AT autoridade tributária e aduaneira

MANUAL WEBSERVICES DO STADA-EXPORTAÇÃO V1.0

Índice

VERSÕES:

DATA	AUTOR	VERSÃO	COMENTÁRIO
18-11-2021	AT	1.0	Versão para publicação

1. Estrutura do envio de dados à AT (SOAP)

Neste ponto pretende-se descrever a metodologia do Webservice do STADA-EXP.

O serviço disponibiliza vários métodos que permitem as seguintes ações no âmbito do STADA-EXP:

- enviar mensagens (5 métodos)
- receber mensagens (2 métodos)
- obter documentos (1 método)

O Webservice é efetuado segundo o protocolo SOAP e é constituído por duas secções:

- **SOAP:Header:**
  - Esta secção inclui todos os campos de autenticação do utilizador que vai ser responsável pela invocação do Webservice;
  - O utilizador corresponderá ao NIF do sujeito passivo ou a um subtilizador desse NIF. No caso de sujeito passivo coletivo, deve indicar obrigatoriamente um subutilizador;
  - A criação de subtilizadores e respetiva credenciação deve ser efetuada no Portal das Finanças utilizando a opção:
    - "GUE - Credenciação".

    Para envio de mensagens de exportação via Webservices deve escolher a **modalidade XML**.

    Caso já possua credenciação para envio de mensagens de Exportação com acesso ao EFAPI não necessita de fazer nova credenciação porque já terá a modalidade XML atribuída.
- **SOAP:Body**
  - Esta secção contém os dados referentes ao sistema STADA-EXP, os quais se detalham no ponto SOAP:Body.

2. SOAP:Header

O desenho do Header tem como requisito garantir a confidencialidade dos dados de autenticação e a impossibilidade de reutilização dos mesmos em ataques Man-in-the-middle (MITM). Por este motivo, só serão aceites invocações que respeitem os procedimentos de encriptação.

O SOAP:Header é construído de acordo com o standard WS-Security, definido pela OASIS e recorrendo à definição do Username Token Profile 1.1, também definido pela mesma organização.
Na seguinte tabela, detalha-se a forma de construção de cada campo, de acordo com as necessidades de segurança específicas do sistema de autenticação do Portal das Finanças.

Parâmetro	Descrição	Obrigatório (S/N)	Tipo de Dados
H.1 – Utilizador (Username)	Identificação do utilizador que vai submeter os dados, composto da seguinte forma e de acordo com a autenticação do Portal das Finanças: **<NIF>/<SUBTILIZADOR>** Exemplos: • 111111111 • 555555555/1 (subtilizador n.º 1) • 555555555/0002 (subtilizador n.º 2) • 555555555/1234 (subtilizador n.º 1234)	S	String
H.2 – Nonce	Chave simétrica gerada a cada pedido e para cifrar o conteúdo dos campos H.3 - Password e H.4 - Created. Cada invocação do Webservice deverá conter esta chave gerada aleatoriamente e a qual não pode ser repetida. Para garantir a confidencialidade, a chave simétrica tem de ser cifrada com a chave pública do Sistema de Autenticação da AT de acordo com o algoritmo RSA e codificada em Base 64. A chave pública do sistema de autenticação do Portal das Finanças, em produção, deve ser obtida na opção de Produtores de Software existente no Portal e-Fatura: https://faturas.portaldasfinancas.gov.pt/. O campo é construído de acordo com o seguinte procedimento:	S	String (base64)

