Streamlined Sales Tax Project Implementation Guide

September 23, 2005

Web Service Implementation chapter
V1.0.1

Version Table

Version of Changes

Version	Date	Comments
1.0	August 1, 2004	Initial Draft
1.1	October 1, 2004	Modifications from the Portland meeting
4.0	December 1, 2004	Modifications from the Miami meeting
5.0	February 3, 2005	Modifications from the Phoenix meeting
6.0	March 15, 2005	Modifications from the Nashville meeting
7.0	April 1, 2005	Modifications from the Austin meeting
8.0	April 24,2005	Modifications from the Atlanta meeting.
9.0	June 1, 2005	Modifications from review o Remove FTP as a transmission option
9.1	July 29, 2005	Added Chapter 13 – Web Service Implementation
9.2	Aug 15, 2005	Revised security token, minor message naming changes, and documentation improvements.
9.3	Sept 23, 2005	Added TransmitterId param to Ack and LastAck operations. Added Message param to Ping operation.

Chapter 13 – Web Service Implementation

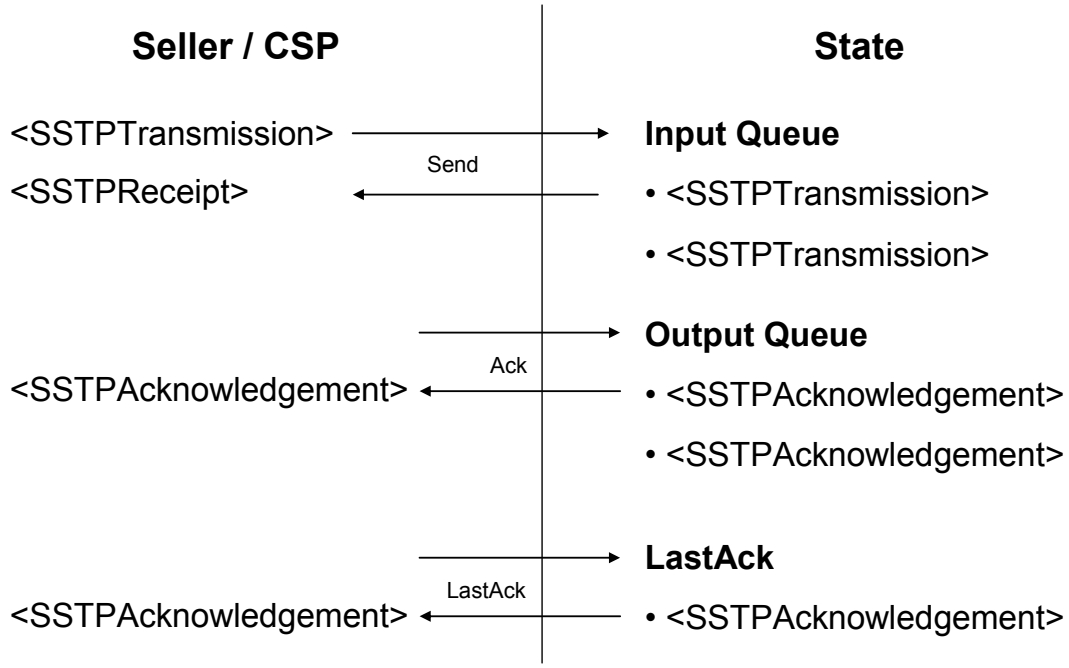
Overview

This chapter is an implementation guide for states that choose to implement a web service for receiving SERs, Payments, and Information Reports. CSPs or sellers may then use or implement a web service client to communicate with the state's web service and transmit the data.

The **efile** web service provides four operations:

> **Ping** – For testing the connection and authentication. This can also be used by a monitoring service to verify the system is operational.
>
> **Send** – Sends a <SSTPTransmission> document and returns a <SSTPReceipt>. The document is queued for processing. When processed, the <SSTPAcknowledgement> document is queued for download by the Ack operation.
>
> **Ack** – Requests the next <SSTPAcknowledgement> document. This can be called until there are no more documents available..
>
> **LastAck** – Requests the previous <SSTPAcknowledgement> document. This is used in the case of an error receiving the previous document using the Ack operation.

The following is an example of the operation flow:



The efile web service has been designed to be easy to implement for both the state and CSPs or Sellers. It adheres to all web service interoperability (WS-I) standards. Sample implementations of both the service and client are available for the Microsoft .NET and Java platforms. The EFileService.wsdl (Web Services Description Language) document can be used alone to implement a service or client on any platform that supports SOAP 1.1 web services.

