembrando que o TABELIÃO não fornece nenhum recurso para a geração de certificados digitais.
- O TABELIÃO está sendo desenvolvido atualmente em ambiente JAVA6 e apesar de manter compatibilidade com JAVA5, é recomendável o uso da versão atual, tanto nos ambientes servidores, como de usuários. O servidor de aplicações onde o Tabelião é desenvolvido e homologado é o JBOSS, atualmente na versão 4.0.5. Para as funcionalidades de assinatura em formato XML é obrigatório o uso do ambiente JAVA6.
- A aplicação Administrativa do Tabelião, deve estar implantada e em funcionamento. Pois a parte cliente fará acesso às informações mantidas por ela.

3 CONFIGURAÇÃO BÁSICA PARA O SISTEMA HOSPEDEIRO

Considera-se, neste documento, que o sistema hospedeiro será desenvolvido de acordo com a plataforma de desenvolvimento PINHÃO-PARANÁ, ou pelo menos em tecnologia compatível, portanto não serão tratadas neste documento as diferentes opções de uso de softwares ou suas versões.

Entende-se como sistema hospedeiro o projeto de software que irá utilizar as funções disponibilizadas pelo componente TABELÃO-PINHÃO.

A acoplagem do TABELIÃO a um sistema hospedeiro deve, para possibilitar a execução das suas funcionalidades independentes de quais serão, passar por alguns procedimentos básicos e obrigatórios que serão os seguintes:

3.1 Cópia de arquivos clientes

O arquivo denominado “cliente” está disponível em forma de um arquivo compactado em formato padrão JAR, e deve ser copiado para a pasta /context/WEB-INF/lib do projeto/sistema hospedeiro.

Para os projetos que utilizam o plugin MAVEN2 do Eclipse para gerenciar as bibliotecas, basta incluir a referência no arquivo POM.XML, conforme exemplo abaixo:

```
<dependency>
  <groupId>tabeliao</groupId>
  <artifactId>tabeliao_client</artifactId>
  <version>1.4.2.2</version>
</dependency>
```

No site da plataforma PINHÃO-PARANÁ, encontram-se os arquivos (.jar) disponíveis para cópia.

A nomenclatura do arquivo cliente é: tabeliao_client-x.x.x.jar

Além da biblioteca cliente é preciso também, obviamente, o arquivo de DataSource que será utilizado pelo JBOSS, para acesso às informações centralizadas e controladas pela aplicação administrativa, conforme o ambiente:

tabeliao-cli-desenv-ds.xml -> Para ambiente de desenvolvimento.

tabeliao-cli-prod-ds.xml -> Para ambiente de produção.

Os arquivos de configuração podem ser encontrados no sistema Documentador(http://www.documentador.celepar.pr.gov.br/). As informações de cada arquivo dependem do ambiente implantado.

Todas as aplicações WEB na plataforma PINHÃO, dependem das bibliotecas listadas no arquivo POM.XML de acordo com o projeto Mínimo. Além destas bibliotecas padrões, para o uso do TABELIÃO são necessárias as seguintes:

```
<dependency>
  <groupId>bouncycastle</groupId>
  <artifactId>bcmail-jdk16</artifactId>
  <version>140</version>
</dependency>
<dependency>
  <groupId>bouncycastle</groupId>
  <artifactId>bcprov-jdk16</artifactId>
  <version>140</version>
</dependency>
```

Para os projetos que farão AUTENTICAÇÃO e/ou geração de ASSINATURA serão necessários também os arquivos:

- bouncycastleLite.jar
- TabeliaoApplet.jar
- plugin.jar

No site da plataforma PINHÃO-PARANÁ, encontram-se os arquivos (.jar) disponíveis para cópia, estes arquivos deverão estar na pasta /context do projeto.

4 COMUNICAÇÃO ENTRE SISTEMA HOSPEDEIRO E O TABELIÃO

A comunicação entre o sistema hospedeiro e o Tabelião é feita da mesma forma que os demais proto-agentes da plataforma PINHÃO, ou seja, através da sua biblioteca cliente e por intermédio de Tag-Libs.

Além do modo padrão, o Tabelião também disponibiliza uma interface para tratamento da comunicação entre o sistema e certificado digital, que pode estar armazenado em arquivo ou hardware criptográfico. Esta comunicação é feita através da *Applet(http://java.sun.com/applets/)* TabeliaoApplet.jar, que permite a execução de funcionalidades como Autenticação e geração de Assinatura Digital. Esta é uma applet assinada digitalmente que executará no ambiente local do usuário, e é a forma mais segura e simples de conseguir o acesso ao certificado digital.

5 AUTENTICAÇÃO

Uma das funcionalidades providas pelo TABELIÃO-PINHÃO é a AUTENTICAÇÃO de agentes através da leitura e validação do certificado. Isto significa que o usuário do sistema poderá utilizar o certificado digital, seja armazenado em arquivo criptografado ou em hardware(token/smartcard), para efetuar a sua autenticação/login.

Esta funcionalidade é, geralmente, um requisito de alta segurança para garantir a autenticidade da identidade do usuário do sistema. E ainda, em alguns casos, para garantir a proteção e inviolabilidade da senha. É útil também para unificar as chaves de acessos à vários sistemas, e obviamente, desde que todos trabalhem com certificados digitais.

Também é altamente recomendado o uso do proto-agente SENTINELA, uma vez que o mesmo já está preparado para o uso do TABELIÃO como autenticador.

É importante lembrar que o TABELIÃO não faz o trabalho de permissões de acesso, que no modelo da plataforma PINHÃO-PARANÁ é feito pelo SENTINELA, uma vez que o próprio TABELIÃO implementa a classe de autenticação do SENTINELA.

As vantagens do uso do TABELIÃO, é que a solução de autenticação ficará independente de plataforma, não necessita de nenhuma configuração do navegador (browser) do cliente, e toda a validação do certificado, da cadeia e das listas de revogação já estão contempladas.

5.1 Arquivos de bibliotecas necessários

Para a funcionalidade de autenticação são necessários, além da biblioteca cliente, os arquivos:

- bouncycastleLite.jar (versão compatível com o cliente)
- TabeliaoApplet.jar (versão compatível com o cliente)

  Estes arquivos deverão ser copiados para a pasta /context/ do sistema hospedeiro.

5.2 Configurações para Autenticação com Sentinela

Considerando que o sistema utilize a solução do SENTINELA, as seguintes configurações deverão ser incluídas no arquivo /context/WEB-INF/sentinela.xml:

```
      <!-- Autenticação TABELIAO -->
      <Atributo name="tabeliao.client.valida.nivel.acesso"
valor="1"/>
      <Atributo name="tabeliao.client.nivel.acesso"
valor="A3"/>
       <Atributo name="tabeliao.client.dias.mensagem.expirando"
valor="30"/>
       <Atributo name="tabeliao.client.valida.cadeia.certificado"
valor="1"/>
       <Atributo name="tabeliao.client.valida.LCR"
valor="1"/>
       <Atributo name="tabeliao.client.valida.certificado.expirado"
valor="1"/>
      <Atributo name="classeAutenticacao"
valor="gov.pr.celepar.tabeliao.client.autenticacao.AutenticacaoTabeliao"/>
```

Sendo que cada atributo corresponde ao seguinte:

- `tabeliao.client.valida.nivel.acesso`

  Aceita os valores 0 e 1.

  Se for 0, não executará a validação do nível de segurança dos certificados utilizados.

  Se for 1, faz a validação conforme o próximo atributo.

- tabeliao.client.nivel.acesso

  Define o nível dos certificados que serão aceitos pelo sistema:

  Os valores podem ser:

  A1 – Aceita desde os A1 até A4, por definição da ICP-BRASIL

  A2 – Aceita desde os A2 até A4, por definição da ICP-BRASIL

  A3 – Aceita apenas A3 e A4, por definição da ICP-BRASIL

  A4 – Aceita apenas A4, por definição da ICP-BRASIL

  Estes valores só serão válidos se o atributo

`tabeliao.client.valida.nivel.acesso` estiver com o valor 1, caso contrário serão desconsiderados.

- tabeliao.client.dias.mensagem.expirando

  Aceita a atribuição de um valor, entre 30 e 60, que é o número de dias anteriores à expiração da validade dos certificados, que habilitará o TABELIÃO a exibir uma mensagem de aviso, quanto à expiração do certificado.

  Por exemplo: Caso o valor definido seja o número 40, todos os usuários de certificados que estejam para expirar em até 40 dias, receberão um alerta sobre a validade dos mesmos e a necessidade de sua renovação junto à autoridade certificadora.

- `tabeliao.client.valida.cadeia.certificado`

  Aceita os valores 0 e 1.

  Se for 0 (zero), não fará a validação da cadeia de certificação. Este valor só é permitido para ambientes de testes, onde eventualmente poderiam ser utilizados certificados sem valor legal.

  Se for 1 (um), fará a validação.

- `tabeliao.client.valida.LCR`

  Aceita os valores 0 e 1.

  Se for 0 (zero), não fará a validação do certificado junto à lista de certificados revogados emitido pela autoridade certificadora. Este valor só é permitido para ambientes de testes, onde eventualmente poderiam ser utilizados certificados sem valor legal. Também, provisoriamente, para fins de login, se o sistema não prevê o uso da funcionalidade de assinatura digital.

  Se for 1, fará validação.

- `tabeliao.client.valida.certificado.expirado`

  Aceita os valores 0 e 1.

Se for 0, não validará se a data de validade do certificado expirou. Permitido

apenas para ambientes de testes, quando da impossibilidade temporária da renovação do certificado.

Se for 1, fará a validação.

- `classeAutenticacao`

  Nome da classe delegada pelo Sentinela para fazer a autenticação do usuário, no caso do Tabelião deve ser sempre: gov.pr.celepar.tabeliao.client.autenticacao.AutenticacaoTabeliao

Importante: No cadastro do sistema no Sentinela, incluir como exceção os arquivos do item 5.1.

5.3 Chamada à autenticação (TAGLIB)

Após a inclusão dos arquivos e a configuração das propriedades do Sentinela, o sistema hospedeiro poderá implementar a funcionalidade de Autenticação/Login.

A página de autenticação/login deve ser construída utilizando a taglib “login”, definida pelo Tabelião na seguinte URI:"http://celepar.pr.gov.br/taglibs/tabeliao.tld">

Esta taglib irá invocar a Applet: TabeliaoApplet.jar, que permitirá o acesso ao dispositivo criptográfico ou arquivo, e fará a leitura e validação do certificado.

