



DuoShare Web Services Getting Started Guide

Document Date: June 26, 2007

DuoShare Web Services Getting Started Guide

Copyright Notice

Copyright © 2007 DuoShare, Inc. All rights reserved.

Documentation version 1.1.

No part of this publication may be reproduced, transmitted, transcribed, stored in a retrieval system, or translated into any language or computer language, in any form or by any means, electronic mechanical, magnetic, optical, chemical, manual, or otherwise, without the prior written permission of DuoShare, Inc.

Trademarks

W3C[®] is a trademark (registered in numerous countries) of the World Wide Web Consortium; marks of W3C are registered and held by its host institutions MIT, ERCIM, and Keio.

Contact

Registration: <http://duoshare.com/registration.html>

Sales: sales@duoshare.com

Support: support@duoshare.com

Phone: 214-691-4884

Table of Contents

Introduction.....	1
Purpose.....	1
Scope.....	1
Typographical Conventions.....	1
Definitions, Acronyms, and Abbreviations.....	1
What You Need to Know.....	2
References.....	2
Getting Started.....	3
Overview.....	3
Communicating with DuoShare.....	3
Sending a Request.....	3
Receiving a Response.....	4
Checking for Errors.....	6
Where To Go From Here.....	7
Appendix A: Web Service Descriptions.....	8
CustaccessEF.....	8
JobcontrolEF.....	8
ListdetailEF.....	8
ListEF.....	8
PostML.....	8
UseraccessEF.....	8
UserEF.....	8

Introduction

Purpose

The purpose of this document is to help developers get started integrating with DuoShare Address Quality Services using web services.

Scope

This guide will help you create a very basic web services client that can communicate with DuoShare. It is beyond the scope of this document to cover XML, SOAP, or language specific implementations of DuoShare web services.

Our WSDL documents are annotated with human readable API documentation, you can find a listing of our WSDL documents in “Appendix A: Web Service Descriptions.”

Typographical Conventions

Different fonts and styles are used throughout this guide to allow certain content to be easily identified.

Examples are indented with numbered lines:

1 A code/protocol example (CR) (LF)

2 A `[variable]` in the example (CR) (LF)

Other conventions include:

- A `variable` referenced from the example above
- A literal *Carriage Return* (CR) and/or *Line Feed* (LF)
- *Italics* are used to denote emphasis on terms that can be found in the index and to reference user-interface elements for some examples.

Definitions, Acronyms, and Abbreviations

Bad Address: A bad address is not an address that is being punished. A bad address could best be described as either an incomplete, incorrect, or an unmatched address.

Delivery Point Validation (DPV)TM: The USPS algorithm used to verify an exact address.

Extensible Markup Language (XML): XML is markup language designed to describe data in a way that both humans and computers can understand. XML has become a standard language for describing interoperable data. See “References” for more information.

Hint data: Hint data can be returned with a bad address when using the PostML web service. See PostML in “Appendix A: Web Service Descriptions.”

HTTPS: HTTPS is not a separate protocol from HTTP. It uses HTTP over an encrypted SSL connection or TLS.

Hypertext Transfer Protocol (HTTP): HTTP is a stateless protocol which defines how messages are formatted,

transmitted, and handled for the World Wide Web. The HTTP protocol is detailed in RFC 2616 (see w3c).

Integrated Development Environment (IDE): Many developers use IDE's to increase their programming productivity.

Organization for the Advancement of Structured Information Standards (OASIS): This is a standards body that has defined several standards used in conjunction with web services.

Secure Sockets Layer (SSL) and Transport Layer Security (TLS): SSL and TLS provide secure communications on the Internet.

Simple Object Access Protocol (SOAP): SOAP is an XML based communication protocol used for sending messages between programs over the Internet. SOAP has numerous benefits including the ability to use ports open on most firewalls, simplicity and extensibility, and platform and language independence. See "References" for more information.

