The fastest way to get a signature.™

DocuSign Connect API

V3.0

Developer Guide

DocuSign®

DocuSignAPI_DevGuide_v3.0_09302009

Table of Contents

Chapter 1: DocuSign API

Definitions of some commonly used terms are given here to familiarize the API user with their use in the DocuSign system.

- **Envelope -** This represents a package used to mail documents to recipients. The envelope carries information about the sender and timestamps to indicate the progress of the delivery procedure. It contains collections of Documents, Tabs and Recipients.
- **Document -** A PDF document that is to be delivered, representing the content to be reviewed and/or signed. Documents have names and are always base64 encoded while in the system.
- **Tab -** This represents a 'stick e-tab' on a document. It is used in two ways. First, it is used to indicate to a recipient where a signature or initials are required. Second, it is used to include various bits of information in a document in a manner similar to Form Fields or Macros. For example, a tab may automatically fill in the Company Name of a recipient when the document is signed.
- **Recipient -** Someone who receives the envelope and, optionally signs and initials the documents where indicated by tabs.

Basic Process Flow

This is a general overview of how the DocuSign system works to familiarize API users with terms and process. It is not a faithful reproduction of how the process is implemented, either on the website or in the API.

A sender has a set of documents that they would like to have signed. The sender supplies the name and email address of each person they want to sign the document, and marks the documents with tabs to indicate where each party should sign or initial. The sender may also choose to let the receiving party freeform sign the document. The sender then places the document into the DocuSign system. The DocuSign system then notifies each recipient, via the supplied email address(es), that they have been asked to sign a document, and provides a link to the envelope.

When a recipient clicks the link, they will see the documents with yellow 'stick e- tabs' in the locations specified by the sender, representing where they need to click to sign or initial the document. If the sender chooses to use freeform signing the recipient will see the document with a note letting them know to click on the document to sign.

When all recipients have clicked on all of their respective 'stick e-tabs' or added their signature via freeform signing, the document is signed.

When this process is initiated via the DocuSign Connect API, the envelope can either be completely specified, which is to say that all 'stick e-tabs' (or freeform signing), documents and recipients have been either supplied in the request, or partially specified, meaning that at least one document is present, but recipients and 'stick e-tabs' may not be. This second case is used to create a 'draft' envelope for the sender and allows him or her to finish sending it online through the web interface.

API Overview

The DocuSign API provides methods that allow partner companies' servers to integrate the DocuSign service into their applications. The service allows partners to build solutions that:

- Submit partially specified envelopes for later completion by the customer.
- Submit completely specified envelopes that are immediately processed for delivery.
- Void an envelope that has been submitted but not yet completed.
- Retrieve the status of an envelope.
- Retrieve the completed PDF of every document in an envelope.
- Retrieve the completed PDF for each separate document in an envelope.
- Retrieve the consumer disclosure acceptance status of a recipient.
- Transfer an envelope to another DocuSign user or account.
- Correct recipient information for an existing envelope en route.
- Resend a notification email to an existing recipient.
- Retrieve the Member level permissions for the optional features.
- Purge the envelope contents from the DocuSign system.
- Withdraw an Authoritative Copy of the envelope.
- Retrieve a list of audit events pertaining to a particular envelope.

These methods can be used by themselves or in addition to linking the customer's experience to the DocuSign site to complete any partially completed processes.

Methods Exposed in the DocuSign API

The DocuSign API exposes the following methods:

- **AcknowledgeAuthoritativeCopyExport** – Returns the key to decrypt the documents returned in the ExportAuthoritativeCopy method. Removes the Authoritative Copy from DocuSign. Available only in the 3.0 API.
- **CorrectAndResendEnvelope** - Correct the specified recipients of the envelope. Allows a sender to modify the name and/or email address of a recipient, change the envelope access authentication information for a recipient, or resend the envelope notification email to a recipient. Supports multiple corrections within an envelope. The return value has a Boolean CorrectionSucceeded for each correction, which indicates the success of the recipient correction.