<table>
<tr><th>Parâmetro</th><th>Descrição</th><th>Obrigatório (S/N)</th><th>Tipo de Dados</th></tr>
<tr><td></td><td>$Nonce := Base64(C_{RSA,Kpub_{SA}}(K_s))$
<table><tr><td>Ks :=</td><td>array de bytes com a chave simétrica de 128 bits, produzida de acordo com a norma AES.</td></tr><tr><td>$C_{RSA,Kpub_{SA}}$:=</td><td>Função de cifra da chave simétrica com o algoritmo RSA utilizando a chave pública do sistema de autenticação $(Kpub_{SA})$</td></tr><tr><td>Base64 :=</td><td>Codificação em Base 64 do resultado.</td></tr></table></td><td></td><td></td></tr>
<tr><td>H.3 – Password</td><td>O campo Password deverá conter a senha do utilizador/subtilizador, a mesma que é utilizada para entrar no Portal das Finanças.
Esta Password tem de ser cifrada através da chave simétrica do pedido (ver H.2 – Nonce) e codificado em Base64.
$Password := Base64(C_{K_S}^{AES,ECB,PKCS5Padding}(SenhaPF))$
<table><tr><td>SenhaPF:=</td><td>Senha do utilizador definido no campo H.1 – Username;</td></tr><tr><td>$C_{K_S}^{AES,ECB,PKCS5Padding}$:=</td><td>Função de cifra utilizando o algoritmo AES, Modelo ECB, PKCS5Padding e a chave simétrica do pedido (Ks).</td></tr><tr><td>Base64 :=</td><td>Codificação em Base 64 do resultado.</td></tr></table></td><td>S</td><td>string (base64)</td></tr>
</table>

<table>
<tr><th>Parâmetro</th><th>Descrição</th><th>Obrigatório (S/N)</th><th>Tipo de Dados</th></tr>
<tr><td>H.4 – Data de sistema (Created)</td><td>
O campo Created deverá conter a data e hora de sistema da aplicação que está a invocar o webservice.

Esta data é usada para validação temporal do pedido, pelo que é crucial que o sistema da aplicação cliente tenha o seu relógio certo.

Sugere-se a sincronização com o Observatório Astronómico de Lisboa:

http://www.oal.ul.pt/index.php?link=acerto

A zona temporal deste campo deverá estar definida para UTC e formatado de acordo com a norma ISO 8601 tal como é definido pelo W3C:

http://www.w3.org/QA/Tips/iso-date

http://www.w3.org/TR/NOTE-datetime

Exemplo: 2017-01-01T19:20:30.45Z

Este campo é cifrado com a chave de pedido (Ks) e codificada em Base 64.

$$Created := Base64(C_{K_S}^{AES,ECB,PKCS5Padding}(Timestamp))$$
<table>
<tr><td>Timestamp :=</td><td>data hora do sistema (UTC);</td></tr>
<tr><td>$C_{K_S}^{AES,ECB,PKCS5Padding} :=$</td><td>Função de cifra utilizando o algoritmo AES, Modelo ECB, PKCS5Padding e a chave simétrica do pedido (K_S).</td></tr>
<tr><td>Base64 :=</td><td>Codificação em Base 64 do resultado.</td></tr>
</table>
</td><td>S</td><td>string (base64)</td></tr>
</table>

2.1. Exemplo de SOAP:Header

Como resultado da aplicação das regras de construção anteriores será produzido um *header* de pedido SOAP tal como se apresenta no seguinte exemplo:

```
<S:Header>
    <wss:Security xmlns:wss="http://schemas.xmlsoap.org/ws/2002/12/secext">
        <wss:UsernameToken>
            <wss:Username>500000016/15</wss:Username>
              <wss:Password>ikCyRV+SWfvZ5c6Q0bhrBQ==</wss:Password>
            <wss:Nonce>
                fkAHne7cqurxpImCfBC8EEc2vskyUyNofWi0ptIijYg4gYCxir++unzfPVPpusloEtmLkcZjf+E6T9/76
                tsCqdupUkxOhWtkRH5IrNwmfEW1ZGFQgYTF21iyKBRzMdsJMhhHrofYYV/YhSPdT4dlgG0tk9
                Z736jFuw061mP2TNqHcR/mQR0yW/AEOC6RPumqO8OAfc9/b4KFBSfbpY9HRzbD8bKiTo20n
                0PtamZevCSVHht4yt/Xwgd+KV70WFzyesGVMOgFRTWZyXyXBVaBrkJS8b6PojxADLcpWRnw5
                +YeOs3cPU2o1H/YgAam1QuEHioCT2YTdRt+9p6ARNElFg==
            </wss:Nonce>
            <wss:Created>>YEWoIoqIY5DOD11SeXz+0i4b/AJg1/RgNcOHOYpSxGk</wss:Created>
        </wss:UsernameToken>
    </wss:Security>
</S:Header>
```

3. SOAP:Body

Nesta secção são definidos os métodos para submissão de informação relativa à declaração de exportação ou ao seu circuito e obtenção de documentos e mensagens de resposta.