Username Token: The Username Token is a header in your SOAP request used for authentication. This is defined in the OASIS Web Services Security standard, see references.

Web Services Description Language (WSDL): WSDL is an XML document used to describe and locate web services. WSDL is not yet a w3c standard.

What You Need to Know

You will find integrating with DuoShare much easier if you have an intermediate understanding of XML, SOAP, and HTTP.

Sometimes web service integration can be made more difficult by using your IDE's tools if you do not understand the standards the tools implement for you. Though an intermediate understanding of the above is by no means a requirement, you will find integrating much easier if you understand how to:

- Create and edit XML documents both by hand and programatically
- Get a trace of HTTP request and response headers in your implementation language
- Read and write HTTP headers
- Read and write SOAP headers
- Get a trace of SOAP request and response messages in your implementation
- Supply a Username Token in SOAP request messages

If you are not familiar with all of the items in the above list, you should take a moment to get a basic understanding of each item. They will become invaluable skills in the debugging process, especially when you want to see exactly what your SOAP messages contain.

References

XML: <http://www.w3schools.com/xml/>

SOAP: <http://www.w3schools.com/soap/>

WSDL: <http://www.w3schools.com/wsd/>

HTTP: <http://www.w3.org/Protocols/>

SSL: <http://wp.netscape.com/eng/ssl3/draft302.txt>

Username Token: <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0.pdf>

OASIS: <http://www.oasis-open.org/>

Getting Started

Overview

In order to connect to any DuoShare web service you will need to do the following:

1. Download the WSDL file for the DuoShare web service you want to use.
2. Review your implementation language's API documentation to see how best to generate your web services proxy class. If you have a dynamic solution available to you with your implementation, a dynamic implementation is preferred over a static one.
3. Configure your HTTP request to transport over SSL and trust the DuoShare server's SSL certificate.

You should not accept a certificate with an untrusted root. When you see that the certificate has a trusted root certificate authority you should ensure that the certificate's subject is *CN=duoshare.com, OU=Corporate, O="DuoShare, Inc", L=Dallas, S=Texas, C=US*.

If you have any errors from an untrusted certificate or the certificate's subject does not match what is provided here when communicating with DuoShare, you should call us immediately.

4. If you are using a web service proxy class or generated web service code, you may need to configure the proxy or generated class to use Username Tokens.

The Username Token must be in *plain text* format. If you send a Digest, the server will not understand your request.

In short, all DuoShare web services can be described as: SOAP over HTTPS using a plain text Username Token.

Communicating with DuoShare

This section covers the basic communication between a web services client and DuoShare.

Sending a Request

A request will need to have a SOAP Header that contains a plain text UsernameToken and a SOAP Body. The following example shows a request for PostMLs validateInteractive operation. The SOAP Body consists of the operation name and any required parameters.

A sample request looks similar to the example shown below (null fields have been omitted from the SOAP Body for clarity):

```
<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"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/08/addressing"
xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"
xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
<soap:Header>
  <wsa:Action></wsa:Action>
  <wsa:MessageID>urn:uuid:99b94ca8-3298-4b28-b59d-485e2bbbfa5</wsa:MessageID>
  <wsa:ReplyTo>
    <wsa:Address>
      http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous
    </wsa:Address>
  </wsa:ReplyTo>
  <wsa:To>https://duoshare.com/dsWS/services/PostML</wsa:To>
  <wsse:Security soap:mustUnderstand="1">
    <wsse:UsernameToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="SecurityToken-b753bb0f-b8db-4633-a94f-3aa507bcd409">
      <wsse:Username> [username] </wsse:Username>
      <wsse:Password
        Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-username-token-profile-1.0#PasswordText">
        [password]
      </wsse:Password>
    </wsse:UsernameToken>
  </wsse:Security>
</soap:Header>
<soap:Body>
  <validateInteractive xmlns="http://ejbf.ds.com">
    <rec xmlns="">
      <error>false</error>
      <firmorrecipientinput>DuoShare, Inc.</firmorrecipientinput>
      <deliveryaddressline1input>10401 Miller Rd</deliveryaddressline1input>
      <deliveryaddressline2input>Suite 150</deliveryaddressline2input>
      <lastlineinput>Dallas, TX 75238</lastlineinput>
      <customerpk> [customerpk] </customerpk>
    </rec>
  </validateInteractive>
</soap:Body>
</soap:Envelope>