- **CreateAndSendEnvelope** - Creates the envelope and initiates the delivery process. This method requires an envelope that includes all necessary information; no sender interaction via the DocuSign website is required.
- **CreateEnvelope** – Creates the envelope without sending it. These envelopes do not contain all of the information necessary for immediate processing. Instead, the sender can be directed to the DocuSign web site to complete the preparation of the envelope.
- **CreateEnvelopeFromTemplates** - **–** Creates and sends envelope based on DocuSign Pro templates and envelope information. Available only in the 3.0 API.
- **EnvelopeAuditEvents** - Returns a XML document of the current envelope events. Available only in the 3.0 API.
- **ExportAuthoritativeCopy** – Export an Authoritative Copy envelope from DocuSign. Returns an encrypted collection for all the documents and an export transaction ID for a given envelope. To decrypt the documents and remove the actual Authoritative Copy from DocuSign the method AcknowledgeAuthoritativeCopyExport must be called. Available only in the 3.0 API.
- **GetAccountMembershipFeaturesList** – Returns the member level permissions for the optional features. The optional features include DocuSign Professional, eOriginal Vaulting, SequentialSigningAPI, SequentialSigningUI and TransactionPoint.
- **GetAccountSettingsList** – Returns the settings for an account.
- **GetRecipientList** – Return a collection of RecipientRecords for the supplied recipient Email address. The purpose of this function is to allow a sender to determine what recipients are available in the system at the given email address.
- **GetRecipientEsignList** – Return a collection of RecipientEsignRecords for the supplied Sender UserName, Email address, and the recipient Email address. The purpose of this function is to allow a sender to determine if an Esign agreement already exists between the sender (as identified by the UserName and Email address) and the recipient (identified by email address only). Each RecipientEsignRecord returned (there may be more than one if there are several recipients at the same email address) has the Recipient's UserName, Email address, and a Boolean indicating whether or not an Esign Agreement exists for that recipient.
- **GetStatusInDocuSignConnectFormat** – This method is reserved for future use.
- **Ping** - Returns true if this method can be reached. Use this method for testing availability. Available only in the 3.0 API.
- **PurgeDocuments** – Allows a sender to purge the documents from an envelope. Only completed documents can be purged.
- **RequestDocumentPDFs** - Return a collection of all of the documents in an envelope. This method differs from RequestPDF in that this method returns individual documents, while RequestPDF returns all of the documents combined into a single, contiguous PDF. Additionally, the RequestDocumentPDFs method

returns the Signing Certificate pdf document, which details the specific attributes of the participants and landmark events of the signing transaction.

- **RequestDocumentPDFsEx** - Returns an extended version the collection of all of the documents from RequestDocumentPDFs. Extensions include originating document ID and document type. Available only in the 3.0 API.
- **RequestDocumentPDFsRecipientsView** – Returns the documents that can be viewed by the given recipient. This is especially useful when using document visibility settings with your account.
- **RequestPDF** – Query the envelope and return its final PDF.
- **RequestRecipientToken** - Specific to InSession Signing implementations.
- **RequestStatus** – Query the envelope and return its current status.
- **RequestStatusEx** - Query the envelope and return its extended status. Available only in the 3.0 API.
- **RequestStatuses** - Query DocuSign for a collection of EnvelopeStatuses. The request can be filtered by date range, StatusCode, sending user or EnvelopeID.
- **RequestStatusesEx** - Query DocuSign for a collection of extended EnvelopeStatuses. The request can be filtered by date range, StatusCode, sending user or EnvelopeID. Available only in the 3.0 API.
- **SendEnvelope** – Send and envelope that is in draft status. Upon sending all the validations that are made with the CreateAndSendEnvelope method apply.
- **TransferEnvelope** - Transfers ownership of the specified envelope to the specified User in the specified Account. Returns Boolean to indicate success of the operation.
- **VoidEnvelope** - Voids the envelope. Returns Boolean to indicate success of the operation.

Authentication

The API methods require authentication. This authentication is supplied as a WS-Security UsernameToken in the SOAP header.

Values for the UsernameToken elements must be populated as follows:

- Username: The DocuSign UserId of the User making the API request.
- Password: The plain-text DocuSign password of the User specified by Username.