Os parâmetros aceitos nesta TAGLIB são os seguintes:

```
<tabeliao:login
  valorBotao="Entrar"
  caminhoBiblioteca=""
  tipoForm=""
  corFundo="#FFFFFF"
  corDentro="#5AACFE"
  corAba="#B8DBFE"
  corAbaSelecionada="#5AACFE"/>
```

Onde os parâmetros possuem as seguintes funções:

- valorBotão: É o texto que será mostrado no botão de ação da applet.
- caminhoBiblioteca: Define um caminho (diretório) local, definido como “padrão” para a biblioteca (.so ou .dll), que irá acessar o SmartCard/Token utilizando o padrão PKCS#11. Obviamente, só é necessário quando são utilizados estes tipos de dispositivos. A Applet do Tabelião permite que o próprio usuário indique o caminho (path) caso o valor padrão não for o mesmo para o seu equipamento.
- Caso este valor não seja informado o “default” será: /usr/lib/opensc/opensc-pkcs11.so, que é a biblioteca para uso em ambiente LINUX/DEBIAN.
- tipoForm: O tipo do formulário de autenticação, cujo valor indica:
    - Se 1 – Apenas a aba para uso de SmartCard/Token será apresentada,

    - Se 2 – Apenas a aba para uso de Arquivo será apresentada,
    - Se 3 – Ambas as abas para uso de SmartCard/Token ou Arquivos serão apresentadas.

- corFundo: Define a cor de fundo para a applet. Valores em hexadecimal.
- corDentro: Define a cor de dentro para a applet. Valores em hexadecimal.
- corAba: Define a cor da aba da applet quando a mesma não está selecionada. Valores em hexadecimal.
- corAbaSelecionada: Define a cor da aba da applet quando a mesma está selecionada. Valores em hexadecimal.
    - Referência de apoio: http://pt.wikipedia.org/wiki/Tabela_de_cores

5.4 Interface de Autenticação para o SENTINELA

O TABELIÃO implementa a AutenticacaoInterface do SENTINELA, através da classe AutenticacaoTabeliao, para fazer a integração de autenticação/login com o Sentinela. Desta forma, a validação do par Usuário(login) e Senha será feita pelo Tabelião e todas as autorizações de acesso ficam por conta do Sentinela.

Portanto, será então necessário, que o usuário tenha as permissões cadastradas no Sentinela. A chave de relacionamento entre o Tabelião e o Sentinela é o CPF do usuário, então, é importante que esta informação esteja correta.

Pacote: gov.pr.celepar.tabeliao.client.autenticacao

Classe: AutenticacaoTabeliao

Métodos da classe:

- UsuarioAutenticado autentica(String certificado, String CPF, String securityCode)

  Faz a leitura e validação do certificado e devolve as informações(nome,email e CPF) e o controle para o Sentinela.

- void inicializar(SentinelaXml sentinelaXml)

  Inicializa/Atribui as propriedades conforme as configurações do item 5.2

5.5 Classes de autenticação da Applet do Tabelião

Serão listadas as classes de autenticação da Applet(TabeliaoApplet.jar) fornecida pela solução. Além do entendimento mais profundo da solução, estes métodos podem servir como referência para soluções de autenticação/login que não utilizem a solução de segurança fornecida pelo proto-agente SENTINELA. Os parâmetros são os mesmos da taglib de login.

Pacote: gov.pr.celepar.tabeliao.client.applet;

Classes:

- TabeliaoApplet
  - Métodos para tratamento de autenticação
    - void init(); Parâmetro ação="autenticacao" faz chamada a handleAutenticacao.
    - void handleAutenticacao(); Instância ActionAutenticacao.
  - ActionAutenticacao
    - Construtor: ActionAutenticacao(JApplet applet)
    - Método: void execute(String arquivo, String pin, boolean isHardware)

6 ASSINATURA DIGITAL – PKCS#7

O formato implementado desde a versão inicial é a assinatura de arquivos utilizando o padrão PKCS#7, que é um formato que pode ser utilizado para todos os tipos de arquivos e informações, e também é compatível com ferramentas de uso em *desktop* como por exemplo o Assinador do ITI, e o Cryptonit.

Por definição, existem duas formas de geração da assinatura neste padrão, que são:

Assinatura Desanexada: É a forma padrão que o Tabelião usa para gerar as assinaturas, e também a mais recomendada. Nesta forma, o resultado da funcionalidade será um arquivo contendo a assinatura, com a extensão .pkcs7, que corresponderá ao conteúdo que foi assinado. A vantagem desta primeira forma é que o conteúdo (ou arquivo) que foi assinado não será manipulado nem modificado, se mantendo no formato original. A desvantagem está em manter os arquivos (assinatura(s)+conteúdo) armazenados sempre em correspondência. Para minimizar este inconveniente a plataforma PINHÃO-PARANÁ, disponibiliza a solução do proto-agente SCRIBA, que é uma ferramenta de GED, e mantém a correspondência e também já possui integração com o Tabelião.

Assinatura Anexada: Nesta forma o resultado da funcionalidade gera apenas um arquivo contendo a assinatura e o conteúdo(ou arquivo) assinado, sendo que assinatura será um envólucro para o conteúdo. Nesta forma, a vantagem é que o conteúdo não se separa da(s) assinatura(s), o que facilita o seu armazenamento. Já a desvantagem, está no fato de que o conteúdo (ou arquivo) só estará acessível quando o mesmo for desanexado da assinatura, e a tarefa de desanexar só é feita por softwares como os já citados acima (Assinador ITI, Cryptonit), ou ainda através do Tabelião. Considerando que o conteúdo pode ser assinado por mais de um certificado, ou seja ter mais que uma assinatura, o arquivo gerado pode conter várias “camadas” de assinaturas, o que gera uma dificuldade ainda maior para acessar o conteúdo. Este tipo de assinatura só é implementada, pelo Tabelião, para arquivos no

ambiente local do usuário, e que após a assinatura serão carregados(*upload*) para o ambiente servidor.

6.1 Arquivos de bibliotecas necessários

Para a funcionalidade de Assinatura pkcs7 são necessários, além da biblioteca cliente, os arquivos:

- bouncycastleLite.jar
- TabeliaoApplet.jar

Estes arquivos deverão ser copiados para a pasta /context/ do sistema hospedeiro.

6.2 Geração

A geração é o processo de assinatura, que irá gerar o arquivo de assinatura no padrão PCKS#7.

6.2.1 Assinatura de Arquivo em ambiente Servidor.

Para o ambiente de aplicações WEB no padrão da plataforma PINHÃO, o Tabelião disponibiliza a funcionalidade de geração de Assinatura Desanexada. Pois considerando que o processo de assinatura obrigatoriamente acontece no ambiente local do usuário, a transmissão de arquivos entre o servidor e o usuário, através do navegador (*browser*) estará limitado por questões de infra-estrutura. Por este motivo o Tabelião implementa uma forma em que apenas o HASH (SHA-1, de acordo com o padrão da ICP-BRASIL) dos arquivos no servidor são enviados para o usuário.

Para esta funcionalidade há uma *taglib,* chamada: assinaturaWeb, definida pelo Tabelião na seguinte URI:"http://celepar.pr.gov.br/taglibs/tabeliao.tld">

Esta taglib irá invocar a Applet: TabeliaoApplet.jar, que permitirá o acesso ao dispositivo criptográfico ou arquivo, e fará a leitura e validação do certificado.

Os parâmetros aceitos nesta taglig são os seguintes:

```
<tabeliao:assinaturaWeb
  valorBotao=""
  caminhoBiblioteca=""
  tipoForm=""
  action=""
  id=""
  idSeparador=""
  ConteudoHandler=""
  corFundo=""
  corDentro=""
  corAba=""
  corAbaSelecionada=""/>
```

Onde os parâmetros possuem as seguintes funções:

- valorBotão: É o texto que será mostrado no botão de ação da applet.
- caminhoBiblioteca: Define um caminho (diretório) local, definido como "padrão" para a biblioteca (.so ou .dll) que irá acessar o SmartCard/Token utilizando o padrão PKCS#11. Obviamente, só é necessário quando forem utilizados estes tipos de dispositivos. A Applet do Tabelião permite que o próprio usuário indique o caminho (path) caso o valor padrão não for o mesmo para o seu equipamento.

  Caso este valor não seja informado o "default" será: /usr/lib/opensc/opensc-pkcs11.so, que é a biblioteca para uso em ambiente LINUX/DEBIAN.
- tipoForm: O tipo do formulário de autenticação, cujo valor indica:
  - Se 1 – Apenas a aba para uso de SmartCard/Token será apresentada,
  - Se 2 – Apenas a aba para uso de Arquivo será apresentada,
  - Se 3 – Ambas as abas para uso de SmartCard/Token ou Arquivos serão apresentadas.
- action (obrigatório): Nome da ação a executar (assinaturaWeb). Esta *action* receberá um *Form* que deve ser estendido da classe gov.pr.celepar.tabeliao.client.form.TabeliaoAssinaturaWebForm, contendo os métodos String[] getCampoIds() e byte[][] getAssinaturas().

- id (obrigatório): Id do documento. Pode ser utilizado mais que um, nesse caso, eles devem estar separados pelo conjunto de caracteres definidos no idSeparador. Ex: 123;222;321;431;543 (Nesse caso o idSeparador dever igual a ";").
- idSeparador: Caractere ou conjunto de caracteres que será utilizado para separar os ids.
- conteudoHandler(obrigatório): Classe que implementa a interface gov.pr.celepar.tabeliao.client.handler.conteudoHandler. Que neste caso, assinaturaWeb, é a classe gov.pr.celepar.tabeliao.client.handler.ConteudoHandlerHashImp. Pois retornará apenas o hash.
- corFundo: Define a cor de fundo para a applet. Valores em hexadecimal.
- corDentro: Define a cor de dentro para a applet. Valores em hexadecimal.
- corAba: Define a cor da aba da applet quando a mesma não está selecionada. Valores em hexadecimal.
- corAbaSelecionada: Define a cor da aba da applet quando a mesma está selecionada. Valores em hexadecimal.

  Referência de apoio: http://pt.wikipedia.org/wiki/Tabela_de_cores

6.2.2 Assinatura de Arquivo Local ou Anexada

O Tabelião também provê uma outra *taglib* que efetua assinatura digital, totalmente no ambiente local do usuário, e desta forma também permite a geração de assinaturas na forma Anexada.

Nesta funcionalidade os arquivos devem estar armazenados no ambiente local do usuário e a assinatura é gerada também no ambiente local. Conseqüentemente, será necessário que o(s) arquivo(s) seja(m) enviado(s) ao servidor.