```

Receiving a Response

The reply from DuoShare will come as soon as the server is done processing the request. This example will

show a response to the above request (null fields have been omitted from the SOAP Body for clarity):

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance">
  <soapenv:Header />
  <soapenv:Body>
    <p75:validateInteractiveResponse xmlns:p75="http://ejbf.ds.com">
      <validateInteractiveReturn>
        <allerrormessages>
          Address Primary Number is found
          Address Secondary Number is found
          Address Matched
        </allerrormessages>
        <errorlevel>0</errorlevel>
        <customerpk>[customerpk]</customerpk>
        <listdetailpk>[listdetailpk]</listdetailpk>
        <listpk>[listpk]</listpk>
        <dpvconfirmind>Y</dpvconfirmind>
        <deliveryaddressline1Answer>
          10401 MILLER RD STE 150
        </deliveryaddressline1Answer>
        <lastlineanswer>DALLAS TX 75238-1238</lastlineanswer>
        <citynameanswer>DALLAS</citynameanswer>
        <statecodeanswer>TX</statecodeanswer>
        <zipcodeanswer>75238</zipcodeanswer>
        <zipplus4Answer>1238</zipplus4Answer>
        <deliveryaddressline1Input>
          10401 Miller Rd
        </deliveryaddressline1Input>
        <deliveryaddressline2Input>Suite 150</deliveryaddressline2Input>
        <lastlineinput>DALLAS,</lastlineinput>
        <originaldeliveryline>10401 Miller Rd</originaldeliveryline>
        <originaldeliveryline2>Suite 150</originaldeliveryline2>
        <originallastline>Dallas, TX 75238</originallastline>
        <originalsecondarynumber>150</originalsecondarynumber>
        <originalzipcode>75238</originalzipcode>
        <dpvcmraind>N</dpvcmraind>
        <zipsector>12</zipsector>
        <addressprimarynumberanswermsg>
          Address Primary Number is found
        </addressprimarynumberanswermsg>
        <countynum>113</countynum>
        <addressmsg>Address Matched</addressmsg>
        <addressprimaryoddevencode>0</addressprimaryoddevencode>
      </validateInteractiveReturn>
    </p75:validateInteractiveResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

```

<deliveryline1Shortanswer>
    10401 MILLER RD STE 150
</deliveryline1Shortanswer>
<addresssecondaryabbranswer>STE</addresssecondaryabbranswer>
<addressprimarynumberanswer>10401</addressprimarynumberanswer>
<streetsuffixprimaryname>ROAD</streetsuffixprimaryname>
<zipsegment>38</zipsegment>
<addresssecondaryoddevencode>E</addresssecondaryoddevencode>
<addresssecondarynumberanswermsg>
    Address Secondary Number is found
</addresssecondarynumberanswermsg>
<addresssecondarynumberanswer>150</addresssecondarynumberanswer>
<preferredlastlinecitystateid>W23428</preferredlastlinecitystateid>
<deliverypointcheckdigitanswer>4</deliverypointcheckdigitanswer>
<congressionaldistrictnum>03</congressionaldistrictnum>
<elotsequencenumberanswer>0031</elotsequencenumberanswer>
<streetsuffix>RD</streetsuffix>
<deliveryline2Shortanswer />
<dpvfootnote2>BB</dpvfootnote2>
<dpvfootnote1>AA</dpvfootnote1>
<citynameshortanswer>DALLAS</citynameshortanswer>
<streetsuffixabbranswer>RD</streetsuffixabbranswer>
<approvedabbrsecunit>STE</approvedabbrsecunit>
<financenum>482270</financenum>
<citymailingname>DALLAS</citymailingname>
<countyname>DALLAS</countyname>
<standerdsecunit>SUITE</standerdsecunit>
<recordtype>H</recordtype>
<streetnameanswer>MILLER</streetnameanswer>
<dpcanswer>25</dpcanswer>
<carrierouteidanswer>C026</carrierouteidanswer>
<originalprimarynumber>10401</originalprimarynumber>
<elotascdesanswer>A</elotascdesanswer>
</validateInteractiveReturn>
</p75:validateInteractiveResponse>
</soapenv:Body>
</soapenv:Envelope>

