Certificados Digitais

Os certificados digitais utilizados na implementação dos Web Services para a integração com o sistema Pamcard, seguem o padrão ICP-Brasil (http://www.iti.gov.br/icp-brasil) e seu gerenciamento é de inteira responsabilidade do contratante.

Todas as conexões utilizadas para suportar as transações realizadas na integração deverão ser seguras através da utilização de protocolo seguro SSL (Secure Socket Layer).

É de responsabilidade única e exclusiva de cada uma das partes garantir a não replicação nem distribuição do certificado em questão e/ou qualquer informação a respeito do mesmo.

- ## Exemplos

Java:

A implementação de cliente de um *Web Service* em Java pode ser feita por meio de diversos *frameworks* como Apache CXF, Axis, JBossWS e etc.

Para este guia foi utilizada a implementação nativa do **JBossWS**, versão compatível com o servidor de aplicação JBoss 5.1. É possível que funcione com outras implementações, entretanto é necessário realizar testes para verificar a compatibilidade.

O primeiro passo é gerar, por meio da ferramenta WSConsume o código cliente para interagir com o *Web Service* em questão a partir do WSDL fornecido. É recomendado que esse WSDL esteja armazenado localmente ao invés de realizar chamadas remotas para o contrato. Por essa razão, faça o download do WSDL antes de utilizar o WSConsume.

Não é o escopo desse guia detalhar o funcionamento da ferramenta WSConsume. Para entender como utilizar essa ferramenta, acesse a documentação oficial.

Uma vez os *stubs* gerados pelo WSConsume, importe-os para o seu projeto para que possa fazer referência ao *Web Service*. A classe que será utilizada como cliente do *Web Service* será similiar à listagem abaixo.

Repare que todas as informações sobre o *Web Service* é definida em "*hard code*" pela ferramenta WSConsume. É uma boa prática retirar esse comportamento da classe e importar essas informações de outra fonte de dados (como um arquivo de propriedades, por exemplo).

```
@WebServiceClient(name = "WSTransacional", targetNamespace =
"http://webservice.pamcard.jee.pamcary.com.br", wsdlLocation =
"file:/c:/TEMP/WSTransacional.wsdl")
public class WSTransacional_Service
          extends Service {

      private final static URL WSTRANSACIONAL_WSDL_LOCATION;
      private final static Logger logger =
Logger.getLogger(br.com.pamcary.ws.WSTransacional_Service.class.getName());

      static {
          URL url = null;
          try {
              URL baseUrl;
              baseUrl =
br.com.pamcary.ws.WSTransacional_Service.class.getResource(".");
              url = new URL(baseUrl, "file:/c:/TEMP/WSTransacional.wsdl");
          } catch (MalformedURLException e) {
              logger.warning("Failed to create URL for the wsdl Location:
'file:/c:/TEMP/WSTransacional.wsdl', retrying as a local file");
              logger.warning(e.getMessage());
          }
          WSTRANSACIONAL_WSDL_LOCATION = url;
      }

      public WSTransacional_Service(URL wsdlLocation, QName serviceName) {
          super(wsdlLocation, serviceName);
      }

      public WSTransacional_Service() {
          super(WSTRANSACIONAL_WSDL_LOCATION, new
QName("http://webservice.pamcard.jee.pamcary.com.br", "WSTransacional"));
      }

      /**
       *
       * @return
       *     returns WSTransacional
       */
      @WebEndpoint(name = "WSTransacional")
      public WSTransacional getWSTransacional() {
          return super.getPort(new
QName("http://webservice.pamcard.jee.pamcary.com.br", "WSTransacional"),
WSTransacional.class);
      }
```

A seguir, é necessário informar às bibliotecas de conexão do Java que a transmissão do*request* será feita de forma segura por meio de SSL e o cliente deverá ser autenticado por meio de certificado digital.

Diferente da configuração do Certificado Digital utilizando .NET, em Java o repositório não deverá ficar sob gerenciamento do Sistema Operacional, e sim armazenado em um local pré determinado no sistema de arquivos do SO.

O certificado digital será acessado por meio da bilbioteca do *framework* JBossWS, a qual ficará responsável por adicionar o certificado na requisição HTTPS ao *Web Service*. Para tal, é necessário adicionar ao seu *classpath* as seguintes bibliotecas (exemplo em Maven):

```
<dependency>
    <groupId>jboss-repo.org.jboss.ws.native</groupId>
    <artifactId>jbossws-native-core</artifactId>
    <version>3.1.2.GA</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.jboss</groupId>
    <artifactId>jbossxb</artifactId>
    <version>2.0.1.GA</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>javax.ejb</groupId>
    <artifactId>ejb-api</artifactId>
    <version>3.0</version>
    <scope>test</scope>
</dependency>
```

