Manual Desarrollador Externo

Envío Automático Documentos Tributarios Electrónicos

OI2003_UPDTE_MDE_1.5

Oficina Internet
Subdirección Informática
Servicio Impuestos Internos

Fecha:31/10/2003

CONTROL DE VERSIONES

CONTROL DE VERSIONES			
Versión	Fecha	Autor	Revisor
1.0	27/11/2002	Zulema Olguín T.	Quentin Sherman
		??	
1.1	12/02/2003	Zulema Olguín T.	Quentin Sherman
Se agrego Capitulo 3 (Pruebas Envío)		??	
1.2	01/04/2003	Zulema Olguín T.	Quentin Sherman
Se reemplazo parámetro MSIE 5.5 por PROG 1.0, en Requerimiento de Request y en Anexo 3		?? El cambio en este parámetro permitirá que el formato de Salida sea en XML	
1.3		Zulema Olguín Traro	Nicolas Chelebifski
Se modificaron los Tipos de Status		?? Se modificaron mensajes de Status	
1.4		Zulema Olguín T.	Nicolas Chelebifski
Se agrego Anexo 4		?? Se detallan problemas frecuentes en uso del manual	
1.5	31/10/2003	?? Zulema Olguín Traro	
Se agregaron Status : 7,8,9, ver punto 2.1		?? Se agregaron Status 7,8,9	

INTRODUCCIÓN

Este documento está dirigido a quienes tengan la misión de implementar el envío automático de Documentos Tributarios Electrónicos(DTE).

La misión del programador, será simular las operaciones de autenticación, mediante certificado digital, y el proceso de upload que normalmente son realizadas por un browser.

En este manual se aborda el proceso de Upload.

Para poder utilizar este manual, es necesario tener previo conocimiento de Formato RFC1867, XML y Certificado Digital.

Nota : Para implementar el envío de DTE, mediante la Autenticación Automática, se recomienda ver: Manual Desarrollador "Ws Autenticación Automática con Certificado Digital (*OI2003_AUTAUTOM_MDE_1.2*).

CAPÍTULO 1

ANÁLISIS DEL SISTEMA

Esta aplicación, permite a los usuarios el enviar los de documentos tributarios Electrónicos , mediante upload.

Esta es una aplicación basada en los métodos browser programa a programa

1.1 Quienes pueden utilizar esta aplicación.

Esta aplicación puede ser utilizada por todas aquellas Empresas que requieran enviar archivos DTE, por Upload.