```

The response is neatly presented in the SOAP Body of the response. With this response you can see that the user inputs sent in the request are returned along with the answers.

Checking for Errors

Every record class you receive from DuoShare as a response will have a few common fields, one of which is the error field. You should always check if error is true. When true, you can then check all of the specific error messages.

Unless otherwise specified in the API documentation, any fields ending with the suffix “msg” will contain an error message corresponding to the field of the same name, but without the suffix.

For example, if there is an error with the `statecodeinput` field, then the `statecodeinputmsg` field will contain the error description.

Also, any fields with the suffix “desc” contain a human readable description for the field of the same name, but without the suffix. For example, if in a `JobcontrolRecord` the `statusid` field is set to “4402”, the `statusiddesc` field might be “Finished.”

You should also be prepared to handle network/connectivity and HTTP errors.

On some occasions you might receive a SOAP Fault in the SOAP Header instead of the expected result. We do not currently have any predefined SOAP Faults, you should treat these as you would a runtime exception.

Where To Go From Here

Now that you are a little more familiar with DuoShare web services, you should look at our web service descriptions, each service description has its own documentation and API details.

Appendix A: Web Service Descriptions

CustaccessEF

WSDL: <https://duoshare.com/dsWS/services/CustaccessEF/wsd/CustaccessEF.wsd>

Use CustaccessEF to lookup the customers the current user has access to. This is useful for allowing users to choose which customer will be used by your application.

JobcontrolEF

WSDL: <https://duoshare.com/dsWS/services/JobcontrolEF/wsd/JobcontrolEF.wsd>

JobcontrolEF allows you to create, schedule, delete, and modify jobs on DuoShare.

ListdetailEF

WSDL: <https://duoshare.com/dsWS/services/ListdetailEF/wsd/ListdetailEF.wsd>

Allows finding, deleting, creating, and updating of a ListdetailRecord. PostaloutputRecords are stored as ListdetailRecords on DuoShare, if you need to purge records from a list on DuoShare, or simply need to see if a specific record exists, this is a good choice. This service does not validate interactively, it only stores and retrieves ListdetailRecords.

ListEF

WSDL: <https://duoshare.com/dsWS/services/ListEF/wsd/ListEF.wsd>

Use this to manage your lists and handle ListRecords that contain general information about a list, including the list's status.

PostML

WSDL: <https://duoshare.com/dsWS/services/PostML/wsd/PostML.wsd>

Provides developers with the ability to do interactive validation of a single address or a batch of 10. Provides hint data upon request.

UseraccessEF

WSDL: <https://duoshare.com/dsWS/services/UseraccessEF/wsd/UseraccessEF.wsd>

Gives you a user's access records. This will allow you to see to what customers a user has access. If the user has access to multiple companies and stores on DuoShare, you can prompt them for which company/store they would like to use.

UserEF

WSDL: <https://duoshare.com/dsWS/services/UserEF/wsd/UserEF.wsd>

Use this to get the currently logged-in user's UserRecord. UseraccessEF and CustaccessEF both require a

UserRecord to retrieve a user's access details.