Example UsernameToken

```
<wsse:Security soap:mustUnderstand="1">
  <wsu:Timestamp wsu:Id="Timestamp-0741d0e0-529f-49bc-bf86-653238d2532b">
  <wsu:Created>2006-01-02T21:26:04Z</wsu:Created>
  <wsu:Expires>2006-01-02T21:31:04Z</wsu:Expires>
  </wsu:Timestamp>
```

```
  <wsse:UsernameToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="SecurityToken-8d4e766e-a8a2-4bb3-a327-89c34bc7f85f">
    <wsse:Username>caa26663-927b-4800-bfdf-d115d1c72f20</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:Nonce>RumCR4p6U4a7hiX9lUlGWA==</wsse:Nonce>
    <wsu:Created>2006-01-02T21:26:04Z</wsu:Created>
  </wsse:UsernameToken>
</wsse:Security>
```

Integrators Key

The API methods require an integrator key to allow you to send requests with the users passed via the UsernameToken. Users with DocuSign privileges will be allowed to be passed in the UsernameToken as long as a valid integrator key is provided. The integrator key is provided to API developers by DocuSign. The integrator key must be placed in front of the user ID that is in the Username node of the UsernameToken. The integrator key must be wrapped with brackets, "[and]". Example format:
<wsse:Username>**[Key Here]**2988541c-4ec7-4245-b520-f2d324062ca3</wsse:Username>

Sample SOAP Header:

```
   <soap:Header>
     <wsa:Action>http://www.docusign.net/API/3.0/GetRecipientEsignList</wsa:Action>
      <wsa:MessageID>uuid:3f9d7626-c088-43b4-b579-2bd5e8026b17</wsa:MessageID>
      <wsa:ReplyTo>
 <wsa:Address>http://schemas.xmlsoap.org/ws/2004/03/addressing/role/anonymous</wsa:Address>
      </wsa:ReplyTo>
      <wsa:To>http://demo.docusign.net/api/3.0/api.asmx</wsa:To>
      <wsse:Security soap:mustUnderstand="1">
        <wsu:Timestamp wsu:Id="Timestamp-8838aa24-9759-4f85-8bf2-26539e14f750">
          <wsu:Created>2006-04-14T14:29:23Z</wsu:Created>
          <wsu:Expires>2006-04-14T14:34:23Z</wsu:Expires>
        </wsu:Timestamp>
        <wsse:UsernameToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="SecurityToken-7c7b695e-cef7-463b-b05a-9e133ea43c41">
          <wsse:Username>[Integrator Key Here]2988541c-4ec7-4245-b520-f2d324062ca3</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:Nonce>SjlScsL5q3cC1CDWrcMx3A==</wsse:Nonce>
          <wsu:Created>2006-04-14T14:29:23Z</wsu:Created>
        </wsse:UsernameToken>
      </wsse:Security>
    </soap:Header>
```

Optional Authentication Mechanism: XML Signature

In addition to the UsernameToken, any account may elect to enforce that all API requests be signed with a valid third-party X.509 certificate 2. If enabled, DocuSign Connect API will validate that the SOAP Body of the message is signed. Please contact your DocuSign representative to enable this setting. Supported certificate authorities are VeriSign and Thawte.