Pode ser útil quando a aplicação deve receber os arquivos já assinados, e também a forma mais viável de prover a assinatura Anexada.

Para esta funcionalidade há uma *taglib,* chamada: assinaDocumento, definida pelo Tabelião na seguinte URI:"http://celepar.pr.gov.br/taglibs/tabeliao.tld">

Esta taglib irá invocar a Applet: TabeliaoApplet.jar, que permitirá o acesso ao dispositivo criptográfico ou arquivo, e fará a leitura e validação do certificado.

Os parâmetros aceitos nesta taglig são os seguintes:

```
<tabeliao:assinaDocumento
  valorBotao=""
  caminhoBiblioteca=""
  tipoForm=""
  nomeArquivo=""
  separadorArquivo=""
  anexaArquivo=boolean
  corFundo=""
  corDentro=""
  corAba=""
  corAbaSelecionada=""/>
```

Onde os parâmetros possuem as seguintes funções:

- valorBotão: É o texto que será mostrado no botão de ação da applet.
- caminhoBiblioteca: Define um caminho (diretório) local, definido como “padrão” para a biblioteca (.so ou .dll) que irá acessar o SmartCard/Token utilizando o padrão PKCS#11. Obviamente, só é necessário quando forem utilizados estes tipos de dispositivos. A Applet do Tabelião permite que o próprio usuário indique o caminho (path) caso o valor padrão não for o mesmo para o seu equipamento.

  Caso este valor não seja informado o “default” será: /usr/lib/opensc/opensc-pkcs11.so, que é a biblioteca para uso em ambiente LINUX/DEBIAN.
- tipoForm: O tipo do formulário de autenticação, cujo valor indica:
  - Se 1 – Apenas a aba para uso de SmartCard/Token será apresentada,
  - Se 2 – Apenas a aba para uso de Arquivo será apresentada,
  - Se 3 – Ambas as abas para uso de SmartCard/Token ou Arquivos serão apresentadas.

- nomeArquivo (obrigatório): Nome do(s) arquivo(s) a ser(em) assinado(s), caso seja mais de um, eles devem estar separados pelo conjunto de caracteres definidos no separadorArquivo. Cada nome deve se referir ao caminho exato do arquivo no equipamento do usuário (ex: /home/usuario/nomearquivo)
- separadorArquivo: Caractere ou conjunto de caracteres que será utilizado para separar os nomes dos arquivos.
- anexaArquivo: Valor booleano (true/false) que indica se a assinatura será, ou não, anexada ao arquivo.
- corFundo: Define a cor de fundo para a applet. Valores em hexadecimal.
- corDentro: Define a cor de dentro para a applet. Valores em hexadecimal.
- corAba: Define a cor da aba da applet quando a mesma não está selecionada. Valores em hexadecimal.
- corAbaSelecionada: Define a cor da aba da applet quando a mesma está selecionada. Valores em hexadecimal.

  Referência de apoio: http://pt.wikipedia.org/wiki/Tabela_de_cores

6.3 Validação Assinatura PKCS7

Para as tarefas de validação de assinatura digital o Tabelião provê algumas classes, através da interface cliente, que permite executar as seguintes funcionalidades:

- Validação da Assinatura: Executa as tarefas básicas, verifica se a assinatura é válida e se confere com o conteúdo.
- Validação do Certificado da Assinatura: Faz a verificação completa da cadeia desde o certificado do assinante, contemplando todas as AC's intermediárias, que houverem, até a RAIZ da ICP-BRASIL. Verifica também todas as LCR's (Listas de Certificados Revogados), uma vez que o TABELIÃO mantém uma base

atualizada com um “cache” de todas as listas do certificados cadastrados.

Para apoio às validações da assinatura PKCS7 a interface Cliente, disponibiliza a seguinte classe:

Pacote:

gov.pr.celepar.tabeliao.core.

Classes:

TabeliaoAssinaturaPKCS7;

- Esta é a classe principal e básica para validação da assinatura PKCS7.
- Construtores
  - TabeliaoAssinaturaPKCS7(**byte**[] assinatura)
  - TabeliaoAssinaturaPKCS7(InputStream assinatura)
  - TabeliaoAssinaturaPKCS7(InputStream conteudo, InputStream assinatura)
  - TabeliaoAssinaturaPKCS7(**byte**[] conteudo, InputStream assinatura)
  - TabeliaoAssinaturaPKCS7(InputStream conteudo, **byte**[] assinatura)
  - TabeliaoAssinaturaPKCS7(**byte**[] conteudo, **byte**[] assinatura)
- Métodos:
  - X509Certificate getCertificadoAssinante(); retorna formato básico (java.security.cert.X509Certificate)
  - byte[] getConteudoAssinado();
  - TabeliaoResultadoValidacao valida(); executa validação da assinatura.
  - TabeliaoResultadoValidacao getResultadoValidacao(); retorno: ver pacote gov.pr.celepar.tabeliao.core.validacao
  - CMSSignedData getCMSSignedData(); Retorna conteúdo da assinatura.
  - Date getDataAssinatura();

7 VERIFICAÇÃO/TRATAMENTO DE CERTIFICADOS DIGITAIS

A solução do Tabelião, através da sua interface cliente, disponibiliza as funcionalidades para a verificação e tratamento dos dados contidos nos certificados digitais, emitidos por autoridades certificadoras credenciadas junto à ICP-BRASIL.

Listaremos abaixo as classes para essas verificações:

Pacote:

gov.pr.celepar.tabeliao.core.

Classes:

TabeliaoCertificate;

- Esta classe encapsula um objeto do tipo X509Certificate e gera interfaces para leitura das informações contidas no certificado.
- Construtores
- TabeliaoCertificate(X509Certificate certificate);
    - TabeliaoCertificate(CertificadoAc certificadoAc);
    - TabeliaoCertificate(byte[] data);
    - TabeliaoCertificate(InputStream is);
    - TabeliaoCertificate(CertificadoPublico certificadoPublico);
- Métodos:
    - `X509Certificate getX509Certificate();` retorna no formato básico (java.security.cert.X509Certificate).
    - TabeliaoDN getCertificadoDe(); Retorna o IssuerDn do certificado no formato TabeliaoDN que funciona como um Properties. O método toString dessa classe retorna o IssuerDn.getName().

- String getNome(); Retorna o nome que está definido no CN do CertificadoPara.
- Date getValidadeDe();
- Date getValidadeAte();
- TabeliaoKeyUsage getTabeliaoKeyUsage(); Retorna um Objeto TabeliaoKeyUsage com as informações sobre o uso do certificado.
- TabeliaoSubjectAlternativeNames getTabeliaoSubjectAlternativeNames(); Retorna o SubjectAlternativeNames do certificado no formato TabeliaoSubjectAlternativeNames. Caso não exista essa informação retorna o valor "null".
- String getEmail(); Retorna o email que está definido no SubjectAlternativeNames do certificado. Equivalente a chamada getTabeliaoSubjectAlternativeNames().getEmail(). Caso não exista essa informação retorna o valor "null".
- boolean hasDadosPF(); Verifica se o certificado possui dados de Pessoa Física. Equivalente a chamada getTabeliaoSubjectAlternativeNames().isDadosPF()
- TabeliaoDadosPF getTabeliaoDadosPF(); Retorna os dados de Pessoa Física do certificado no formato TabeliaoDadosPF. Equivalente a chamada getTabeliaoSubjectAlternativeNames().getTabeliaoDadosPF(). Retorna o valor "null" caso o certificado não possua os dados de Pessoa Física.
- boolean hasDadosPJ();Verifica se o certificado possui dados de Pessoa Jurídica. Equivalente a chamada getTabeliaoSubjectAlternativeNames().isDadosPJ()
- TabeliaoDadosPJ getTabeliaoDadosPJ(); Retorna os dados de Pessoa

Jurídica do certificado no formato TabeliaoDadosPJ. Equivalente a chamada getTabeliaoSubjectAlternativeNames().getTabeliaoDadosPJ(). Retorna o valor “null” caso o certificado não possua os dados de Pessoa Jurídica.

- boolean hasDadosEquipamento(); Verifica se o certificado possui dados de Equipamento. Equivalente a chamada getTabeliaoSubjectAlternativeNames().isDadosEquipamento().
- TabeliaoDadosEquipamento geTabeliaoDadosEquipamento(); Retorna os dados de Equipamento do certificado no formato TabeliaoDadosEquipamento. Equivalente a chamada getTabeliaoSubjectAlternativeNames().getTabeliaoDadosEquipamento(). Retorna o valor “null” caso o certificado não possua os dados de Equipamento.
- int getPathLength(); Retorna o valor PathLength do BasicConstraint do certificado. 1 - Caso seja de uma AC. 0 - Caso seja de Usuário Final.
- boolean isCertificadoAc(); Verifica se o certificado é de uma Autoridade Certificadora (AC). “true” - Caso seja de uma AC. “false” - Caso seja de Usuário Final.
- String getTipoCertificado(); Retorna qual o tipo do certificado (A1, A2, A3, A4, S1, S2, S3, S4). Retorna o valor “null” caso não possua a extensão CertificatePolicies.
- String getAuthorityKeyIdentifier(); Retorna o valor da extensão AuthorityKeyIdentifier no formato String. Caso essa informação não esteja no certificado, retorna “null”.
- String getSubjectKeyIdentifier(); Retorna o valor da extensão SubjectKeyIdentifier no formato String. Caso essa informação não esteja no certificado, retorna “null”.
- String getCRLDistributionPoint(); Retorna a URL da Lista de Certificados Revogados (CRL). Mesmo que a CRL possua mais que uma fonte, retorna

apenas a URL. Caso essa informação não esteja no certificado, retorna "null".

- DERObject getExtensionValue(String oid); Retorna um Objeto do tipo DERObject contendo o valor da extensão passado através do OID.
- TabeliaoCertificate getCertificadoGerador(); Retorna o certificado da Autoridade Certificadora.
- List<TabeliaoCertificate> getCadeiaCertificados(); Carrega uma Lista com a cadeia de certificados, onde o primeiro certificado é o certificado Raiz, e o último certificado é o do assinante.
- List<TabeliaoCertificate> getCadeiaCertificadosInversa(); Carrega uma lista com a cadeia de certificados na ordem inversa, ou seja, o primeiro certificado do assinante e por último o certificado Raiz.
- TabeliaoLCR getTabeliaoLCR(); Recupera a LCR a partir do endereço no certificado e adiciona à base do Tabelião, onde será mantido o seu "cache".
- TabeliaoResultadoValidacao valida(); Executa a validação completa do certificado.
- TabeliaoResultadoValidacao getResultadoValidacao(); Faz uma chamada ao método valida();
- void validaValidadeCertificado(TabeliaoResultadoValidacao trv); Faz uma chamada recursiva ao método, assumindo o "default" de 30 dias para vencimento.
- String toString(); Retorna em formato texto o resultado da validação.