Uma vez especificado um diretório no sistema (por exemplo`/var/java/certificados/meu_certificado.pfx`), é necessário configurar a biblioteca do JBossWS para criar os objetos de transmissão SSL. A listagem abaixo ilustra como criar esses objetos e consumir o mesmo *Web Service* gerado por meio do *WSConsume*.

```
@Test
public void testWSTransacional_ServiceURLQName() throws IOException,
GeneralSecurityException {
    System.setProperty("javax.net.debug", "all"); //isso vai te ajudar a identificar
problemas na conexão segura.

    logger.debug("Declarando o endpoint e o parâmetro de entrada.");
    WSTransacional_Service service = new WSTransacional_Service(new
URL("file:/C:/TEMP/WSTransacional.wsdl"), new
QName("http://webservice.pamcard.jee.pamcary.com.br", "WSTransacional"));
    WSTransacional serviceCall = service.getWSTransacional();

    logger.debug("Adicionando o certificado no SSL");

((BindingProvider)serviceCall).getRequestContext().put(StubExt.PROPERTY_KEY_STORE,
"file:/var/java/meu_keystore.pfx");

((BindingProvider)serviceCall).getRequestContext().put(StubExt.PROPERTY_KEY_STORE_PA
SSWORD, "minha_senha_keystore");

((BindingProvider)serviceCall).getRequestContext().put(StubExt.PROPERTY_KEY_STORE_TY
PE, "PKCS12");

    logger.debug("Declarando os valores de entrada que serão compostos no Request");

    RequestTO request = new RequestTO();

    logger.debug("Definindo o contexto e os 'fieldTO'");
    FieldTO field = new FieldTO();

    field.setKey("");
    field.setValue("");
    request.setContext("");

    request.getFields().add(field);

    try{
    logger.debug("Executando o request");
    ResponseTO response = serviceCall.execute(request);

    logger.debug("Tratando da resposta.");

    if(response.getFields() != null) {
        for(FieldTO item : response.getFields()) {
            logger.info(item.getKey() + ": " + item.getValue());
        }
    } else {
        logger.debug("Requisição OK, mas resposta vazia. Algo pode estar errado.");
    }
    } catch (Exception ex) {
        logger.error("Erro durante a requisição: ", ex);
        fail("Erro na requisicao");
    }
}
```

Para que a aplicação execute corretamente se estiver utilizando Java 1.6 ou superior, é necessário adicionar como argumento de inicialização da JVM o caminho das bibliotecas do JBossWS do diretório endorsed do servidor JBossWS: -Djava.endorsed.dirs:/var/java/jboss-5.1.0.GA/lib/endorsed/

Neste diretório devem estar contidas as seguintes bibilotecas:

- jbossws-native-jaxrpc.jar
- jbossws-native-jaxws.jar
- jbossws-native-jaxws-ext.jar
- jbossws-native-saaj.jar

Após essas configurações, o *Web Service* estará pronto para uso, bastando adicionar as funcionalidades de negócio.

É importante frisar que para cada *framework* escolhido para realizar uma conexão SSL, é preciso verificar na documentação correspondente a forma de adicionar o objetoSSLSocketFactory na requisição HTTPS.

Há também a possibilidade de adicionar os dados do certificado digital diretamente na JVM, independente do *framework* utilizado. Dessa forma, ficará a cargo do administrador do SO gerenciar os certificados e garantir que a JVM vai entregar o certificado correto para a aplicação.

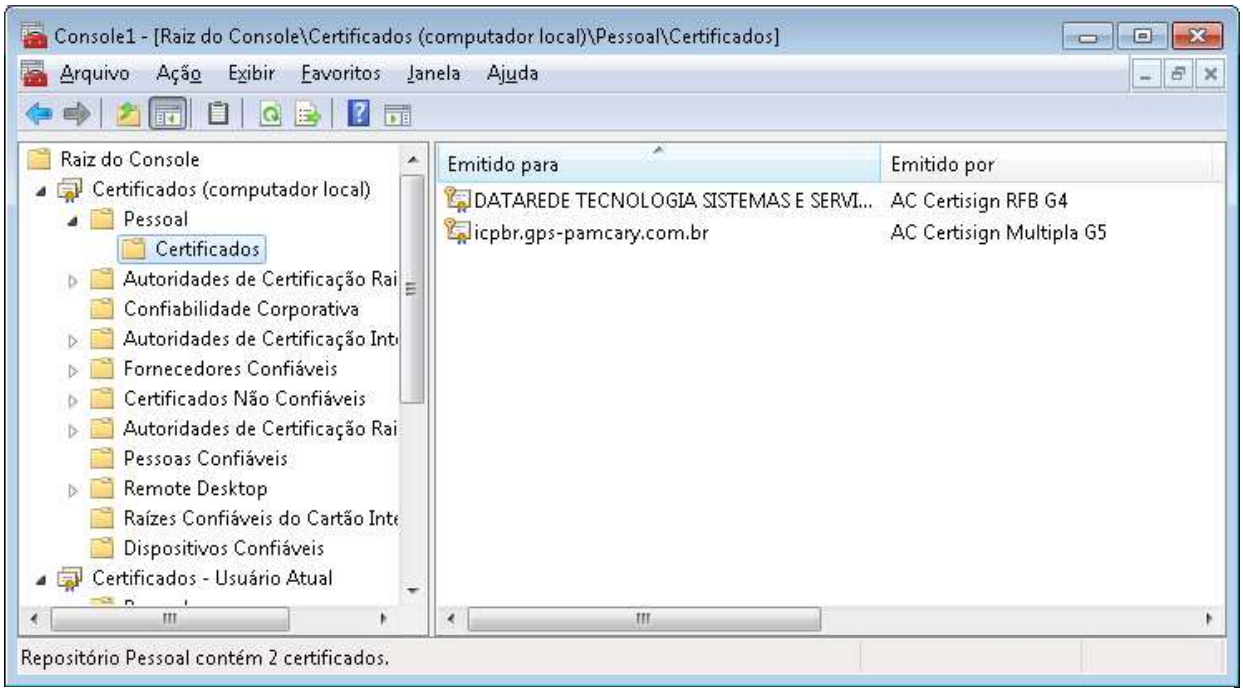
Esse método pode ser mais fácil de configurar no início, mas pode se tornar um problema caso a aplicação faça uso de diversas conexões SSL com diferentes provedores de serviço.