Os métodos definidos para este Webservice são:

Método	Descrição
obterDocumentosPDF	Método que permite a um operador obter documentos PDF, que já tenham sido criados, a partir de um número de aceitação e um número de modelo.
obterMensagensEntregues	Método que permite ao operador obter mensagens de resposta XML já entregues anteriormente. A pesquisa pode ser feita por datas, número de aceitação ou número de referência local. O método recebe ainda um offset de modo a controlar a quantidade de mensagens recebidas e retorna um indicador que diz se há mais.
obterMensagensNaoEntregues	Método que permite ao operador obter as últimas mensagens XML que ainda não foram entregues. A pesquisa retorna um indicador que diz se há mais mensagens.
processaMensagemEX001A	Método que permite enviar as mensagens: **EX001A** – Declaração (Criação e alteração de dados)
processaMensagemEX007A	Método que permite enviar as mensagens: **EX007A** – Pedido de Anulação da Declaração
processaMensagemEX008A	Método que permite enviar as mensagens: **EX008A** – Comunicação de Mercadoria Apresentada
processaMensagemEX011A	Método que permite enviar as mensagens: **EX011A** – Resposta à Proposta de Retificação
processaMensagemEX013A	Método que permite enviar as mensagens: **EX013A**– Informação sobre Localização de Movimento

3.1. Exemplo de SOAP:Body – Método obterDocumentoPDF

Como resultado da aplicação do método **obterDocumentosPDF** será produzido um *body* de pedido SOAP tal como se apresenta no seguinte exemplo:

```
<soapenv:Body>
   <web:obterDocumentosPDF>
          <numeroAceitacao>15PT00011520000200</numeroAceitacao>
          <versao>1</versao>
          <revisao>0</revisao>
          <modelo>3</modelo>
   </web:obterDocumentosPDF>
</soapenv:Body>
```

Como resultado desta invocação será devolvido o seguinte:

```
<ResultadoDocumentos>
      <FicheiroResposta>
            <nomeFicheiro> </nomeFicheiro>
            <respostaFicheiro> Documento em base64</respostaFicheiro>
      </FicheiroResposta>
</ResultadoDocumentos>
```

3.2. Exemplo de SOAP:Body – Método obterMensagensNaoEntregues

Como resultado da aplicação do método **obterMensagensNaoEntregues** será produzido um *body* de pedido SOAP tal como se apresenta no seguinte exemplo:

```
<soapenv:Body>
      <web:obterMensagensNaoEntregues>
      </web:obterMensagensNaoEntregues>
</soapenv:Body>
```

A este pedido o sistema responde da seguinte forma:

```
<env:Body>
   <m:obterMensagensNaoEntreguesResponse >
     <return>
       <java:Resultado>
         <java:CodigoResultado>0</java:CodigoResultado>
         <java:Resultado>Mensagens encontradas.</java:Resultado>
       </java:Resultado>
       <java:mensagens>
         <java:Xml>Mensagem XML em base64</java:Xml>
         <java:NomeFicheiro>240033990.W1992815</java:NomeFicheiro>
       </java:mensagens>
       <java:mensagens>
         <java:Xml>Mensagem XML em base64</java:Xml>
         <java:NomeFicheiro>240033990.W1992812</java:NomeFicheiro>
       </java:mensagens>
       <java:mensagens>
         <java:Xml>Mensagem XML em base64</java:Xml>
         <java:NomeFicheiro>240033990.W1992813</java:NomeFicheiro>
       </java:mensagens>
       <java:existeMensagemSeguinte>false</java:existeMensagemSeguinte>
     </return>
   </m:obterMensagensNaoEntreguesResponse>
 </env:Body>
```

3.3. Exemplo de SOAP:Body – Método obterMensagensEntregues

Como resultado da aplicação do método **obterMensagensEntregues** será produzido um *body* de pedido SOAP tal como se apresenta no seguinte exemplo:

```
<soapenv:Body>
   <web:obterMensagensEntregues>
            <numeroAceitacao>66</numeroAceitacao>
            <numeroReferenciaLocal>44</numeroReferenciaLocal>
           <dataInicio>2018-05-23</dataInicio>
           <dataFim>2018-05-24</dataFim>
           <offset>0</offset>
   </web:obterMensagensEntregues>
 </soapenv:Body>
```