CertificadoExtra;

- Esta classe possui as informações extras definidas pela ICP-BRASIL para os certificados de Pessoa Física, Pessoa Jurídica e Equipamentos. Estes campos estão definidos no DOC-ICP-04 v2.0 de 18/04/2006.
- Construtores
    - CertificadoExtra(X509Certificate certificate)
- Métodos:
    - boolean isCertificadoPF(); Pessoa Física.
    - boolean isCertificadoPJ(); Pessoa Jurídica.
    - boolean isCertificadoEquipamento(); Equipamento.
    - OID_2_16_76_1_3_1 getOID_2_16_76_1_3_1(); retorna objeto da classe definida no pacote: `gov.pr.celepar.tabeliao.core.oid`
    - OID_2_16_76_1_3_5 getOID_2_16_76_1_3_5(); retorna objeto da classe definida no pacote: `gov.pr.celepar.tabeliao.core.oid`
    - OID_2_16_76_1_3_6 getOID_2_16_76_1_3_6(); retorna objeto da classe definida no pacote: `gov.pr.celepar.tabeliao.core.oid`
    - OID_2_16_76_1_3_2 getOID_2_16_76_1_3_2(); retorna objeto da classe definida no pacote: `gov.pr.celepar.tabeliao.core.oid`
    - OID_2_16_76_1_3_3 getOID_2_16_76_1_3_3(); retorna objeto da classe definida no pacote: `gov.pr.celepar.tabeliao.core.oid`
    - OID_2_16_76_1_3_4 getOID_2_16_76_1_3_4(); retorna objeto da classe definida no pacote: `gov.pr.celepar.tabeliao.core.oid`
    - OID_2_16_76_1_3_7 getOID_2_16_76_1_3_7(); retorna objeto da classe definida no pacote: `gov.pr.celepar.tabeliao.core.oid`
    - OID_2_16_76_1_3_8 getOID_2_16_76_1_3_8(); retorna objeto da classe definida no pacote: `gov.pr.celepar.tabeliao.core.oid`
    - String getEmail(); Endereço de e-mail armazenado no certificado.

TabeliaoDadosEquipamento;

- Esta classe possui as informações definidas pela ICP-BRASIL para os certificados de Equipamentos. Estes campos estão definidos no DOC-ICP-04 v2.0 de 18/04/2006. Ver também *TabeliaoSubjectAlternativeNames.*

- Construtores:
    - TabeliaoDadosEquipamento(OID_2_16_76_1_3_2 oid1, OID_2_16_76_1_3_3 oid2, OID_2_16_76_1_3_4 oid3, OID_2_16_76_1_3_8 oid4);
- Métodos:
    - String getNomeResponsavel(); OID = 2.16.76.1.3.2 e conteúdo = nome do responsável pelo certificado;
    - String getNomeEmpresarial(); OID = 2.16.76.1.3.8 e conteúdo = nome empresarial constante do CNPJ (Cadastro Nacional de Pessoa Jurídica), sem abreviações, se o certificado for de pessoa jurídica;
    - String getCNPJ(); OID = 2.16.76.1.3.3 e conteúdo = Cadastro Nacional de Pessoa Jurídica (CNPJ) da pessoa jurídica titular do certificado;
    - Date getDataNascimento(); OID = 2.16.76.1.3.4 e conteúdo = das 8 (oito) posições, com a data de nascimento do responsável pelo certificado.
    - String getNis(); OID = 2.16.76.1.3.4 e conteúdo = nas 11 (onze) posições com o número de Identificação Social – NIS (PIS, PASEP ou CI).
    - String getRg(); OID = 2.16.76.1.3.4 e conteúdo = nas 15 (quinze) posições com o número do RG do responsável;.
    - getOrgaoUfExpedidor(); OID = 2.16.76.1.3.4 e conteúdo = nas 6 (seis) posições contendo as siglas do órgão expedidor do RG e respectiva UF;

TabeliaoDadosPF;

- Esta classe possui as informações definidas pela ICP-BRASIL para os certificado de Pessoa Física. Estes campos estão definidos no DOC-ICP-04 v2.0 de 18/04/2006. Ver também *TabeliaoSubjectAlternativeNames.*
- Construtores:
    - TabeliaoDadosPF(OID_2_16_76_1_3_1 oid1, OID_2_16_76_1_3_5 oid2, OID_2_16_76_1_3_6 oid3)
- Métodos:
    - String getCPF(); OID = 2.16.76.1.3.1 e conteúdo = das 11 (onze) posições com o Cadastro de Pessoa Física (CPF) do titular.
    - Date getDataNascimento(); OID = 2.16.76.1.3.1 e conteúdo = das 8 (oito) posições com a data de nascimento do titular.

- String getNis(); OID = 2.16.76.1.3.1 e conteúdo = das 11 (onze) posições com o número de Identificação Social NIS (PIS, PASEP ou CI).
- String getRg(); OID = 2.16.76.1.3.1 e conteúdo = das 15 (quinze) posições com o número do Registro Geral RG do titular.
- String getOrgaoUfExpedidorRg(); OID = 2.16.76.1.3.1 e conteúdo = das 6 (seis) posições com as siglas do órgão expedidor do RG e respectiva UF.
- String getTituloEleitor(); OID = 2.16.76.1.3.5 e conteúdo = das 12 (doze) posições, o número de inscrição do Título de Eleitor.
- String getSecaoTituloEleitor(); OID = 2.16.76.1.3.5 e conteúdo = das 4 (quatro) posições contendo a Seção.
- String getZonaTituloEleitor(); OID = 2.16.76.1.3.5 e conteúdo = das 3 (três) posições contendo a Zona Eleitoral.
- String getMunicipioUfTituloEleitor(); OID = 2.16.76.1.3.5 e conteúdo = das 22 (vinte e duas) posições com o município e a UF do Título de Eleitor.
- String getInss(); OID = 2.16.76.1.3.6 e conteúdo = nas 12 (doze) posições o número do Cadastro Específico do INSS (CEI) da pessoa física titular do certificado.

TabeliaoDadosPJ;

- Esta classe possui as informações extras definidas pela ICP-BRASIL para os certificados de Pessoa Jurídica. Estes campos estão definidos no DOC-ICP-04 v2.0 de 18/04/2006. Ver também *TabeliaoSubjectAlternativeNames.*
- Construtores:
  - TabeliaoDadosPJ(OID_2_16_76_1_3_2 oid1, OID_2_16_76_1_3_3 oid2, OID_2_16_76_1_3_4 oid3, OID_2_16_76_1_3_7 oid4)
- Métodos:
  - String getNomeResponsavel(); OID = 2.16.76.1.3.2 e conteúdo = nome do responsável pelo certificado;
  - String getCNPJ(); OID = 2.16.76.1.3.3 e conteúdo = Cadastro Nacional de Pessoa Jurídica (CNPJ) da pessoa jurídica titular do certificado;
  - Date getDataNascimento(); OID = 2.16.76.1.3.4 e conteúdo = das 8

(oito) posições, com a data de nascimento do responsável pelo certificado;

- String getNis(); OID = 2.16.76.1.3.4 e conteúdo = nas 11 (onze) posições com o número de Identificação Social – NIS (PIS, PASEP ou CI);
- String getRg(); OID = 2.16.76.1.3.4 e conteúdo = nas 15 (quinze) posições com o número do RG do responsável;
- getOrgaoUfExpedidor(); OID = 2.16.76.1.3.4 e conteúdo = nas 6 (seis) posições contendo as siglas do órgão expedidor do RG e respectiva UF;
- String getINSS(); OID = 2.16.76.1.3.7 e conteúdo = nas 12 (doze) posições o número do Cadastro Específico do INSS (CEI) da pessoa jurídica titular do certificado;

TabeliaoDN;

- Classe para retorno de DN (Distinguished Name): Conjunto de dados que identifica de modo inequívoco uma entidade ou indivíduo pertencente ao mundo físico no mundo digital (por exemplo: país=BR, estado=Parana, nome organizacional=Sua Empresa S.A., nome comum=José da Silva). (Fonte: ICP-BRASIL).

  Estende java.util.Properties.

- Construtores:
  - TabeliaoDN(String dn)
- Métodos:
  - String toString();

TabeliaoKeyUsage;

- Classe para identificar os atributos de usos do certificado.
- Construtores:
  - TabeliaoKeyUsage(X509Certificate cert)
- Métodos:

- boolean isDigitalSignature(); Se True -> Permite assinatura digital.
- boolean isNonRepudiation(); Se True -> Estabelece condição de não repudio.
- boolean isKeyEncipherment(); Se True -> Permite cifragem de chave criptográfica.
- boolean isDataEncipherment(); Se True -> Permite cifrar dados diretamente, exceto chaves criptográficas.
- boolean isKeyAgreement(); Se True -> Permite a obtenção de chaves de sessão. (ex. https).
- boolean isKeyCertSign(); Se True -> Permite que seja usada para assinatura de certificados. (somente AC).
- boolean isCRLSign(); Se True -> Permite que seja usada para assinatura de LCR (Lista de Certificados Revogados). (somente AC).
- boolean isEncipherOnly(); Se True -> usado somente em conjunto com a opção keyAgreement e indica que a chave de sessão obtida somente pode ser usada para cifrar dados.
- boolean isDecipherOnly(); semelhante ao encipherOnly, porém a chave de sessão obtida somente pode executar a operação de decifrar.
- String toString(); Retorna todos em formato texto.

TabeliaoLCR;

- Classe para tratar Lista de Certificados Revogados, trabalha com o formato X509CRL (java.security.cert.x509CRL).
- Construtores:
  - TabeliaoLCR(InputStream is)
  - TabeliaoLCR(byte[] data)
- Métodos:
  - X509CRL getCRL(); Retorna a LCR do certificado.

TabeliaoSubjectAlternativeNames;

- Classe que permite instanciar um certificado de modo mais genérico, sem pré-

identificação do tipo (PF,PJ ou Equipamento) através de seus OID's. A classe retorna o objeto conforme o tipo encontrado.