.NET:

No ambiente Windows, antes de iniciar a codificação é necessário importar o Certificado Digital em conjunto com sua chave privada (normalmente no formato .pfx) para o banco de dados local do servidor:

1. Vá em **Iniciar>Executar** digite **mmc** e clique em **OK**.
2. Na aba **Arquivo**, clique **Adicionar/Remover Snap-in**.
3. Na caixa de diálogo **Adicionar ou Remover Snap-ins**, clique em **Certificados**, em seguinda clique em **Adicionar**, escolha **Conta de computador**, clique em **Avançar**, e então em **Concluir**.
4. Na caixa de diálogo **Adicionar ou Remover Snap-ins**, clique em **OK.**
5. No menu esquerdo, expanda o item **Certificados (Computador local)**, e então **Pessoal**.
6. Clique com o botão direito em **Pessoal**, clique **Todas as tarefas**, clique **Importar**, e então **Avançar**.
7. No campo **Nome do arquivo**, entre com o caminho do arquivo PFX, e então em **Avançar**.
8. No campo **Senha**, entre com a senha do seu PFX, e então clique em **Avançar** duas vezes.
9. Clique em **Finalizar** e **OK**.

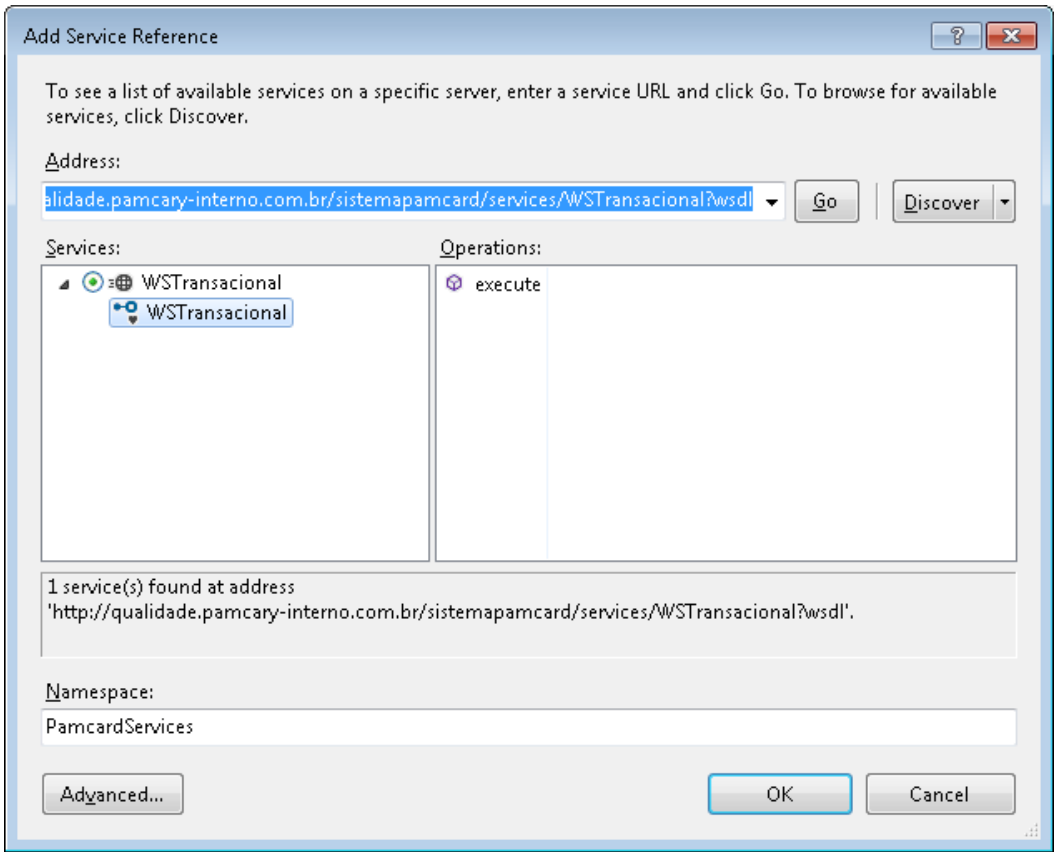
MMC com o Snap-in de Certificados da Máquina



Feito esses passos, o certificado já está pronto para uso e referenciado na aplicação.