A este pedido o sistema responde da seguinte forma:

```
<env:Body>
   <m:obterMensagensNaoEntreguesResponse >
     <return>
       <java:Resultado>
         <java:CodigoResultado>0</java:CodigoResultado>
         <java:Resultado>Mensagens encontradas.</java:Resultado>
       </java:Resultado>
       <java:mensagens>
         <java:Xml>Mensagem XML em base64</java:Xml>
         <java:NomeFicheiro>240033990.W1992815</java:NomeFicheiro>
       </java:mensagens>
       <java:mensagens>
         <java:Xml>Mensagem XML em base64</java:Xml>
         <java:NomeFicheiro>240033990.W1992812</java:NomeFicheiro>
       </java:mensagens>
       <java:mensagens>
         <java:Xml>Mensagem XML em base64</java:Xml>
         <java:NomeFicheiro>240033990.W1992813</java:NomeFicheiro>
       </java:mensagens>
       <java:existeMensagemSeguinte>false</java:existeMensagemSeguinte>
     </return>
   </m:obterMensagensNaoEntreguesResponse>
 </env:Body>
```

3.4. Exemplo de SOAP:Body – Método processaMensagemEX001A

Como resultado da aplicação do método **processaMensagemEX001A** será produzido um *body* de pedido SOAP tal como se apresenta no seguinte exemplo:

```
<soapenv:Body>
    <web:processaMensagemEX001A>
        <SubmeterDeclaracaoMensagem>
            <java:Mensagem>Mensagem EX001A em base64</java:Mensagem>
        </SubmeterDeclaracaoMensagem>
    </web:processaMensagemEX001A>
</soapenv:Body>
```

A este pedido o sistema responde da seguinte forma:

```
<env:Body>
    <m:processaMensagemXMLEX001AResponse >
        <return>
            <java:CodigoResultado>0</java:CodigoResultado>
            <java:Resultado>Mensagem submetida com sucesso<java:Resultado>
        </return>
    </m:processaMensagemXMLEX001AResponse>
</env:Body>
```

3.5. Exemplo de SOAP:Body – Método processaMensagemEX007A

Como resultado da aplicação do método **processaMensagemEX007A** será produzido um *body* de pedido SOAP tal como se apresenta no seguinte exemplo:

```
<soapenv:Body>
    <web:processaMensagemEX007A>
        <SubmeterDeclaracaoMensagem>
            <java:Mensagem>Mensagem EX007A em base64</java:Mensagem>
        </SubmeterDeclaracaoMensagem>
    </web:processaMensagemEX007A>
</soapenv:Body>
```

A este pedido o sistema responde da seguinte forma:

```
<env:Body>
    <m:processaMensagemXMLEX007AResponse >
        <return>
            <java:CodigoResultado>0</java:CodigoResultado>
            <java:Resultado>Mensagem submetida com sucesso<java:Resultado>
        </return>
    </m:processaMensagemXMLEX007AResponse>
</env:Body>
```

3.6. Exemplo de SOAP:Body – Método processaMensagemEX008A

Como resultado da aplicação do método **processaMensagemEX008A** será produzido um *body* de pedido SOAP tal como se apresenta no seguinte exemplo:

```
<soapenv:Body>
    <web:processaMensagemEX008A>
        <SubmeterDeclaracaoMensagem>
            <java:Mensagem>Mensagem EX008A em base64</java:Mensagem>
        </SubmeterDeclaracaoMensagem>
    </web:processaMensagemEX008A>
</soapenv:Body>
```

A este pedido o sistema responde da seguinte forma:

```
<env:Body>
    <m:processaMensagemXMLEX008AResponse >
        <return>
            <java:CodigoResultado>0</java:CodigoResultado>
            <java:Resultado>Mensagem submetida com sucesso<java:Resultado>
        </return>
    </m:processaMensagemXMLEX008AResponse>
</env:Body>
```

3.7. Exemplo de SOAP:Body – Método processaMensagemEX011A

Como resultado da aplicação do método **processaMensagemEX011A** será produzido um *body* de pedido SOAP tal como se apresenta no seguinte exemplo:

```
<soapenv:Body>
    <web:processaMensagemEX011A>
        <SubmeterDeclaracaoMensagem>
            <java:Mensagem>Mensagem EX011A em base64</java:Mensagem>
        </SubmeterDeclaracaoMensagem>
    </web:processaMensagemEX011A>
</soapenv:Body>
```