- Construtores:
    - `TabeliaoSubjectAlternativeNames(X509Certificate certificate)`
- Métodos:
    - boolean isDadosPF();
    - TabeliaoDadosPF getTabeliaoDadosPF();
    - boolean isDadosPJ()
    - TabeliaoDadosPJ getTabeliaoDadosPJ();
    - boolean isDadosEquipamento();
    - TabeliaoDadosEquipamento getTabeliaoDadosEquipamento();
    - String getEmail();

Pacote:

gov.pr.celepar.tabeliao.core.oid

Classes:

OID_2_16_76_1_3_1;

- Classe básica para tratamento de atributos de alguns atributos de Pessoa Física, de acordo com os padrões definidos no DOC-ICP-04 v2.0 de 18/04/2006, pela ICP-BRASIL.

  Estende OIDGenerico.

- Construtores
  - OID_2_16_76_1_3_1()
- Métodos:
  - void inicializa();
  - String getDataNascimento();
  - String getCPF();
  - String getNIS();
  - String getRg();
  - String getOrgaoUfExpedidor();

OID_2_16_76_1_3_2;

- Classe básica para tratamento de atributos de alguns atributos de Pessoa Jurídica ou Equipamento, de acordo com os padrões definidos no DOC-ICP-04 v2.0 de 18/04/2006, pela ICP-BRASIL.

  Estende OIDGenerico.

- Construtores
  - OID_2_16_76_1_3_2()
- Métodos:
  - void inicializa();
  - String getNome();

OID_2_16_76_1_3_3;

- Classe básica para tratamento de atributos de alguns atributos de Pessoa Física, de acordo com os padrões definidos no DOC-ICP-04 v2.0 de 18/04/2006, pela ICP-BRASIL.

  Estende OIDGenerico.

- Construtores
  - OID_2_16_76_1_3_3()
- Métodos:
  - void inicializa();
  - String getCNPJ();

OID_2_16_76_1_3_4;

- Classe básica para tratamento de atributos de alguns atributos de Pessoa Física, de acordo com os padrões definidos no DOC-ICP-04 v2.0 de 18/04/2006, pela ICP-BRASIL.

  Estende OIDGenerico.

- Construtores
  - OID_2_16_76_1_3_4()
- Métodos:
  - void inicializa();
  - String getDataNascimento();
  - String getCPF();
  - String getNIS();
  - String getRg();
  - String getOrgaoUfExpedidor();

OID_2_16_76_1_3_5;

- Classe básica para tratamento de atributos de alguns atributos de Pessoa Física, de acordo com os padrões definidos no DOC-ICP-04 v2.0 de 18/04/2006, pela ICP-BRASIL.

  Estende OIDGenerico.

- Construtores
  - OID_2_16_76_1_3_5()
- Métodos:
  - void inicializa();
  - String getTitulo();
  - String getZona();
  - String getSecao();
  - String getMunicipioUf();

OID_2_16_76_1_3_6;

- Classe básica para tratamento de atributos de alguns atributos de Pessoa Física, de acordo com os padrões definidos no DOC-ICP-04 v2.0 de 18/04/2006, pela ICP-BRASIL.

  Estende OIDGenerico.

- Construtores
  - OID_2_16_76_1_3_6()
- Métodos:
  - void inicializa();
  - String getInss();

OID_2_16_76_1_3_7;

- Classe básica para tratamento de atributos de alguns atributos de Pessoa Jurídica, de acordo com os padrões definidos no DOC-ICP-04 v2.0 de 18/04/2006, pela ICP-BRASIL.

  Estende OIDGenerico.

- Construtores
    - OID_2_16_76_1_3_7()
- Métodos:
    - void inicializa();
    - String getInss();

OID_2_16_76_1_3_8;

- Classe básica para tratamento de atributos de alguns atributos de Equipamento, de acordo com os padrões definidos no DOC-ICP-04 v2.0 de 18/04/2006, pela ICP-BRASIL.

  Estende OIDGenerico.

- Construtores
    - OID_2_16_76_1_3_8()
- Métodos:
    - void inicializa();
    - String getNome();

OIDGenerico;

- Classe Genérica para tratamento de atributos de alguns atributos de Pessoa Física, Pessoa Jurídica e Equipamento de acordo com os padrões definidos no DOC-ICP-04 v2.0 de 18/04/2006, pela ICP-BRASIL.
- Construtores
    - protected OIDGenerico()
- Métodos:
    - OIDGenerico getInstance(byte[] data)
    - OIDGenerico getInstance(DerValue der)
    - void inicializa()
    - void inicializa(Object[] campos)
    - String getOid();

- String getData();

Pacote:

gov.pr.celepar.tabeliao.core.validacao

Classes:

TabeliaoResultadoValidacao;

- Classe para tratamento dos resultados da validação de um certificado digital.
- Construtores
  - TabeliaoResultadoValidacao()
- Métodos:
  - void addOk(String validacao);
  - void addAviso(String validacao, String descricao);
  - void addErro(String validacao, String descricao);
  - boolean hasAviso();
  - boolean hasErro();
  - String getMensagem(String validacao);
  - String getDescricao(String validacao);
  - List<String> getValidacoesOk();
  - List<String> getValidacoesAviso();
  - List<String> getValidacoesErro();
  - String toString();

8 TAGLIBS AUXILIARES

Além das TAGLIBs já apresentadas, o Tabelião também fornece outras que complementam a geração e validação de assinaturas, e validação de certificados.

Todas elas estão definidas pelo Tabelião na seguinte URI:"http://celepar.pr.gov.br/taglibs/tabeliao.tld">

São elas:

- mensagemAviso: Mostra uma mensagem de aviso do Tabelião, caso exista.
- mensagemErro: Mostra uma mensagem de erro do Tabelião, caso exista.
- infoValidacao: Mostra informações do resultado de validação e possui os seguintes atributos:
    - resultadoValidacao: Objeto do tipo TabeliaoResultadoValidacao.
    - margem: Margem a ser incluída antes de cada informação.
    - corOK: Cor para as mensagens de OK.
    - corAviso: Cor para as mensagens de Aviso.
    - corErro: Cor para as mensagens de Erro.

9 ASSINATURA EM FORMATO ENVELOPED-XML

A partir da versão 1.2.0 da biblioteca cliente do Tabelião, estão disponibilizadas as facilidades para geração e validação de assinaturas digitais no formato EnvelopED-XML.

As especificações do formato e as políticas de uso seguem os padrões recomendados pelos documentos: DOC-ICP-15.05 (Política Padrão XADES), DOC-ICP-15.0 (Perfil XADES ICP-Brasil) e DOC-ICP-15 (Assinaturas Digitais na ICP-Brasil) da ICP-BRASIL.

A implementação da versão 1.4.2.2, contempla apenas os requisitos obrigatórios, nas versões subseqüentes está planejada a utilização dos outros atributos. Os outros formatos que forem reconhecidos, também serão contemplados nas novas versões do TABELIÃO.

O formato EnvelopED-XML, por definição, só pode ser utilizado para gerar assinaturas para arquivos em formato XML, o exemplo mais conhecido deste tipo de assinatura é o projeto NF-e (Nota Fiscal Eletrônica).

9.1 Arquivos de bibliotecas necessários

Se o projeto irá utilizar a solução completa do TABELIÃO, para geração das assinaturas, serão necessários os seguintes arquivos:

- bouncycastleLite.jar (versão compatível com o cliente)
- TabeliaoApplet.jar (versão compatível com o cliente)

  Estes arquivos deverão ser copiados para a pasta /context/ do sistema hospedeiro.

Se estas bibliotecas não forem utilizadas, o processo de acesso ao certificado e a chave privada deverão ser implementados pelo projeto. E também não será possível utilizar as TAG-LIBs fornecidas.

9.2 Geração da assinatura

A geração é o processo de assinatura, que irá criar um novo arquivo XML contendo a assinatura no padrão EnvelopED-XML.

Diferentemente do processo para assinatura PKCS7, o processo de assinatura EnvelopED-XML não trabalhará apenas com HASH (Resumo do Arquivo), mas com o arquivo completo. Por isto é importante considerar a quantidade e tamanho dos arquivos que serão assinados de cada vez. Por se tratar de arquivos em formato XML, obviamente o seu tamanho é muito inferior a muitos outros tipos.

9.2.1 Arquivos em Ambiente Servidor

Definiu-se a assinatura em ambiente servidor, como o processo no qual os arquivos em formato XML estão armazenados no servidor de aplicações ou local centralizados, ao qual somente o servidor de aplicações tem acesso, e o processo de assinatura é executado de tal forma que os arquivos devam ser selecionados pelo própria aplicação e depois assinados pelo usuário. Os arquivos assinados são, portanto, outros arquivos e se necessário devem substituir ou não os arquivos originais, e esta tarefa também é de responsabilidade da aplicação.

Para esta funcionalidade foi criada uma *TAGLIB,* chamada: assinarEnvelopedXmlWeb, definida pelo Tabelião na seguinte URI: "http://celepar.pr.gov.br/taglibs/tabeliao.tld"

Esta taglib irá invocar a Applet: TabeliaoApplet.jar, que permitirá o acesso ao dispositivo criptográfico ou arquivo, e fará a leitura e validação do certificado.

Os parâmetros aceitos nesta taglib são os seguintes:

```
<tabeliao:assinarEnvelopedXmlWeb
  valorBotao=""
  caminhoBiblioteca=""
  tipoForm=""
  action=""
  conteudosArquivos=""
  separadorArquivo=""
  nomeTagAssinar=""
  politicaId=""
  politicaUri=""
  contraAssinatura=""
  corFundo=""
  corDentro=""
  corAba=""
  corAbaSelecionada=""/>
```

Onde os parâmetros possuem as seguintes funções:

- valorBotão: É o texto que será mostrado no botão de ação da applet.
- caminhoBiblioteca: Define um caminho (diretório) local, definido como "padrão" para a biblioteca (.so ou .dll) que irá acessar o SmartCard/Token utilizando o padrão PKCS#11. Obviamente, só é necessário quando forem utilizados estes tipos de dispositivos. A Applet do Tabelião permite que o próprio usuário indique o caminho (path) caso o valor padrão não for o mesmo para o seu equipamento.

  Caso este valor não seja informado o "default" será: /usr/lib/opensc/opensc-pkcs11.so,

  que é a biblioteca para uso em ambiente LINUX/DEBIAN.