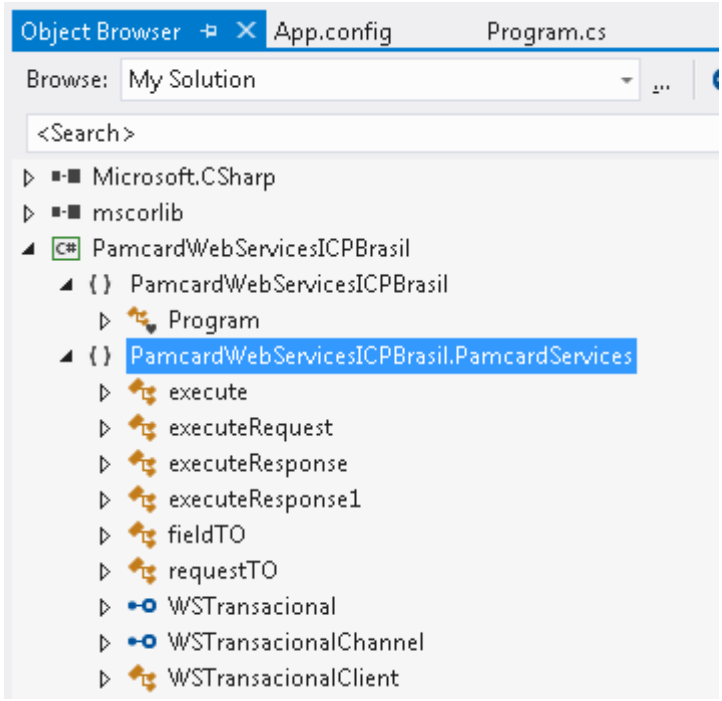
***Nota**: cada tipo de aplicação (Web/Client) que vai consumir o Web Service pode requerer uma configuração diferente na hora de armazenar o certificado. A estratégia desenvolvida acima funcionou em uma prova de conceito de aplicação Client. Para maiores informações, consulte a documentação oficial aqui e aqui.*

Com o certificado armazenado no local correto, utilizando o Visual Studio, adicione uma referência a um Serviço no seu projeto. Para adicionar essa referência será necessário possuir o endereço do WSDL ou o arquivo salvo em algum local do servidor (prefira esta última opção). A figura abaixo ilustra essa configuração no VS.



Adicionando uma referência a um Serviço

Dê ao nome da referência o que melhor fizer sentido para o *Web Service* que será consumido. Após adicionar a referência, o VS vai criar todas as classes necessárias para interagir com o serviço referenciado. A figura abaixo ilustra a disposição dos objetos que fazem referência ao *Web Service* no projeto.



Referências criadas automaticamente pelo VS para consumir o WS

Adicionada a referência, acesse o arquivo de configurações da aplicação (`App.config` ou`Web.config` a depender do tipo de projeto) e complemente com as informações dispostas na listagem abaixo.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <startup>
    <supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.5" />
  </startup>
  <system.serviceModel>
    <behaviors>
      <endpointBehaviors>
        <behavior name="authPamcard">
          <clientCredentials>
            <clientCertificate findValue="MINHA EMPRESA LTDA:01010101000100"
storeLocation="LocalMachine" storeName="My" x509FindType="FindBySubjectName"/>
          </clientCredentials>
        </behavior>
      </endpointBehaviors>
    </behaviors>
    <bindings>
      <basicHttpBinding>
        <binding name="WSTransacionalBinding">
          <security mode="Transport">
            <transport clientCredentialType="Certificate" />
          </security>
        </binding>
      </basicHttpBinding>
    </bindings>
    <client>
      <endpoint address="https://ENDERECO DO ENDPOINT"
          binding="basicHttpBinding" bindingConfiguration="WSTransacionalBinding"
          behaviorConfiguration="authPamcard"
contract="PamcardServices.WSTransacional" name="WSTransacional" />
    </client>
  </system.serviceModel>
</configuration>
```

Há alguns pontos dessa configuração que devem ser esclarecidos. Primeiramente, no nó ***behavior, endpointBehaviors, behavior*** deve ser adicionada a informação sobre o certificado que será utilizado pelo cliente. Essa informação é definida no nó ***clientCredentials, clientCertificate***.

No nó ***clientCertificate*** é feita a referência ao certificado digital adicionado no servidor. Para que o .NET *framework* consiga acessar o certificado é fundamental apontar em qual local o certificado foi instalado. No caso do exemplo, o Certificado está no repositório Local.

Para escolher entre os certificados do repositório, utilizamos os itens **x509FindType** e **findValue** para procurar por um certificado cujo *Common Name* é, por exemplo, "MINHA EMPRESA INC: 010000102-01". É possível adaptar essa busca conforme a necessidade. A lista de todos os parâmetros possíveis para este campo estão neste link.

Após configurar o repositório de busca do certificado, é necessário especificar que o certificado será utilizado para autenticar o cliente no servidor. Essa configuração é feita no nó ***basicHttpBinding, binding, security***.

Com as configurações realizadas, é possível implementar o código para interagir com o *Web Service*. A listagem abaixo exemplifica essa implementação. Como toda a configuração foi feita por meio do arquivo XML discutido acima, não é preciso informar em nenhuma parte do código o tipo de transporte, certificado ou qualquer informação sobre autenticação. Tudo é feito em *background*pelo *framework* do .NET.