CAPÍTULO 2

VISIÓN GENERAL ENVÍO AUTOMÁTICO DE DTE

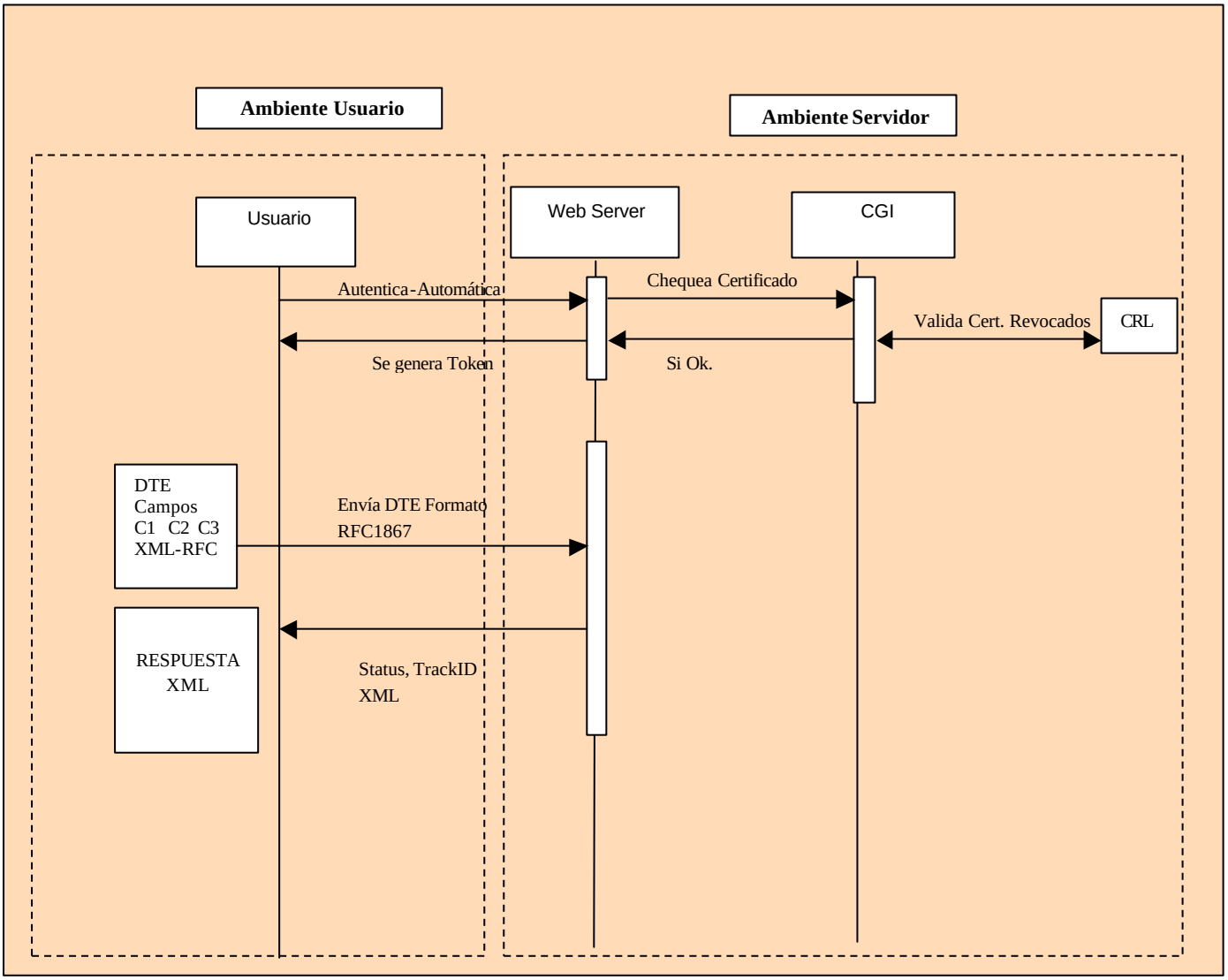

Figura 2.0

De acuerdo al diagrama superior (figura 2.0), para que un usuario pueda enviar en forma automática los archivos DTE, lo primero que debe hacer es autenticarse, mediante el uso de la Autenticación Automática.

Tal como se mencionó anteriormente, esta es una aplicación basada en los métodos browser programa a programa, por lo que se debe simular la autenticación, tal como si se tratara de una autenticación por browser.

La autenticación valida el certificado y chequea que no esté revocado.

Si los datos del certificado no son válidos, el sistema rechaza la autenticación.

Si los datos del certificado son válidos, el Web Server genera en forma automática una cookie llamada TOKEN.

Un Token es un identificador único el cual es almacenado en el Header del **Response**, y permite la búsqueda de toda la información relacionada a una sesión del usuario.

Ver Ejemplo Token, ANEXO 1

Cuando la aplicación Cliente recibe los datos del **Response**, el usuario puede enviar su archivo por Upload.

Normalmente el envío de archivos mediante Upload se realiza a través del browser, sin embargo como este sistema se basa en el método programa a programa, la parte browser es reemplazada por una aplicación Cliente que simula el tradicional proceso de upload, para lo cual se debe conectar hacia el sitio del SII mediante Socket.

El archivo que se enviará por upload debe tener formato XML, cuyos campos deben ser validados contra un archivo Schema entregado por el SII (EnvioDTE.xsd) y debe cumplir con las normas descritas en los estándares RFC-1867 (The Requests for Comments).

Ver ejemplo: Formato Archivo para envío por Upload ANEXO 2

El programa Cliente al momento de hacer upload al archivo DTE (en formato XML), agrega los datos del **Response**, incluido el Token en el Header del archivo y lo envía hacia el sitio del SII.

Ver ejemplo : Archivo Upload generado por Cliente con su Header ANEXO 3

Una vez recibido el archivo DTE, nuestro sistema, genera una salida en formato XML, que indica el status de la recepción del archivo.

2.1 Los tipos de Status son:

✍✍0 = Upload OK
✍✍1 = El Sender no tiene permiso para enviar
✍✍2 = Error en tamaño del archivo (muy grande o muy chico)
✍✍3 = Archivo cortado (tamaño <> al parámetro size)
✍✍5 = No está autenticado
✍✍6 = Empresa no autorizada a enviar archivos
✍✍7= Esquema Invalido
✍✍8= Firma del Documento
✍✍9= Sistema Bloqueado
✍✍Otro = Error Interno.

2.2 Formato de Salida

Como se mencionó anteriormente, una vez que el SII, ha recepcionado el archivo DTE, nuestro sistema genera una respuesta en formato XML, que indica el status de la recepción del archivo recibido.