- tipoForm: O tipo do formulário de autenticação, cujo valor indica:
  - Se 1 – Apenas a aba para uso de SmartCard/Token será apresentada,
  - Se 2 – Apenas a aba para uso de Arquivo será apresentada,
  - Se 3 – Ambas as abas para uso de SmartCard/Token ou Arquivos serão apresentadas.
- action (obrigatório): Nome da ação a ser processada na execução da assinatura. Esta *action* receberá um *Form* que deve ser estendido da classe gov.pr.celepar.tabeliao.client.form.TabeliaoAssinaturaXmlWebForm, que contém os métodos getArquivosXmlAssinados(), getCampoArquivosXmlAssinados(), getConteudoCampoArquivosXmlAssinados() e setCampoArquivosXmlAssinados(). Nesta ação poderão ser definidos os métodos para tratar os arquivos gerados.
- conteudosArquivos (obrigatório): Conteúdo(s) do(s) documento(s) a ser(em) assinado(s), em Base64. Pode ser utilizado mais que um, nesse caso, eles devem estar separados pelo conjunto de caracteres definidos no separadorArquivo. Ex: AA;BB;ZZ

(Nesse caso o separadorArquivo será igual a ";"). Por definição do próprio projeto, neste versão, optou-se por codificar os dados (XML) que serão transmitidos do servidor para o cliente (Applet) em formato base64, e a biblioteca do TABELIÃO fornece a classe utilitária (`gov.pr.celepar.tabeliao.util.Base64Utils`) com os métodos `String base64Encode(byte[] aData)` e `byte[] base64Decode(String aData)`, que devem ser usados para tratar o envio e recebimento dos arquivos.

- separadorArquivo: Caractere ou conjunto de caracteres que será utilizado para separar os arquivos a serem assinados.
- nomeTagAssinar: nome da TAG no arquivo XML que será assinada. Para assinatura da TAG, é preciso que a mesma tenha um ID/id/iD. Se não for informada uma TAG o arquivo é assinado por completo, o que permite apenas uma ÚNICA assinatura deste tipo.
- politicaId: Identificador da política de assinatura que está sendo utilizada.
- politicaUri: Endereço WEB/Internet onde pode ser encontrado o arquivo digital com a política de assinatura que está sendo utilizada.
- contraAssinatura: Indica se será efetuada uma Contra-Assinatura, para estes casos é preciso que o arquivo já contenha uma ou mais assinaturas, para serem contra-assinadas. Aceita os valores "*true*" ou "*on*".
- corFundo: Define a cor de fundo para a applet. Valores em hexadecimal.
- corDentro: Define a cor de dentro para a applet. Valores em hexadecimal.
- corAba: Define a cor da aba da applet quando a mesma não está selecionada. Valores em hexadecimal.
- corAbaSelecionada: Define a cor da aba da applet quando a mesma está selecionada. Valores em hexadecimal.

  Referência de apoio: http://pt.wikipedia.org/wiki/Tabela_de_cores

9.2.2 Arquivos em Ambiente Local

Grande parte deste processo é idêntico ao do item 9.2.

A diferença principal deste procedimento é que os arquivos XML, deverão estar armazenados no ambiente local do equipamento do usuário ou em uma área na qual ele tenha acesso. Esta funcionalidade é útil quando o usuário precisa assinar um arquivo que está somente em seu ambiente. Mas ao contrário do ambiente servidor, onde a aplicação pode executar todos os procedimentos de controle do arquivo assinado, neste caso o arquivo será gerado também no ambiente local e, portanto, depois de assinado deverá ser enviado ao ambiente servidor, caso necessário.

Para esta funcionalidade foi criada também uma *TAGLIB,* chamada: assinarEnvelopedXml, definida pelo Tabelião na seguinte URI: "http://celepar.pr.gov.br/taglibs/tabeliao.tld"

Esta taglib irá invocar a Applet: TabeliaoApplet.jar, que permitirá o acesso ao dispositivo criptográfico ou arquivo, e fará a leitura e validação do certificado.

Os parâmetros aceitos nesta taglib são os seguintes:

```
<tabeliao:assinarEnvelopedXml
  valorBotao=""
  caminhoBiblioteca=""
  tipoForm=""
  nomeArquivo=""
  separadorArquivo=""
  nomeTagAssinar=""
  politicaId=""
  politicaUri=""
  contraAssinatura=""
  corFundo=""
  corDentro=""
  corAba=""
  corAbaSelecionada=""/>
```

Onde os parâmetros possuem as seguintes funções:

- valorBotão: É o texto que será mostrado no botão de ação da applet.
- caminhoBiblioteca: Define um caminho (diretório) local, definido como “padrão” para

a biblioteca (.so ou .dll) que irá acessar o SmartCard/Token utilizando o padrão PKCS#11. Obviamente, só é necessário quando forem utilizados estes tipos de dispositivos. A Applet do Tabelião permite que o próprio usuário indique o caminho (path) caso o valor padrão não for o mesmo para o seu equipamento.

Caso este valor não seja informado o "default" será: /usr/lib/opensc/opensc-pkcs11.so,

que é a biblioteca para uso em ambiente LINUX/DEBIAN.

- tipoForm: O tipo do formulário de autenticação, cujo valor indica:
  - Se 1 – Apenas a aba para uso de SmartCard/Token será apresentada,
  - Se 2 – Apenas a aba para uso de Arquivo será apresentada,
  - Se 3 – Ambas as abas para uso de SmartCard/Token ou Arquivos serão apresentadas.
- nomeArquivo (obrigatório): Nome do documento a ser assinado. Pode ser utilizado mais que um, nesse caso, eles devem estar separados pelo conjunto de caracteres definidos no separadorArquivo. Ex: A.xml;B.xml;Z.xml (Nesse caso o separadorArquivo será igual a ";"), este nome deve conter o caminho exato da localização do arquivo no ambiente do usuário (ex: /home/usuario/nomeArquivo).
- separadorArquivo: Caractere ou conjunto de caracteres que será utilizado para separar os arquivos a serem assinados.
- nomeTagAssinar: nome da TAG no arquivo XML que será assinada. Para assinatura da TAG, é preciso que a mesma tenha um ID/id/iD. Se não for informada uma TAG o arquivo é assinado por completo, o que permite apenas uma ÚNICA assinatura deste tipo.
- politicaId: Identificador da política de assinatura que está sendo utilizada.
- politicaUri: Endereço WEB/Internet onde pode ser encontrado o arquivo digital com a política de assinatura que está sendo utilizada.
- contraAssinatura: Indica se será efetuada uma Contra-Assinatura, para estes casos é preciso que o arquivo já contenha uma ou mais assinaturas, para serem contra-assinadas. Aceita os valores "*true*" ou "*on*".
- corFundo: Define a cor de fundo para a applet. Valores em hexadecimal.
- corDentro: Define a cor de dentro para a applet. Valores em hexadecimal.
- corAba: Define a cor da aba da applet quando a mesma não está selecionada. Valores em hexadecimal.
- corAbaSelecionada: Define a cor da aba da applet quando a mesma está selecionada.

Valores em hexadecimal.

Referência de apoio: http://pt.wikipedia.org/wiki/Tabela_de_cores

9.2.3 Geração da assinatura – Sem TAGLIB

Para os casos em que a assinatura do arquivo XML deva ser feita de forma diferente das que foram apresentadas, como por exemplo: pela própria aplicação (utilizando um certificado próprio), a biblioteca cliente do TABELIÃO fornece uma classe para geração da assinatura. Nestes casos o acesso à chave privada e o certificado deve ser implantado pela aplicação.

Pacote:

gov.pr.celepar.tabeliao.core.

Classe:

GerarEnvelopedXML;

- Classe para geração do arquivo XML Assinado no formato Enveloped-XML, criando as propriedades XADES reconhecidas pela ICP-BRASIL de acordo com os documentos DOC-ICP-15.05 e DOC-ICP-15.02. Na versão 1.2.0 estão somente as obrigatórias.
- Métodos:
  - Document assinarArquivoEnvelopedXml(String aFileName, String tagToSign, String politicaId, String politicaUri, boolean contraAssinatura, PrivateKeyAndCertChain privateKeyAndCertChain);
  - Document assinarArquivoEnvelopedXml(InputStream aFile, String tagToSign, String politicaId, String politicaUri, boolean contraAssinatura,PrivateKeyAndCertChain privateKeyAndCertChain)
  - Document assinarArquivoEnvelopedXml(Document arquivoDocument, String tagToSign, String politicaId, String politicaUri, boolean contraAssinatura, PrivateKeyAndCertChain privateKeyAndCertChain)
  - Document assinarArquivoEnvelopedXml(byte[] conteudo, String

tagToSign, String politicaId, String politicaUri, boolean contraAssinatura, PrivateKeyAndCertChain privateKeyAndCertChain)

- Document assinarArquivoEnvelopedXml(String conteudoString, String tagToSign, String politicaId, String politicaUri, boolean contraAssinatura,PrivateKeyAndCertChain privateKeyAndCertChain)
- Document assinarArquivoEnvelopedXml(InputSource conteudoInputSource, String tagToSign, String politicaId, String politicaUri, boolean contraAssinatura,PrivateKeyAndCertChain privateKeyAndCertChain)

9.3 Validação de Assinatura EnvelopED-XML

Para o formato EnvelopED-XML, nas tarefas de validação da assinatura digital o Tabelião provê algumas classes, através da interface cliente, que permite executar as seguintes funcionalidades:

- Validação da Assinatura: Executa as tarefas básicas, verifica se a assinatura é válida e se confere com o conteúdo. Também é feito parse do arquivo XML para garantir que o mesmo seja válido.
- Validação do Certificado da Assinatura: Idêntico ao do PCKS7, pois os formatos de Certificados são sempre os mesmos (X509). Faz a verificação completa da cadeia desde o certificado do assinante, contemplando todas as AC's intermediárias, que houverem, até a RAIZ da ICP-BRASIL. Verifica também todas as LCR's (Listas de Certificados Revogados), uma vez que o TABELIÃO mantém uma base atualizada com um "cache" de todas as listas dos certificados cadastrados.

Para apoio às validações da assinatura EnvelopED-XML a interface Cliente, disponibiliza a seguinte classe:

Pacote:

gov.pr.celepar.tabeliao.core.