```
static void callService()
{
    System.Console.Out.WriteLine("Declarando o endpoint e o parâmetro de entrada.");
    PamcardServices.WSTransacionalClient wsTransacional = new
PamcardServices.WSTransacionalClient();

    PamcardServices.execute request;
    request = new PamcardServices.execute();
    request.arg0 = new PamcardServices.requestTO();

    System.Console.Out.WriteLine("Declarando os valores de entrada que serão
compostos no Request");
    System.Console.Out.WriteLine("Definindo o contexto e os 'fieldTO'");

    request.arg0.context = "";

    //Como exemplo, só adicionamos um campo, mas é possível adicionar quantos forem
necessários no Array
    PamcardServices.fieldTO field1 = new PamcardServices.fieldTO();
    field1.key = "";
    field1.value = "";

    request.arg0.fields = new PamcardServices.fieldTO[1];
    request.arg0.fields.SetValue(field1, 0);

    try
    {
        System.Console.Out.WriteLine("Executando o request.");
        PamcardServices.executeResponse response = wsTransacional.execute(request);

        System.Console.Out.WriteLine("Tratando da resposta.");

        PamcardServices.fieldTO[] itensResposta = response.@return;

        if (itensResposta != null)
        {
            foreach (PamcardServices.fieldTO item in itensResposta)
            {
                System.Console.Out.WriteLine(item.key + ": " + item.value);
            }
        }
```

```
else
        {
            System.Console.Out.WriteLine("Requisição OK, mas resposta vazia. Algo
pode estar errado.");
        }

    }

catch (Exception ex)
    {
        System.Console.Out.WriteLine("Erro durante a requisição: ");
        System.Console.Out.WriteLine(ex.Message);
    }
    finally
    {
        System.Console.In.ReadLine();
    }
}
```

Agora é possível focar apenas nas regras de negócio e nas variáveis de entrada e saída da aplicação.

Apêndices

- ## WSDL

O XML do WSDL é formado por apenas 02 tags principais: context / fields e não há um XSD para este processo de integração, pois a validação é de acordo com o conjunto de parâmetros de cada uma das operações. Para obter o WSDL inclua "?WSDL" no final da URL utilizada

No campo **context** deverá ser enviada a função correspondente à funcionalidade de acordo com a tabela disponibilizada no item "Funcionalidades" desta documentação.

No campo **fields** deverá ser informado um conjunto de tags, contendo a chave e o valor, de acordo com a funcionalidade utilizada. Os parâmetros serão descritos para cada uma das funcionalidades como Campo de Entrada.

A estrutura das informações do retorno de cada funcionalidade é idêntico à estrutura da requisição, entretanto os parâmetros retornados serão os descritos como Campos de Retorno.

Abaixo os endereços para consumo do XML:

Utilize preferencialmente o endereço abaixo para o desenvolvimento da sua integração (Padrão Document):

Endereço do Ambiente de Homologação:
https://qualidade.gps-pamcary.com.br/sistemapamcard/services/WSTransacional

Endereço do Ambiente de Produção:
https://www.gps-pamcary.com.br/sistemapamcard/services/WSTransacional

- ## Exemplos

Segue abaixo o exemplo dos XML´s de chamada e retorno do **WSTransacional**:

Entrada

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:web="http://webservice.pamcard.jee.pamcary.com.br">
  <soapenv:Header/>
  <soapenv:Body>
    <web:execute>
      <arg0>
        <context>FindCard</context>
        <fields>
          <key>viagem.contratante.documento.numero</key>
          <value>06181156000100</value>
        </fields>
        <fields>
          <key>viagem.cartao.numero</key>
          <value>4417810025749012</value>
        </fields>
      </arg0>
    </web:execute>
  </soapenv:Body>
</soapenv:Envelope>
```