Example SOAP Envelope with XML Signature

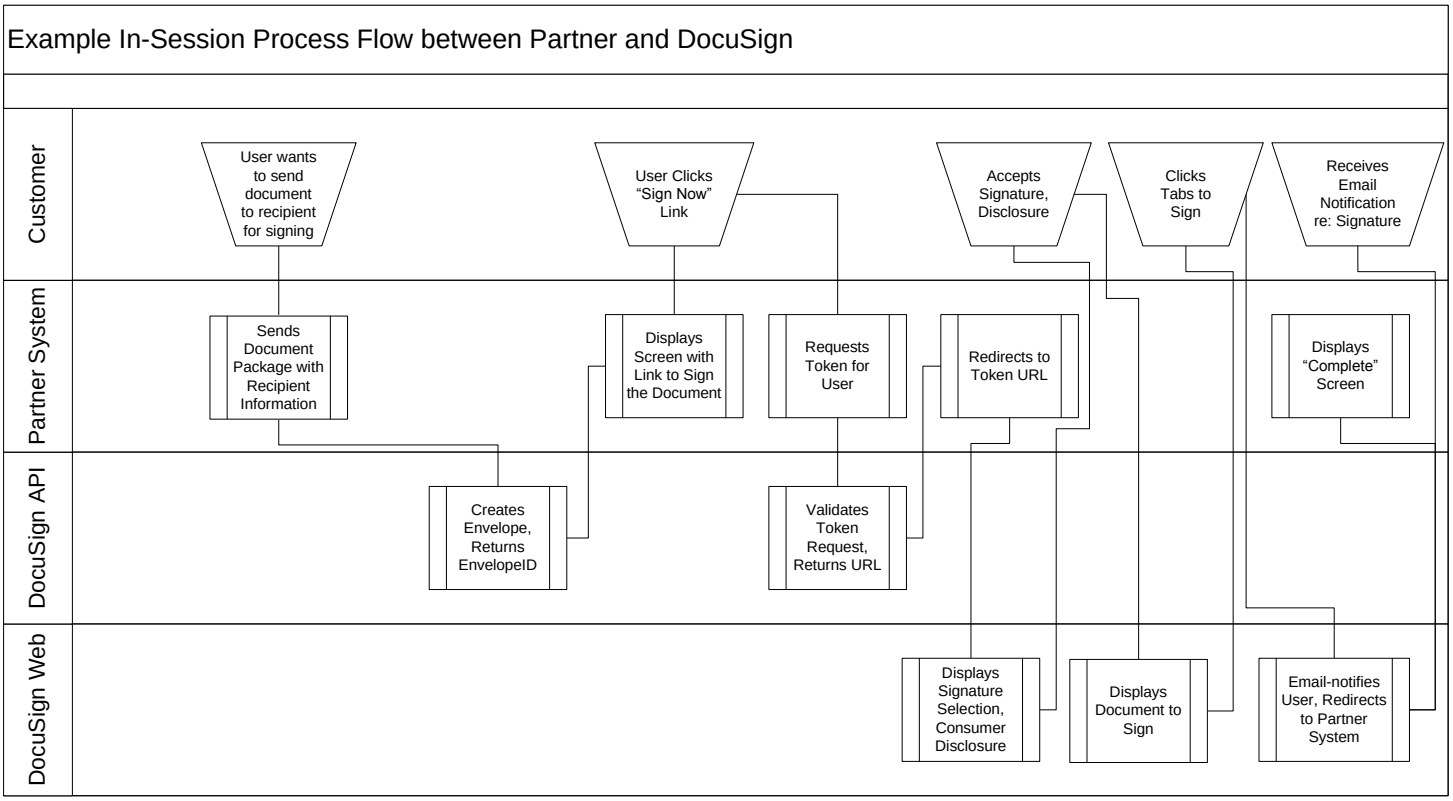
```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:wsa="http://schemas.xmlsoap.org/ws/2004/03/addressing" xmlns:wsse="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">
   <soap:Header>
    <wsa:Action>http://www.docusign.net/API/3.0/GetRecipientEsignList</wsa:Action>
     <wsa:MessageID>uuid:3f9d7626-c088-43b4-b579-2bd5e8026b17</wsa:MessageID>
     <wsa:ReplyTo>
 <wsa:Address>http://schemas.xmlsoap.org/ws/2004/03/addressing/role/anonymous</wsa:Address>
     </wsa:ReplyTo>
     <wsa:To>http://demo.docusign.net/api/3.0/api.asmx</wsa:To>
     <wsse:Security soap:mustUnderstand="1">
       <wsu:Timestamp wsu:Id="Timestamp-8838aa24-9759-4f85-8bf2-26539e14f750">
         <wsu:Created>2006-04-14T14:29:23Z</wsu:Created>
         <wsu:Expires>2006-04-14T14:34:23Z</wsu:Expires>
       </wsu:Timestamp>
       <wsse:UsernameToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd" wsu:Id="SecurityToken-7c7b695e-cef7-463b-b05a-9e133ea43c41">
         <wsse:Username>2988541c-4ec7-4245-b520-f2d324062ca3</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:Nonce>SjlScsL5q3cC1CDWrcMx3A==</wsse:Nonce>
         <wsu:Created>2006-04-14T14:29:23Z</wsu:Created>
       </wsse:UsernameToken>
       <wsse:BinarySecurityToken ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-
token-profile-1.0#X509v3" EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-
message-security-1.0#Base64Binary" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
wssecurity-utility-1.0.xsd" wsu:Id="SecurityToken-b7825bf5-1d1c-446c-8f99-