Classes:

TabeliaoAssinaturaEnvelopedXML;

- Esta é a classe principal e básica para tratamento e validação da assinatura em formato Enveloped-XML, ela estende a classe básica TabeliaoAssinaturaXML.
- Construtores
  - TabeliaoAssinaturaEnvelopedXML(byte[] arquivoAs);
  - TabeliaoAssinaturaEnvelopedXML(InputStream arquivoAs);

- TabeliaoAssinaturaEnvelopedXML(Document arquivoAs);
- TabeliaoAssinaturaEnvelopedXML(String arquivoAs);
- TabeliaoAssinaturaEnvelopedXML(InputSource arquivoAs);

- Métodos:
  - valida(); Executa a validação de todas as assinaturas contidas no arquivo XML, faz com que as listas: assinaturas, resultadosValidacao e certificadosDasAssinaturas, sejam populadas. Permitindo assim, que os outros métodos "*get*" possam retornar resultados.
  - TabeliaoResultadoValidacao getResultadoValidacao(int ix); Retorna o resultado da validação para o índice ix referente a assinatura correspondente. Se não houver validação para o índice retornará nulo.
  - List<TabeliaoResultadoValidacao> getResultadosValidacoes(); Retorna todos os resultados das validações. Se não houver validações retornará nulo.
  - String getUriTagAssinada(int ix); Retorna valor URI/ID da TAG assinada, na assinatura de índice ix, se vazio ou "null" significa que o arquivo todo foi assinado.
  - String getNomeTagAssinada(int ix);Retorna o nome da TAG que foi assinada, na assinatura de índice ix, se vazio ou null significa que o arquivo todo foi assinado.
  - int getQuantidadeAssinaturas(); Retorna a quantidade de assinaturas no arquivo XML, se igual a 0(zero) significa que não há assinatura.
  - XMLSignature getAssinatura(int ix); Retorna a assinatura XML do arquivo conforme o índice (ix).
  - List<XMLSignature> getAssinaturasAssinantes(); Retorna a lista da assinaturas contidas no arquivo XML.
  - TabeliaoCertificate getCertificadoAssinante(int ix); Retorna o certificado do assinante conforme o índice (ix).
  - List<TabeliaoCertificate> getCertificadosAssinantes (); Retorna a lista de Certificados contidos no arquivos XML
  - Date getDataAssinatura(int ix); Retorna a data de Assinatura contida em SigningTime.
  - NodeList getSigningCertificate(int ix); Retorna o XML que contém

SigningCertificate, que é uma das propriedades assinadas obrigatórias da ICP-BRASIL. Contém cert = conforme DOC-ICP-04; CertDigest: /DigestMethod e /DigestValue; IssuerSerial: /X509IssuerName e /X509SerialNumber

- String getSignaturePolicyIdentifier (int ix); Retorna SignaturePolicyIdentifier, que identifica qual é a política de assinatura. É uma das propriedades assinadas obrigatórias da ICP-BRASIL.
- String getDataObjectFormat(int ix); Retorna DataObjectFormat do objeto Assinado . É uma das propriedades assinadas obrigatórias da ICP-BRASIL.
- String getSigPolicyId(int ix); Retorna SigPolicyId, que é uma das propriedades de SignaturePolicyIdentifier, que por sua vez é uma das propriedades assinadas obrigatórias da ICP-BRASIL.
- String getSPURI(int ix); Retorna SPURI, que é uma das propriedades de SignaturePolicyIdentifier, que por sua vez é uma das propriedades assinadas obrigatórias da ICP-BRASIL.
- boolean isContraAssinatura(int ix); retorna “true”, se a assinatura indicada pelo índice ix é uma contra-assinatura.
- NodeList getContraAssinatura(int ix); Retorna um NodeList da propriedade CounterSignature, se houver. Esta lista contém o elemento Signature que foi contra assinado, este por sua vez também pode possuir um elemento CounterSignature recursivamente igual.
- boolean validarVigencias(): Executa a validação da(s) Data(s) de Validade(s), para o(s) certificado(s) da(s) assinatura(s) no arquivo. Tem como base para validade o atributo ValidadeAte, e toma como base a data atual do sistema. É preciso que o método valida() tenha sido executado.
- boolean validarCadeias(): Executa a validação da(s) cadeia(s) de certificação, para o(s) certificado(s) da(s) assinatura(s) no arquivo Eh preciso que o método valida() tenha sido executado. Este método fará acesso a base de dados de cadeias do Tabelião.
- boolean validarLCR():Executa a validação da LRC (Lista de Certificados Revogados), para o(s) certificado(s) da(s) assinatura(s) no arquivo. É preciso que o método valida() tenha sido executado. Este método fara acesso a base de dados LCR do Tabelião.

10 ASSINATURA EM FORMATO ENVELOPING-XML

A partir da versão 1.4.0 da biblioteca cliente do Tabelião, estão disponibilizadas as facilidades para geração e validação de assinaturas digitais no formato EnvelopING-XML.

As especificações do formato e as políticas de uso seguem os padrões recomendados pelos documentos: DOC-ICP-15.05 (Política Padrão XADES), DOC-ICP-15.0 (Perfil XADES ICP-Brasil) e DOC-ICP-15 (Assinaturas Digitais na ICP-Brasil) da ICP-BRASIL.

A implementação da versão 1.4.2.2 contempla apenas os requisitos obrigatórios, nas versões subseqüentes está planejada a utilização dos outros atributos. Os outros formatos que forem reconhecidos também serão contemplados nas novas versões do TABELIÃO.

O formato EnvelopING-XML, por definição, só pode ser utilizado para gerar assinaturas para arquivos em formato XML, este formato é muito comum para troca de mensagens por Web-Services.

10.1 Arquivos de bibliotecas necessários

Se o projeto irá utilizar a solução completa do TABELIÃO, para geração das assinaturas, serão necessários os seguintes arquivos:

- bouncycastleLite.jar (versão compatível com o cliente)
- TabeliaoApplet.jar (versão compatível com o cliente)

  Estes arquivos deverão ser copiados para a pasta /context/ do sistema hospedeiro.

Se estas bibliotecas não forem utilizadas, o processo de acesso ao certificado e a chave privada deverão ser implementados pelo projeto. E também não será possível utilizar as *TAG-LIB*s fornecidas pela solução do Tabelião.

10.2 Geração da assinatura

A geração é o processo de assinatura, que irá criar um novo arquivo XML contendo a assinatura no padrão EnvelopING-XML.

Diferentemente do processo para assinatura PKCS7, o processo de assinatura EnvelopING-XML, mesmo no processo WEB, não trabalhará apenas com HASH (Resumo do Arquivo) mas com o arquivo completo. Por isto é importante considerar a quantidade e tamanho dos arquivos que serão assinados de cada vez. Por se tratar de arquivos em formato XML, obviamente o seu tamanho é muito inferior a muitos outros tipos.

10.2.1 Arquivos em Ambiente Servidor

Definiu-se a assinatura em ambiente servidor, como o processo no qual os arquivos em formato XML estão armazenados no servidor de aplicações ou local centralizados, ao qual somente o servidor de aplicações tem acesso, e o processo de assinatura é executado de tal forma que os arquivos devam ser selecionados pela própria aplicação e depois assinados pelo usuário. Os arquivos assinados são, portanto, outros arquivos e se necessário devem substituir os arquivos originais e esta tarefa também é de responsabilidade da aplicação.

Para esta funcionalidade foi criada uma *TAGLIB,* chamada: assinarEnvelopingXmlWeb, definida pelo Cliente do Tabelião na seguinte URI: "http://celepar.pr.gov.br/taglibs/tabeliao.tld"

Esta *taglib* irá invocar a Applet: TabeliaoApplet.jar, que permitirá o acesso ao dispositivo criptográfico ou arquivo, e fará a leitura e validação do certificado.

Os parâmetros aceitos nesta *taglib* são os seguintes:

```
<tabeliao:assinarEnvelopingXmlWeb
  valorBotao=""
  caminhoBiblioteca=""
  tipoForm=""
  action=""
  conteudosArquivos=""
  separadorArquivo=""
  politicaId=""
  politicaUri=""
  corFundo=""
  corDentro=""
  corAba=""
  corAbaSelecionada=""/>
```

Onde os parâmetros possuem as seguintes funções:

- valorBotão: É o texto que será mostrado no botão de ação da applet.
- caminhoBiblioteca: Define um caminho (diretório) local, definido como "padrão" para a biblioteca (.so ou .dll) que irá acessar o SmartCard/Token utilizando o padrão PKCS#11. Obviamente, só é necessário quando forem utilizados estes tipos de dispositivos. A Applet do Tabelião permite que o próprio usuário indique o caminho (path) caso o valor padrão não for o mesmo para o seu equipamento.

  Caso este valor não seja informado o "default" será: /usr/lib/opensc/opensc-pkcs11.so, que é a biblioteca para uso em ambiente LINUX/DEBIAN.
- tipoForm: O tipo do formulário de autenticação, cujo valor indica:
  - Se 1 – Apenas a aba para uso de SmartCard/Token será apresentada,
  - Se 2 – Apenas a aba para uso de Arquivo será apresentada,
  - Se 3 – Ambas as abas para uso de SmartCard/Token ou Arquivos serão apresentadas.
- action (obrigatório): Nome da ação a ser processada na execução da assinatura. Esta *action* receberá um *Form* que deve ser estendido da classe gov.pr.celepar.tabeliao.client.form.TabeliaoAssinaturaXmlWebForm, que contém os métodos getArquivosXmlAssinados(), getCampoArquivosXmlAssinados(),

getConteudoCampoArquivosXmlAssinados() e setCampoArquivosXmlAssinados(). Nesta ação poderão ser definidos os métodos para tratar os arquivos gerados.

- conteudosArquivos (obrigatório): Conteúdo(s) do(s) documento(s) a ser(em) assinado(s). Como pode ser utilizado mais que um, nesse caso, eles devem estar separados pelo conjunto de caracteres definidos no separadorArquivo. Ex: AA;BB;ZZ (Nesse caso o separadorArquivo será igual a ";"). Por definição do próprio projeto, nesta versão, optou-se por codificar os dados (XML) que serão transmitidos do servidor para o cliente (Applet) em formato base64, e a biblioteca do TABELIÃO fornece a classe utilitária (`gov.pr.celepar.tabeliao.util.Base64Utils`) com os métodos `String base64Encode(byte[] aData)` e `byte[] base64Decode(String aData)`, que devem ser usados para tratar o envio e recebimento dos arquivos.
- separadorArquivo: Caractere ou conjunto de caracteres que será utilizado para separar os arquivos a serem assinados.
- politicaId: Identificador da política de assinatura que está sendo utilizada.
- politicaUri: Endereço WEB/Internet onde pode ser encontrado o arquivo digital com a política de assinatura que está sendo utilizada.
- corFundo: Define a cor de fundo para a applet. Valores em hexadecimal.
- corDentro: Define a cor de dentro para a applet. Valores em hexadecimal.
- corAba: Define a cor da aba da applet quando a mesma não está selecionada. Valores em hexadecimal.
- corAbaSelecionada: Define a cor da aba da applet quando a mesma está selecionada. Valores em hexadecimal.

  Referência de apoio: http://pt.wikipedia.org/wiki/Tabela_de_cores

10.2.2 Arquivos em Ambiente Local

Grande parte deste processo é idêntico ao do item 10.2.1 .