Retorno

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/">
```

```
   <env:Header/>
   <env:Body>
      <ns2:executeResponse xmlns:ns2="http://webservice.pamcard.jee.pamcary.com.br">
         <return>
            <fields>
               <key>mensagem.codigo</key>
               <value>4</value>
            </fields>
            <fields>
               <key>mensagem.descricao</key>
               <value>Erro de validação do cartão na processadora [Conta/Cartão não cadastrado - Verifique
CODM99]</value>
            </fields>
         </return>
      </ns2:executeResponse>
   </env:Body>
</env:Envelope>
```

Apêndices

Adaptador

Este adaptador é destinado para a integração com o Sistema Pamcard através de um sistema de arquivos texto parametrizado.
Para a utilização deste pacote será encaminhado um e-mail com a estrutura de arquivos, juntamente com os arquivos necessários a execução do mesmo.

Para a utilização da integração, o sistema de origem deverá gerar arquivos de entrada (um arquivo para cada transação), e deverá depositá-lo na pasta correspondente, conforme detalhado abaixo.

Cada arquivo de entrada gerado será correspondente a uma única operação e, após o processamento, será gerado um arquivo de saída com o mesmo nome do arquivo de entrada na pasta correspondente.

Este adaptador está homologado para o ambiente Windows.

Instalação

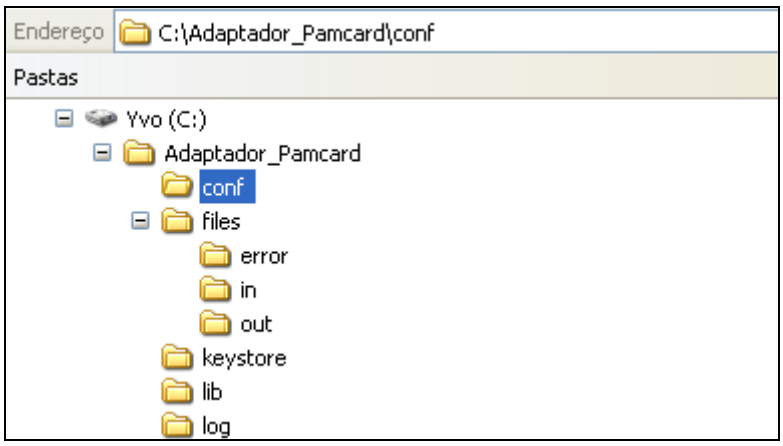
Salvar e descompactar o arquivo "Adaptador_Pamcard.zip" no diretório destinado ao processamento desta integração.

Atenção: Os arquivos vêm configurados para utilização do diretório "c:/", caso utilize um diretório diferente, esta informação deverá ser atualizada nos seguintes arquivos:

\Adaptador_Pamcard\conf\wspamcard-client.properties
\Adaptador_Pamcard\keystore\clientkeystore.properties
\Adaptador_Pamcard\wspamcard-client.bat

Estrutura de Diretórios (versão windows)

O software Client deve utilizar a seguinte estrutura de diretórios:



Pasta Adaptador_Pamcard

Este diretório contém os arquivos de inicialização do Adaptador:

- wspamcard-client.bat – script para inicialização do processo

Pasta /conf

Este diretório contém os arquivos de configuração do Adaptador:

- log4j.properties: Arquivo de configuração para a API log4j.
- wspamcard-client.properties: Arquivo de configuração do Adaptador. Os seguintes parâmetros deverão ser ajustados:
  - ✓ URL de conexão com o Web Service Transacional: transacional.url= **https://qualidade.gps-pamcary.com.br/sistemapamcard/services/WSTransacional**
  - ✓ Intervalo em segundos para leitura de um lote de arquivos: file.read.interval=10
  - ✓ Parâmetro para configurar tempo de resposta do Roteirizador: request.timeout=180

Pasta /files/error

Diretório onde serão colocados os arquivos de entrada para os quais houve algum erro de validação nos parâmetros informados.

Pasta files/in

Diretório onde devem ser colocados os arquivos de entrada.

Obs.: A ordem dos campos não será considerada para a leitura dos campos, podendo estes estar dispostos de forma aleatória.

Pasta files/out

Diretório onde serão colocados os arquivos de saída processados pelo adaptador. Este arquivo terá a informação da mensagem do retorno enviado pelo servidor de integração Pamcard e demais campos de saída, conforme a operação realizada.

A mensagem de retorno indicará se o processo ocorreu com sucesso ou não.

Pasta /keystore

Este diretório contém os arquivos de configuração do Adaptador para o certificado digital:

- *[nome_do_certificado]*.pfx: Arquivo do certificado digital, nomeado conforme a identificação do cliente.
- clientkeystore.properties: Arquivo de propriedades do certificado digital. Os seguintes parâmetros deverão ser ajustados:
  - o identificação do caminho do certificado: pamcard.prop.keystore.client.keystore.path=C:/ClienteWSTransacional/keystore/*[nome_do_certificado]*.pfx
  - o identificação da senha do certificado: pamcard.prop.keystore.client.keystore.pass=*[senha do certificado]*

IMPORTANTE: a indicação da pasta Arquivo de Programas poderá sofrer alterações conforme a instalação do Windows: "Arquiv~1" (para sistema de arquivos com 8 bits) ou "Program Files" (para versões em inglês).

Pasta /lib

Bibliotecas utilizadas pelo adaptador.

Pasta /log

Diretório do arquivo de log gerado pelo adaptador, a saber:

- o wspamcard-client.log: Arquivo gerado pelo adaptador, contendo o log de todas as transações

realizadas.

Geração do Arquivo

Segue abaixo a sequência ideal para o correto processamento dos arquivos:

1. O sistema legado do cliente gera o arquivo de acordo com as parametrizações necessárias para a operação a ser realizada. Este arquivo deverá ter a extensão **.TMP**, ou seja, [nome do arquivo escolhido pelo cliente].tmp;
2. Caso o arquivo tenha sido gerado em outra pasta, deverá ser copiado para a pasta IN do pacote Adaptador;
3. Após finalizada a geração e/ou cópia do arquivo com sucesso, renomear a extensão do arquivo com uma extensão diferente de **.TMP**.
4. O arquivo será processado no próximo ciclo de processamento

Configuração

Para o correto funcionamento do sistema as configurações abaixo deverão ser verificadas:

Configuração 01:

Na pasta "C:\Adaptador_Pamcard\" editar o arquivo "wspamcard-client.bat"

Configurar os seguintes itens:

✓ **Definir o caminho do JAVA**
✓ **Definir o diretório onde ficará o Adaptador (Client)**