Because of the complexity of the schemas and the batch processing nature of the service, the SSTP schemas are not directly linked (imported) into the WSDL. Keeping the WSDL and schemas separate allows for future changes to the schemas without affecting the web service interface. The Send operation accepts an arbitrary (any) XML element and the Ack and LastAck operations return any XML Element. It is up to the client and service to perform schema validation. States are expected to use the <TransmissionHeader transmissionVersion="2005V02"> transmissionVersion value to detect and handle current and future versions of the schemas.

Namespaces

The EFile service WSDL uses the following namespace to identify its custom elements: http://streamlinedsalestax.org/efile. Note that namespace names are case sensitive. The efile namespace is all lowercase.

The SSTP schemas do not define a namespace. Therefore, a xmlns="" is required on all SSTP schema elements. For example:

```
<Send xmlns="http://streamlinedsalestax.org/efile">
  <Transmission>
    <SSTPTransmission xmlns="">...</SSTPTransmission>
  </Transmission>
</Send>
```

Security

Security is a critical implemention issue and there are many available options. It is necessary to balance the security needs with the capabilities of the various implementation platforms, tools and personnel. The EFile service will use a combination of SSL (HTTPS protocol) and a Username/Password security token for security and authentication. This is equivalent to the security used for the web site based, document upload and download implementation alternative.

SSTP will use a standard WS-Security token for authentication. This is designed to provide interoperability with existing and future software and hardware security systems. A valid WS-Security <Security> SOAP header element must be included in all requests. For convenience, this is explicitly defined in the WSDL.

SSTP will use plain text passwords since the entire transmission is being encrypted using SSL. Therefore, it is required to specify the <Password> Type attribute as: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText. An example is shown below.

```
<soap:Header>
    <Security soap:mustUnderstand="1" xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-secext-1.0.xsd">
      <UsernameToken>
        <Username>CSP0000001</Username>
        <Password Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-
profile-1.0#PasswordText">password</Password>
      </UsernameToken>
    </Security>
</soap:Header>
```

The <Username> should contain the transmitter id. The <Password> should contain the transmitter password as registered with the SSTP registration service.

See also: http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf

SOAP Details

The SSTP EFile web service uses the document/literal wrapped SOAP binding style. This means that each operation contains a single "wrapped" parameter element that can be validated by a schema. This style is WS-I compliant and supported by the major web service platforms. It is well suited to the document tranmission/acknowledgement nature of this web service and is actually necessary for this particular interface to make it WS-I compliant. See the Send operation details below for an example. Refer to http://www-128.ibm.com/developerworks/webservices/library/ws-whichwsdl/ for more details on this topic.

Operations

Ping

The Ping operation may used by the client to verify that the web service is operational. However, it should not be called needlessly by clients. States can also use Ping along with a monitoring service to monitor system availablility. It returns an information text string indicating the name and version of the web service. The content of this is not currently standardized. It should not be used for determination of interface or schema versions.

The Message parameter is for states' private use. It is useful within the context of a monitoring service to perform a system status check based on a private parameter value.

Note: It is recommended, but not required for states to implement authentication for the Ping method. However, clients should always include the <Security> token.

Soap Request

```
POST /efile/EFileService.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://streamlinedsalestax.org/efile/Ping"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <UsernameToken>
        <Username>string</Username>
        <Password Type="string">string</Password>
      </UsernameToken>
    </Security>
  </soap:Header>
  <soap:Body>
    <Ping xmlns="http://streamlinedsalestax.org/efile">
      <Message>string</Message>
    </Ping>
  </soap:Body>
</soap:Envelope>
```

Soap Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <PingResponse xmlns="http://streamlinedsalestax.org/efile">
      <PingResult>string</PingResult>
    </PingResponse>
  </soap:Body>
</soap:Envelope>
```

Send

The Send operation is used to transmit an SER, Information Report, Payment or other transmission to the state. The transmitter should be authenticated and the document and the document should be queued for processing. Document content errors should be handled in the acknowledgement. An <SSTPReceipt> is returned in the response.

Soap Request

```
POST /efile/EFileService.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://streamlinedsalestax.org/efile/Send"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <UsernameToken>
        <Username>string</Username>
        <Password Type="string">string</Password>
      </UsernameToken>
    </Security>
  </soap:Header>
  <soap:Body>
    <Send xmlns="http://streamlinedsalestax.org/efile">
      <Transmission>
        <SSTPTransmission xmlns="">...</SSTPTransmission>
      </Transmission>
    </Send>
  </soap:Body>
</soap:Envelope>
```

Soap Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <SendResponse xmlns="http://streamlinedsalestax.org/efile">
      <Receipt>
        <SSTPReceipt xmlns="">...</SSTPReceipt>
```

```
    </Receipt>
   </SendResponse>
  </soap:Body>
</soap:Envelope>
```

Note: An <SSTPReceipt> element is in the process of being added to the SSTP schemas.