A diferença principal deste procedimento é que os arquivos XML, deverão estar armazenados no ambiente local do equipamento do usuário ou em uma área na qual ele tenha acesso. Esta funcionalidade é útil quando o usuário precisa assinar um arquivo que está somente em seu ambiente. Mas ao contrário do ambiente servidor, onde a aplicação pode executar todos os procedimentos de controle do arquivo assinado, neste caso o arquivo será gerado também no ambiente local e, portanto, depois de assinado deverá ser enviado ao ambiente servidor, caso necessário. E todas as validações ficam a cargo do usuário.

Para esta funcionalidade foi criada também uma *TAGLIB,* chamada: assinarEnvelopingXml, definida pelo Cliente do Tabelião na seguinte URI: "http://celepar.pr.gov.br/taglibs/tabeliao.tld"

Esta taglib irá invocar a Applet: TabeliaoApplet.jar, que permitirá o acesso ao dispositivo criptográfico ou arquivo, e fará a leitura e validação do certificado.

Os parâmetros aceitos nesta taglib são os seguintes:

```
<tabeliao:assinarEnvelopingXml
  valorBotao=""
  caminhoBiblioteca=""
  tipoForm=""
  nomeArquivo=""
  separadorArquivo=""
  politicaId=""
  politicaUri=""
  corFundo=""
  corDentro=""
  corAba=""
  corAbaSelecionada=""/>
```

Onde os parâmetros possuem as seguintes funções:

- valorBotão: É o texto que será mostrado no botão de ação da applet.
- caminhoBiblioteca: Define um caminho (diretório) local, definido como “padrão” para a biblioteca (.so ou .dll) que irá acessar o SmartCard/Token utilizando o padrão PKCS#11. Obviamente, só é necessário quando forem utilizados estes tipos de

dispositivos. A Applet do Tabelião permite que o próprio usuário indique o caminho (path) caso o valor padrão não for o mesmo para o seu equipamento.

Caso este valor não seja informado o “default” será: /usr/lib/opensc/opensc-pkcs11.so,

que é a biblioteca para uso em ambiente LINUX/DEBIAN.

- tipoForm: O tipo do formulário de autenticação, cujo valor indica:
  - Se 1 – Apenas a aba para uso de SmartCard/Token será apresentada,
  - Se 2 – Apenas a aba para uso de Arquivo será apresentada,
  - Se 3 – Ambas as abas para uso de SmartCard/Token ou Arquivos serão apresentadas.
- nomeArquivo (obrigatório): Nome do documento a ser assinado. Pode ser utilizado mais que um, nesse caso, eles devem estar separados pelo conjunto de caracteres definidos no separadorArquivo. Ex: A.xml;B.xml;Z.xml (Nesse caso o separadorArquivo será igual a ";"). Cada nome deve representar exatamente a localização do arquivo no ambiente do usuário (ex: /home/usuario/nomeArquivo).
- separadorArquivo: Caractere ou conjunto de caracteres que será utilizado para separar os arquivos a serem assinados.
- politicaId: Identificador da política de assinatura que está sendo utilizada.
- politicaUri: Endereço WEB/Internet onde pode ser encontrado o arquivo digital com a política de assinatura que está sendo utilizada.
- corFundo: Define a cor de fundo para a applet. Valores em hexadecimal.
- corDentro: Define a cor de dentro para a applet. Valores em hexadecimal.
- corAba: Define a cor da aba da applet quando a mesma não está selecionada. Valores em hexadecimal.
- corAbaSelecionada: Define a cor da aba da applet quando a mesma está selecionada. Valores em hexadecimal.

  Referência de apoio: http://pt.wikipedia.org/wiki/Tabela_de_cores

10.2.3 Geração da assinatura – Sem TAGLIB

Para os casos em que a assinatura do arquivo XML deva ser feita de forma diferente das que foram apresentadas, como por exemplo: pela própria aplicação (utilizando um

certificado próprio), a biblioteca cliente do TABELIÃO fornece uma classe para geração da assinatura. Nestes casos o acesso à chave privada e o certificado deve ser implantado pela própria aplicação.

Pacote:

gov.pr.celepar.tabeliao.core.

Classe:

GerarEnvelopingXML;

- Classe para geração do arquivo XML Assinado no formato Enveloping-XML, criando as propriedades XADES reconhecidas pela ICP-BRASIL de acordo com os documentos DOC-ICP-15.05 e DOC-ICP-15.02. Na versão 1.4.0 estão somente as obrigatórias.
- Métodos:
  - Document assinarArquivoEnvelopingXml(String aFileName, String, String politicaId, String politicaUri, PrivateKeyAndCertChain privateKeyAndCertChain);
  - Document assinarArquivoEnvelopingXml(InputStream aFile, String politicaId, String politicaUri, PrivateKeyAndCertChain privateKeyAndCertChain)
  - Document assinarArquivoEnvelopingXml(Document arquivoDocument, String politicaId, String politicaUri, PrivateKeyAndCertChain privateKeyAndCertChain)
  - Document assinarArquivoEnvelopingXml(byte[] conteudo, String politicaId, String politicaUri, PrivateKeyAndCertChain privateKeyAndCertChain)
  - Document assinarArquivoEnvelopingXml(String conteudoString, String politicaId, String politicaUri, PrivateKeyAndCertChain privateKeyAndCertChain)
  - Document assinarArquivoEnvelopingXml(InputSource conteudoInputSource, String politicaId, String politicaUri, PrivateKeyAndCertChain privateKeyAndCertChain)

10.3 Validação de Assinatura EnvelopING-XML

Para o formato EnvelopING-XML, nas tarefas de validação da assinatura digital o Tabelião provê algumas classes, através da interface cliente, que permitem executar as seguintes funcionalidades:

- Validação da Assinatura: Executa as tarefas básicas, verifica se a assinatura é válida e se confere com o conteúdo. Também é feito parse do arquivo XML para garantir que o mesmo seja válido.
- Validação do Certificado da Assinatura: Idêntico ao do PCKS7, pois os formatos de Certificados são sempre os mesmos (X509). Faz a verificação completa da cadeia desde o certificado do assinante, contemplando todas as AC's intermediárias, que houverem, até a RAIZ da ICP-BRASIL. Verifica também todas as LCR's (Listas de Certificados Revogados), uma vez que o TABELIÃO mantém uma base atualizada com um "cache" de todas as listas dos certificados cadastrados.

Para apoio às validações da assinatura EnvelopING-XML a interface Cliente, disponibiliza a seguinte classe:

Pacote:

gov.pr.celepar.tabeliao.core.

Classes:

`TabeliaoAssinaturaEnvelopingXML`;

- Esta é a classe principal e básica para tratamento e validação da assinatura em formato Enveloped-XML, ela estende a classe básica TabeliaoAssinaturaXML.
- Construtores
  - TabeliaoAssinaturaEnvelopingXML(byte[] arquivoAs);
  - TabeliaoAssinaturaEnvelopingXML(InputStream arquivoAs);

- TabeliaoAssinaturaEnvelopingXML(Document arquivoAs);
- TabeliaoAssinaturaEnvelopingXML(String arquivoAs);
- TabeliaoAssinaturaEnvelopingXML(InputSource arquivoAs)

- Métodos:

  - valida(); Executa a validação de todas as assinaturas contidas no arquivo XML, faz com que as listas: assinaturas, resultadosValidacao e certificadosDasAssinaturas, sejam populadas. Permitindo assim, que os outros métodos "*get*" possam retornar resultados.
  - TabeliaoResultadoValidacao getResultadoValidacao(int ix); Retorna o resultado da validação para o índice ix referente a assinatura correspondente. Se não houver validação para o índice retornará nulo.
  - List<TabeliaoResultadoValidacao> getResultadosValidacoes(); Retorna todos os resultados das validações. Se não houver validações retornará nulo.
  - String getUriTagAssinada(int ix); Retorna valor URI/ID da TAG assinada, na assinatura de índice ix, se vazio ou "null" significa que o arquivo todo foi assinado.
  - int getQuantidadeAssinaturas(); Retorna a quantidade de assinaturas no arquivo XML, se igual a 0(zero) significa que não há assinatura.
  - XMLSignature getAssinatura(int ix); Retorna a assinatura XML do arquivo conforme o índice (ix).
  - List<XMLSignature> getAssinaturasAssinantes(); Retorna a lista da assinaturas contidas no arquivo XML.
  - TabeliaoCertificate getCertificadoAssinante(int ix); Retorna o certificado do assinante conforme o índice (ix).
  - List<TabeliaoCertificate> getCertificadosAssinantes (); Retorna a lista de Certificados contidos no arquivos XML
  - Date getDataAssinatura(int ix); Retorna a data de Assinatura contida em SigningTime.
  - NodeList getSigningCertificate(int ix); Retorna o XML que contém SigningCertificate, que é uma das propriedades assinadas obrigatórias da ICP-BRASIL. Contém cert = conforme DOC-ICP-04; CertDigest: /DigestMethod e /DigestValue; IssuerSerial: /X509IssuerName e /X509SerialNumber

- String getSignaturePolicyIdentifier (int ix); Retorna SignaturePolicyIdentifier, que identifica qual é a política de assinatura. É uma das propriedades assinadas obrigatórias da ICP-BRASIL.
- String getDataObjectFormat(int ix); Retorna DataObjectFormat do objeto Assinado . É uma das propriedades assinadas obrigatórias da ICP-BRASIL.
- String getSigPolicyId(int ix); Retorna SigPolicyId, que é uma das propriedades de SignaturePolicyIdentifier, que por sua vez é uma das propriedades assinadas obrigatórias da ICP-BRASIL.
- String getSPURI(int ix); Retorna SPURI, que é uma das propriedades de SignaturePolicyIdentifier, que por sua vez é uma das propriedades assinadas obrigatórias da ICP-BRASIL.
- boolean validarVigencias(): Executa a validação da(s) Data(s) de Validade(s), para o(s) certificado(s) da(s) assinatura(s) no arquivo. Tem como base para validade o atributo ValidadeAte, e toma como base a data atual do sistema. É preciso que o método valida() tenha sido executado.
- boolean validarCadeias(): Executa a validação da(s) cadeia(s) de certificação, para o(s) certificado(s) da(s) assinatura(s) no arquivo é preciso que o método valida() tenha sido executado. Este método fará acesso a base de dados de cadeias do Tabelião.
- boolean validarLCR():Executa a validação da LRC (Lista de Certificados Revogados), para o(s) certificado(s) da(s) assinatura(s) no arquivo. É preciso que o método valida() tenha sido executado. Este método fará acesso a base de dados LCR do Tabelião.