```
@echo off
rem ------------------------------------------------------------------------
rem WS PAMCARD CLIENT
rem ------------------------------------------------------------------------

set JRE_HOME=C:\Arquivos de programas\Java\jre1.5.0_06\

set WSCLIENT_HOME=C:\Adaptador_Pamcard

cd %WSCLIENT_HOME%

set WSCLIENT_PATH=lib\axis-1.3.jar;
set WSCLIENT_PATH=%WSCLIENT_PATH%;lib\commons-discovery-0.2.jar;
set WSCLIENT_PATH=%WSCLIENT_PATH%;lib\commons-logging-1.1.jar;
set WSCLIENT_PATH=%WSCLIENT_PATH%;lib\hst-core-exception.jar;
set WSCLIENT_PATH=%WSCLIENT_PATH%;lib\hst-core-helper.jar;
set WSCLIENT_PATH=%WSCLIENT_PATH%;lib\hst-core-to.jar;
set WSCLIENT_PATH=%WSCLIENT_PATH%;lib\hst-core-util.jar;
set WSCLIENT_PATH=%WSCLIENT_PATH%;lib\hst-security-certif.jar;
set WSCLIENT_PATH=%WSCLIENT_PATH%;lib\hst-util-validate.jar;
set WSCLIENT_PATH=%WSCLIENT_PATH%;lib\hst-util.jar;
set WSCLIENT_PATH=%WSCLIENT_PATH%;lib\jaxrpc-1.1.jar;
set WSCLIENT_PATH=%WSCLIENT_PATH%;lib\lista.txt;
set WSCLIENT_PATH=%WSCLIENT_PATH%;lib\log4j-1.2.9.jar;
set WSCLIENT_PATH=%WSCLIENT_PATH%;lib\pamcard-ws-client.jar;
set WSCLIENT_PATH=%WSCLIENT_PATH%;lib\saaj.jar;
set WSCLIENT_PATH=%WSCLIENT_PATH%;lib\servlet-api.jar;
```

```
set WSCLIENT_PATH=%WSCLIENT_PATH%;lib\wsdl4j.jar;

echo ------------------------------------------------------------------------
echo .
echo   WS PAMCARD CLIENT
echo .
echo   JRE_HOME: %JRE_HOME%
echo .
echo   WSCLIENT_HOME: %WSCLIENT_HOME%
echo .
echo   WSCLIENT_PATH: %WSCLIENT_PATH%
echo .
echo ------------------------------------------------------------------------
echo .

"%JRE_HOME%/bin/java" -cp %WSCLIENT_PATH%
br.com.pamcary.jee.pamcard.webservice.client.pamcard.service.WSPamcardClientService

pause
```

Configuração 02:

Na pasta "C:\Adaptador_Pamcard\keystore" editar o arquivo "clientkeystore.properties",

Configurar os seguintes itens:

✓ **Registrar o nome do certificado**
✓ **Registrar a senha do certificado**

```
pamcard.prop.keystore.client.keystore.path=C:/Adaptador_Pamcard/keystore/[nome do certificado]
pamcard.prop.keystore.client.keystore.pass=[senha do certificado]
```

Configuração 03:

Na pasta "C:\Adaptador_Pamcard\conf" editar o arquivo "wspamcard-client.properties",

Configurar os seguintes itens:

✓ **Definir a URL de conexao com o Sistema Pamcard**
Endereço do Ambiente de Homologação:
https://qualidade.gps-pamcary.com.br/sistemapamcard/services/WSTransacional
Endereço do Ambiente de Produção:
https://www.gps-pamcary.com.br/sistemapamcard/services/WSTransacional
✓ **Definir o caminho completo do certificado**
✓ **Definir o CNPJ do Contratante – Sem Edição**