Ack

The Ack operation requests the next <SSTPAcknowledgement>, if available. States are expected to implement a queueing mechanism for acknowledgements. Ack will be called repetitively to retrieve all available acknowledgements. When no acknowledgements are available, nothing is returned. Once a document has been retrieved, the service should place it in a "LastAck" holding area for the LastAck operation. If there is an existing "LastAck" document, it can be discarded.

States have up to 72 hours to process a transmission. It is recommended that clients do not check for acknowledgements more than four times (opinions?) a day?

A TransmitterId has been added to enable state implementions that are unable to access the SOAP security header. Clients are still required to provide the security header, but should also provide the TransmitterId parameter.

Note: Acknowledgements are not guarenteed to be returned in the same order as the original transmission. The acknowledgement should be matched with the corresponding transmission using the <TransmissionId>.

Soap Request

```
POST /efile/EFileService.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://streamlinedsalestax.org/efile/Ack"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <UsernameToken>
        <Username>string</Username>
        <Password Type="string">string</Password>
      </UsernameToken>
    </Security>
  </soap:Header>
  <soap:Body>
    <Ack xmlns="http://streamlinedsalestax.org/efile">
      <TransmitterId>string</TransmitterId>
    </Ack>
  </soap:Body>
</soap:Envelope>
```

Soap Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
```

```
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <AckResponse xmlns="http://streamlinedsalestax.org/efile">
      <Acknowledgement>
        <SSTPAcknowledgement xmlns="">...</SSTPAcknowledgement>
      </Acknowledgement>
    </AckResponse>
  </soap:Body>
</soap:Envelope>
```

LastAck

The LastAck operation requests the previous <SSTPAcknowledgement> document, if available. This is a recovery mechanism in case the previous Ack download failed. If no documents are available, nothing is returned. The "LastAck" document remains in the holding area until the next Ack operation pushes it out.

A TransmitterId has been added to enable state implementions that are unable to access the SOAP security header. Clients are still required to provide the security header, but should also provide the TransmitterId parameter.

Soap Request

```
POST /efile/EFileService.asmx HTTP/1.1
Host: localhost
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://streamlinedsalestax.org/efile/LastAck"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <Security xmlns="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">
      <UsernameToken>
        <Username>string</Username>
        <Password Type="string">string</Password>
      </UsernameToken>
    </Security>
  </soap:Header>
  <soap:Body>
    < LastAck xmlns="http://streamlinedsalestax.org/efile">
        <TransmitterId>string</TransmitterId>
    </LastAck>
  </soap:Body>
</soap:Envelope>
```

Soap Response

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
```

```
    <LastAckResponse xmlns="http://streamlinedsalestax.org/efile">
      <Acknowledgement>
        <SSTPAcknowledgement xmlns="">...</SSTPAcknowledgement>
      </Acknowledgement>
    </LastAckResponse>
  </soap:Body>
</soap:Envelope>
```

Error Handling

In general, document errors should be reported using the <SSTPAcknowledgement> <Errors> collections from the Ack or LastAck operations. SOAP faults should only be generated for errors that can't be handled otherwise. One example, would be an authentication fault.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:Client</faultcode>
      <faultstring>Unable to validate the security token.</faultstring>
      <faultactor>EFileService</faultactor>
      <detail>
        <string>The Username / Password is not valid.</string>
      </detail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

Note: <detail> content should be placed within a <string> child element (as shown above) for interoperability with some SOAP clients that expect an element.

Other conditions that warrant a SOAP fault include:

- Unknown SOAP operation
- Missing or invalid Security header
- Unknown TransmitterId
- Send document is not well formed or significantly violates the schema such that a receipt cannot be generated. For example, no TransmissionId

WSDL

The current EFileService.wsdl is included with the schema set.

Samples

The following service and client sample implementations are provided as examples. These samples correctly implement the interface defined by the WSDL. They are only minimally functional, but should serve as an implementation template or reference. They can be deployed as a web service and come with basic unit tests. They are interoperable, meaning that .Net can call Java and vice versa.

Each zip package includes a readme.txt file with additional details.

Microsoft .NET

efile4cs-1.0.x.zip – contains a sample C# efile web service, client and unit tests which run under the Microsoft .NET 1.1 framework and IIS 5 or 6.

Java

efile4j-service-1.0.x.zip – contains a sample Java efile web service which runs under Java 1.4 and Apache Axis 1.2
efile4j-client-1.0.x.zip – contains a sample Java efile client and unit tests which run under Java 1.4