Los Parámetros de Salida son:

✍✍ Rut y Dv de quien envía ✍✍ Rut y Dv de la Compañía ✍✍ Nombre archivo DTE enviado ✍✍ Fecha, Hora, min., seg, de recepción. ✍✍ Estado de la recepción del DTE ✍✍ Indica Número de Atención	<RUTSENDER>**1-9**</RUTSENDER> <RUTCOMPANY>**3-5**</RUTCOMPANY> <FILE>**EnvioEjemplo.xml**</FILE> <TIMESTAMP>**2002-11-25 18:51:44**</TIMESTAMP> <STATUS>**0**</STATUS> <TRACKID>**39**</TRACKID>

Ejemplo Formato de Salida, Status 0 (Recepción OK)

```
<?xml version="1.0" ?>
- <RECEPCIONDTE>
    <RUTSENDER>1-9</RUTSENDER>
    <RUTCOMPANY>3-5</RUTCOMPANY>
    <FILE>EnvioEjemplo.xml</FILE>
    <TIMESTAMP>2002-11-25 18:51:44</TIMESTAMP>
    <STATUS>0</STATUS>
    <TRACKID>39</TRACKID>
  </RECEPCIONDTE>
```

Figura 2.1

Nota: De acuerdo a la figura 2.1 ,si la recepción del archivo DTE, tiene status 0, se le asigna un **TrackID**, que corresponde al número de atención o identificador del envío.

```
<TRACKID>39</TRACKID>
```

Ejemplo Formato de Salida, Status 5 (Error Interno (WRT).

```
<?xml version="1.0" ?>
- <RECEPCIONDTE>
    <RUTSENDER>1-9</RUTSENDER>
    <RUTCOMPANY >3-5</RUTCOMPANY >
    <FILE>EnvioEjemplo.xml</FILE>
    <TIMESTAMP>2002-11-25 18:51:44</TIMESTAMP >
    <STATUS>5</STATUS>
  </RECEPCIONDTE>
```

Figura 2.2

Ejemplo Formato de Salida, Status 7 (Eschema invalido).

```
<RECEPCIONDTE>
 <RUTSENDER>07880442-4</RUTSENDER>
 <RUTCOMPANY>88888888-8</RUTCOMPANY>
 <FILE>ENVFIN_100_sign.xml</FILE>
 <TIMESTAMP>2003-10-31 10:04:26</TIMESTAMP>
 <STATUS>7</STATUS>
  <DETAIL>
    <ERROR>LSX-00265: attribute "version" value "3.2" is wrong (must be ".2")
   </ERROR>
    <ERROR>LSX-00213: only 0 occurrences of particle "sequence", minimum is 1</ERROR>
  </DETAIL>
</RECEPCIONDTE>
```

Figura 2.3

2. REQUERIMIENTOS DEL REQUEST

Como se mencionó anteriormente, el programa Cliente, debe incluir en el header del request lo siguiente:

```
POST /cgi_dte/UPL/DTEUpload HTTP/1.0^M
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-powerpoint, application/ms-excel,
application/msword, */*^M
Referer: {URL que referencia a upload Ej.
http://empresaabc.cl/test.html}^M
Accept-Language: es-cl^M
Content-Type: multipart/form-data: boundary={boundary data: Ej. ------
----------------------7d23e2a11301c4}^M
Accept-Encoding: gzip, deflate^M
User-Agent: Mozilla/4.0 (compatible; PROG 1.0; Windows NT 5.0; YComp
5.0.2.4)^M
Host: {Host Id. Ej: https://maullin.sii.cl}^M
Content-Length: {largo total de mensaje sin Req. Header. Ej.: 10240}^M
Connection: Keep-Alive^M
Cache-Control: no-cache^M
Cookie: TOKEN={Entregado por Autenticación. Ej.: YZD0II2ApZjlM}^M
^M
```

Figura 2.3

Nota: El parámetro (**PROG 1.0**), encerrado en el circulo, ver Figura 2.3, permitirá que el formato de salida sea en un XML

CAPÍTULO 3
PRUEBAS DE ENVIO

Para realizar un UPLOAD en forma automática, el programa cliente debe simular un Browser, es decir, el programa debe realizar la comunicación tipo Socket de TCP/IP con encriptación de datos SSL. Una vez conectado al sitio, el programa debe enviar los datos tal como se muestra en ANEXO 3, para lo cuál, debe reemplazar los campos con sus datos correspondientes. Los siguientes son los datos para realizar la conexión:

1. Host : maullin.sii.cl
2. Puerto : 443
3. Tipo de encriptación : SSL

ANEXO 1

1.- Ejemplo de TOKEN

```
TOKEN=gd43dh6sfE34Kd3
```

Figura 2.4

ANEXO 2

1.- Formato Archivo para envío por Upload (XML)

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<EnvioDTE xmlns="http://www.sii.cl/SiiDte"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.sii.cl/SiiDte EnvioDTE.xsd"
          version="1.0">
<SetDTE ID="SetDoc">
……
</EnvioDTE>
```

Figura 2.5

ANEXO 3

1.- Ejemplo Archivo generados por Cliente Con su respectivo Header

Nota: Lo destacado con rojo corresponde al Header.