```
#diretorio de entrada de arquivos
file.input.dir=files/in/

#diretorio de saida de arquivos
file.output.dir=files/out/

#diretorio de arquivos com erro
```

file.error.dir=files/error/

#intervalo (em segundos) para leitura de um lote de arquivos
file.read.interval=10
request.timeout=12000 ➔ configura timeout para resposta do roteirizador

#URL de conexao com o Sistema Pamcard
wspamcard.url= **https://qualidade.gps-pamcary.com.br/sistemapamcard/services/WSTransacional**

#caminho completo para o keystore
keystore.path=**c:/Adaptador_Pamcard/keystore/clientkeystore.properties**

#nome do certificado no keystore
keystore.alias=***[CNPJ do Contratante, sem edição]***

Configuração 04:

Configuração para clientes que utilizam **PROXY** para acesso a Internet:

✓ **Editar o arquivo "wspamcard-client.bat".**

Na linha onde contém o comando:
"%JRE_HOME%/bin/java" -cp
%WSCLIENT_PATH%br.com.pamcary.jee.pamcard.webservice.client.pamcard.service.WSPamcardClientService

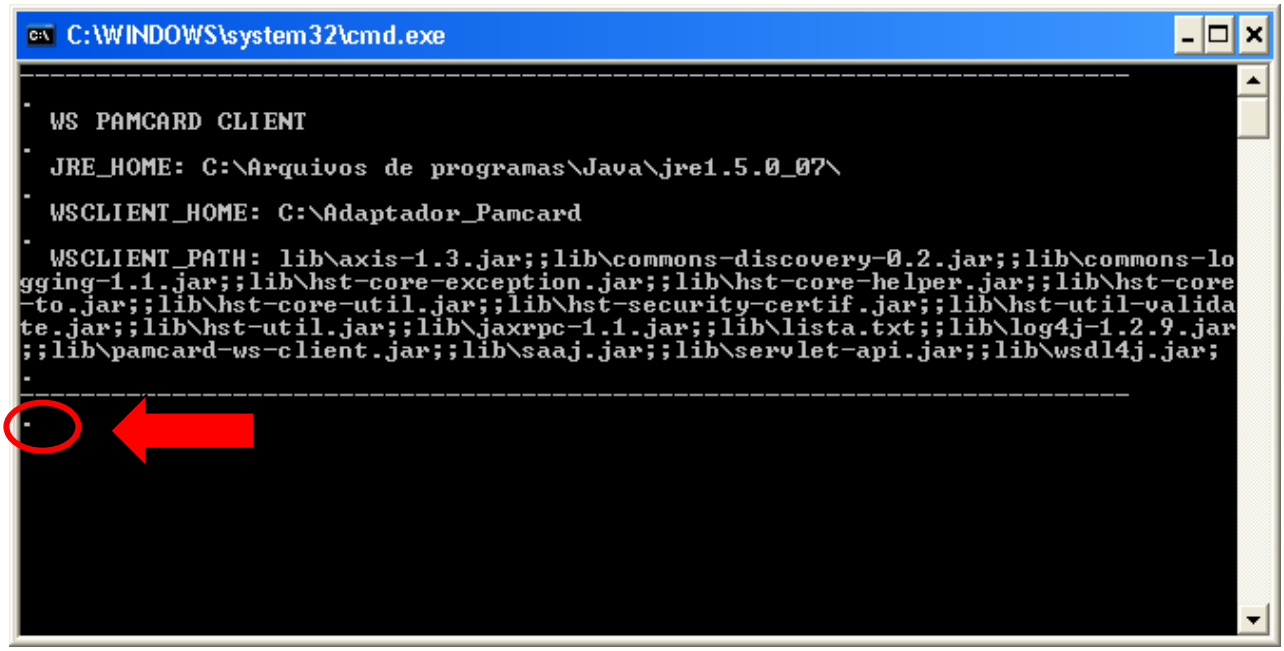
Adicionar as seguintes variáveis **(em negrito)**, entre os colchetes coloque as informações do seu servidor proxy, e do usuário que irá realizar a autenticação:

"%JRE_HOME%/bin/java" **-Dhttp.proxyHost=[servidor proxy] -Dhttp.proxyPort=[porta do proxy] -Dhttp.proxyUserName=[usuario] -Dhttp.proxyPassword=[senha]** -cp %WSCLIENT_PATH% br.com.pamcary.jee.pamcard.webservice.client.pamcard.service.WSPamcardClientService

Se o proxy não possuir usuário e senha eliminar do comando os parâmetros:
-Dhttp.proxyUserName=[usuario]
-Dhttp.proxyPassword=[senha]

- ## Inicialização do sistema

Executar o arquivo "wspamcard-client.bat"
Irá abrir uma tela do MS-DOS e ficar somente um ponto no rodapé da tela. Isto indicará que o Client esta funcionando normalmente. Caso apareça qualquer mensagem no rodapé da tela do MS-DOS, indica que houve alguma falha na configuração do Client.


C:\WINDOWS\system32\cmd.exe
WS PAMCARD CLIENT
JRE_HOME: C:\Arquivos de programas\Java\jre1.5.0_07\
WSCLIENT_HOME: C:\Adaptador_Pamcard
WSCLIENT_PATH: lib\axis-1.3.jar;;lib\commons-discovery-0.2.jar;;lib\commons-logging-1.1.jar;;lib\hst-core-exception.jar;;lib\hst-core-helper.jar;;lib\hst-core-to.jar;;lib\hst-core-util.jar;;lib\hst-security-certif.jar;;lib\hst-util-validate.jar;;lib\hst-util.jar;;lib\jaxrpc-1.1.jar;;lib\lista.txt;;lib\log4j-1.2.9.jar;;lib\pamcard-ws-client.jar;;lib\saaj.jar;;lib\servlet-api.jar;;lib\wsdl4j.jar;