A este pedido o sistema responde da seguinte forma:

```
<env:Body>
    <m:processaMensagemXMLEX011AResponse >
        <return>
            <java:CodigoResultado>0</java:CodigoResultado>
            <java:Resultado>Mensagem submetida com sucesso<java:Resultado>
        </return>
    </m:processaMensagemXMLEX011AResponse>
</env:Body>
```

3.8. Exemplo de SOAP:Body – Método processaMensagemEX013A

Como resultado da aplicação do método **processaMensagemEX013A** será produzido um *body* de pedido SOAP tal como se apresenta no seguinte exemplo:

```
<soapenv:Body>
    <web:processaMensagemEX013A>
        <SubmeterDeclaracaoMensagem>
            <java:Mensagem>Mensagem EX013A em base64</java:Mensagem>
        </SubmeterDeclaracaoMensagem>
    </web:processaMensagemEX013A>
</soapenv:Body>
```

A este pedido o sistema responde da seguinte forma:

```
<env:Body>
    <m:processaMensagemXMLEX013AResponse >
        <return>
            <java:CodigoResultado>0</java:CodigoResultado>
            <java:Resultado>Mensagem submetida com sucesso<java:Resultado>
        </return>
    </m:processaMensagemXMLEX013AResponse>
</env:Body>
```

3.9. Código Resultado

A cada invocação, o sistema poderá responder com as mensagens abaixo descritas:

Código	Descrição
0	Mensagem submetida com sucesso/Mensagens encontradas.
1	Canal WebService inativo.
2	Erro de processamento.
3	Tipo de mensagem enviada desconhecida.
4	O utilizador não possui o perfil correto para utilizar o canal webservice.
5	Não foram encontrados resultados para a operação pretendida.
6	Utilizador inválido.
7	Documento não encontrado.
8	Modelo de documento inválido.

3.10. Tipos de Modelos de documentos admitidos

Na invocação do método **obterDocumentosPDF** requer a inserção do um argumento “modelo”.

Modelo	Descrição	Observações
2	T5N	Exemplar de Controlo T5N
3	Exemplar 3	Informação da Autorização de Saída ou Certificação de Saída para o Expedidor/Exportador (conforme fase da declaração)
4	Documento Acompanhamento	Informação da Autorização de Saída para Movimentos ECS
5	CCE	Informação sobre Certificados Comprovativos de Exportação
6	Exemplar 3 FN	Informação Certificação de Saída para o Fornecedor Nacional (sem dados do Destinatário)

4. Assinatura certificado SSL (CSR)

A invocação dos serviços web pressupõe um processo de autenticação mediante a validação da chave privada da aplicação, do conhecimento exclusivo da entidade aderente, sendo a respetiva chave pública comunicada e assinada pela AT. O certificado a ser utilizado na operação é assinado pela AT, a pedido da entidade aderente. Para este efeito, a entidade aderente deve efetuar um pedido de certificado (CSR – *Certificate Signing Request*).

O CSR é um pequeno ficheiro de texto cifrado que contém o certificado SSL e toda a informação necessária para que a AT possa assinar e devolver o certificado assinado digitalmente, para que possa ser utilizado no processo de autenticação na invocação do serviço web.

Os procedimentos para geração do CSR são simples, mas variam de acordo com a tecnologia web utilizada pela entidade aderente, razão pela qual devem ser consultados os respetivos manuais de apoio de cada ferramenta.