```
POST /cgi_dte/UPL/DTEUpload HTTP/1.0^M
Accept: image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
application/vnd.ms-powerpoint, application/ms-excel,
application/msword, */*^M
Referer: {URL que referencia a upload Ej.
http://empresaabc.cl/test.html}^M
Accept-Language: es-cl^M
Content-Type: multipart/form-data: boundary={boundary data: Ej. ------
----------------------7d23e2a11301c4}^M
Accept-Encoding: gzip, deflate^M
User-Agent: Mozilla/4.0 (compatible; PROG 1.0; Windows NT 5.0; YComp
5.0.2.4)^M
Host: {Host Id. Ej: https://maullin.sii.cl}^M
Content-Length: {largo total de mensaje sin Req. Header. Ej.: 10240}^M
Connection: Keep-Alive^M
Cache-Control: no-cache^M
Cookie: TOKEN={Entregado por Autenticación. Ej.: YZD0II2ApZjlM}^M
^M
{Comienzo de Multipart/Form-data}
----------------------------7d23e2a11301c4^M
Content-Disposition: form-data; name="rutSender"^M
^M
1^M
----------------------------7d23e2a11301c4^M
Content-Disposition: form-data; name="dvSender"^M
^M
9^M
----------------------------7d23e2a11301c4^M
Content-Disposition: form-data; name="rutCompany"^M
^M
3^M
----------------------------7d23e2a11301c4^M
Content-Disposition: form-data; name="dvCompany"^M
^M
5^M
----------------------------7d23e2a11301c4^M
Content-Disposition: form-data; name="archivo"; filename="EnvioEjemplo.xml"^M
Content-Type: text/xml^M
^M
<?xml version="1.0" encoding="ISO-8859-1"?>^M
<EnvioDTE xmlns="http://www.sii.cl/SiiDte" ^M
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ^M
          xsi:schemaLocation="http://www.sii.cl/SiiDte EnvioDTE.xsd" ^M
          version="1.0">^M
<SetDTE ID="SetDoc">^M
……..
</EnvioDTE>^M
^M
----------------------------7d23e2a11301c4--^M
```

Figura 2.6

ANEXO 4

Problemas Frecuentes Detectados en la Implementación del Envío.

Algunos de los problemas frecuentes detectados en el envió se relacionan con:

Campo	Problema	Solución

boundary	Diferencia en la composición del **boundary** (Header y Cuerpo del Mensaje)	El boundary del "**Cuerpo del Mensaje"**, debe llevar **dos caracteres "más"** que los caracteres del boundary **del Header**.
		
		Además el boundary, del **Final de mensaje**, debe llevar dos guiones al comienzo y dos guiones al final, antes del control M. Ver Ejemplos.
		
		1-Ejemplo boundary: **Header**
		
		`Content-Type: multipart/form-data: boundary=7d23e2a11301c4^M`
		
		2-Ejemplo boundary: **Cuerpo del mensaje,** debe tener 2 guiones al inicio.
		
		`--7d23e2a11301c4^M`
		
		3-Ejemplo boundary**: Final de mensaje**, debe llevar dos guiones del comienzo más dos adicionales antes del control M.
		`--7d23e2a11301c4----^M`

Largo del Mensaje	El largo del mensaje incluye datos del Header	El largo del mensaje no debe incluir datos del Header, tal como se muestra en la figura 2.6 el Header corresponde a los datos encerrado en el recuadro, la línea ***{Comienzo de Multipart/Form-data},*** corresponde a una línea en blanco, por lo que el inicio del mensaje comenzaría en el primer carácter o guión después de ***{Comienzo de Multipart/Form-data}*** Ejemplo: ***{Comienzo de Multipart/Form-data}*** `--7d23e2a11301c4^M` **Donde el circulo indica el inicio del mensaje**.
Host	Uso del campo Host en programas C y que además están utilizando la librería de OpenSSL	Los programas en lenguaje C, los cuales además utilizan la librería OpenSSL, "**NO deben incluir el Campo Host".**
`Control M (^M)`	Falta el Control M	-Los caracteres Control M **(^M**), forman parte del archivo, por lo que estos deben ser enviados. -La forma de parsear los Control M **(^M),** depende del de Cliente: ?? Si el Cliente es un programa en C++ y utiliza el **objeto string** para la concatenación, se debe indicar cada fin de línea con "\r\n", para parsear el **^M.** ?? Si el Cliente es un programa en C o C++ y utiliza **char**, se debe indicar cada fin de línea con "\n".