A informação que o CSR deve conter não pode ultrapassar os tamanhos máximos, conforme a descrição seguinte:

Campo CSR	Descrição	Tamanho Máximo
C = Country	O código ISO de 2 letras referente ao local da sede. Por exemplo, no caso de Portugal é “PT”.	2 (chars)
ST = Province, Region, County or State	Distrito da sede.	32 (chars)
L = Town/City	Local da sede.	32 (chars)
CN = Common Name	Neste campo deve ser indicado o número de identificação fiscal da entidade aderente.	9 (chars)
O = Business Name / Organisation	Designação legal da empresa.	180 (chars)
OU = Department Name /Organisational Unit	Departamento para contacto.	180 (chars)
E = An email address	O endereço de correio eletrónico para contacto, geralmente do responsável pela emissão do CSR ou do departamento de informática. Tem que ser um endereço de email válido.	80 (chars)
Key bit length	Chave pública do certificado SSL tem de ser gerada com 2048 bits.	2048 (bits)

A utilização de carateres especiais (e.g., portugueses, línguas latinas, etc.) não é aceite em nenhum dos campos acima indicados, uma vez que a utilização desses carateres vai invalidar a assinatura digital do certificado SSL.

Como resultado deste processo, a AT procederá à assinatura do certificado e remete em resposta ao pedido o certificado assinado para integração na chave privada da entidade aderente.

O certificado terá a validade de 12 meses a contar da data da assinatura e deve ser solicitado um novo no mínimo um mês antes da expiração do anterior.

4.1. Gerar um certificado SSL

Um certificado SSL é uma chave RSA composta por duas partes: chave privada e chave pública.
Como a chave privada deve ser apenas do conhecimento da entidade aderente, a emissão da mesma tem sempre de ser efetuada pelo próprio, em computador próprio, e nunca num *site* ou serviço web que encontre para o efeito.

Existem diversas ferramentas para geração de certificados SSL, proprietárias e Opensource. A AT utiliza a ferramenta OpenSSL, que é a ferramenta Opensource de referência, livre de custos de utilização.

Para gerar um certificado SSL, cada entidade aderente deve fazê-lo no seu próprio computador, utilizando o seguinte comando:

openssl req -new -subj "**/C=**PT**/ST=**Distrito da Sede**/L=**Local da Sede**/O=**Empresa **/OU=**Departamento de Informatica**/CN=**555555555**/emailAddress=**informatica@empresa.pt" **–newkey rsa:2048 -nodes -out** 555555555.csr **-keyout** 555555555.key

Cada entidade aderente deve substituir a informação específica no comando anterior pelos seus dados, uma vez que os apresentados são apenas exemplificativos, e não deve alterar a informação indicada a **Bold**.

Como resultado, do comando anterior será gerado o certificado SSL e serão produzidos dois ficheiros:

- 555555555.csr – Ficheiro com o pedido CSR a enviar à AT;
- 555555555.key – Ficheiro com a chave privada gerada.

4.2. Verificar conteúdo do CSR gerado

Antes de enviar o CSR para assinatura digital pela AT, pode e deve ser verificado o conteúdo do ficheiro para garantir que toda a informação está como pretendido. Para tal, deve ser usado o seguinte comando:

```
openssl req -text -noout -in 555555555.csr
```

Cada entidade aderente deve substituir os parâmetros que não estão a **Bold** pelos nomes dos ficheiros corretos.

4.3. Integrar certificado com a chave privada

Depois de receber o certificado SSL assinado pela chave digital da AT, é necessário integrar esse certificado com a chave privada gerada no passo anterior (555555555.key). Para tal, deve ser usado o seguinte comando:

```
openssl pkcs12 -export -in 555555555.crt -inkey 555555555.key –out 555555555.pfx
```

Cada entidade aderente deve substituir os parâmetros que não estão a **Bold** pelos nomes dos ficheiros corretos.

Como resultado, o certificado SSL assinado pela AT é integrado com a chave privada e gravada com uma *password* de acesso que cada entidade aderente deve definir na execução do comando.

5. Endereços Úteis

5.1. Página de produtores de software (certificados digitais)

https://www.portaldasfinancas.gov.pt/pt/external/factemipf/painelInicialProdSoftware.action

5.2. Gestão de subtilizadores no Portal das Finanças

https://www.acesso.gov.pt/gestaoDeUtilizadores/consulta?partID=PFIN

5.3. WSDL do envio de dados à AT por Webservice

Ambiente de testes/qualidade:

https://servicos.portaldasfinancas.gov.pt:808/CanalWS/STDEXP/StadaExpWS

5.4. Endereços para envio de dados à AT por Webservice

Ambiente de testes/qualidade:

https://servicos.portaldasfinancas.gov.pt:808/CanalWS/STDEXP/StadaExpWS

Ambiente de produção:

"A definir"