88dff075fab8">MIIEdDCCAlygAwIBAgIDAgIOMA0GCSqGSIb3DQEBBQUAMHkxEDAOBgNVBAoTB1Jvb3QgQ0ExHjAcBg
NVBAsTFWh0dHA6Ly93d3cuY2FjZXJ0Lm9yZzEiMCAGA1UEAxMZQ0EgQ2VydCBTaWduaW5nIEF1dGhvcml0eTEhMB8G
CSqGSIb3DQEJARYSc3VwcG9ydEBjYWNlcnQub3JnMB4XDTA2MDMwNzE3NTU0NVoXDTA2MDkwMzE3NTU0NVowJjEk
MCIGA1UEAxMbZGVtb3NhbXBsZWNlcnQuZG9jdXNpZ24uY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDhg
GA7HXHcotoKuVTN5+3yqeL1A4M/bYNLKyFFQYYFANVXcb7D+la2Oh8YI2TBHTgNGccJxeCsefZh1LHLguE4/kMMvz62m
bdohhnjXeJH55O4YqdILxZ3r1EJbOrSZyHEwn1/PvGwj2cDF0QHnvqfgvsuozlJmRYNKXpnD9QzzwIDAQABo4HbMIHYMA
wGA1UdEwEB/wQCMAAwNAYDVR0lBC0wKwYIKwYBBQUHAwIGCCsGAQUFBwMBBglghkgBhvhCBAEGCisGAQQBgjcKA
wMwCwYDVR0PBAQDAgWgMDIGCCsGAQUFBwEBBCYwJDAiBggrBgEFBQcwAYYWaHR0cDovL29jc3AuY2FjZXJ0Lm9yZz
BRBgNVHREESjBIghtkZW1vc2FtcGxlY2VydC5kb2N1c2lnbi5jb22gKQYIKwYBBQUHCAWgHQwbZGVtb3NhbXBsZWNlcnQ
uZG9jdXNpZ24uY29tMA0GCSqGSIb3DQEBBQUAA4ICAQBzgTyhXvDMmt6UwB6ZR8qVGa4Jhch68kS0X5vQjMa0wXdfJjU
3C13rnNujX8wefPOrX79vyS4CVVXW3QRTUp+hyaodwi2ed0msz0vo71VUU6KGUNp3GClY5NjMzmz9y900SjL74shWiDsb
hT/yMGDp8cZt6nFSUabhaBM5NFRvguwI1hAjugT6QnLjeUxblVuS2s2b90Kj+jh/w3gw5f/0XCxgNBdz66jvxFxk0wKl3xLoL
s/a2nh4TOvEdUixZEPvsJyQNFY/+5cmlVM6/Nl/T2TjMcEuvhpBvizDvNykluEdi0R5vftkURlE/InmmGt37eH3xB3YeyFDR7u
Y6qJREBswNWKd8xqK7cb4XXntY1L5XEIdW/5ZNn3JdAg2Vq05lHF5+i/i+5Uc4GDcQlxmTbiSvt4z0tobnHUsrsrBbLRlcmzr
uZbX5OKgJ/r1hO3JTEhFzR5KCogDfsCdAqCC+ClKc0GPSSgt6Cu36F5ytE04KTXBltU/Ex3kDDIlm7OX/os7kSIZf9IAejr4Cx
x1wdenjy//n89qLjbd0u11e038g41/orQiBYn7opj3wmbPdRZ+gsMpIxhhnmUMFWMIJMIidzZwRa9n3E+l49ZjElCDuBRnU
oaGDhmSnEYJrtv/uaQ5UulvMtMge7FrqiIDzI1A++nBfGRMS0EHLdfBdg==</wsse:BinarySecurityToken>
      <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
        <SignedInfo>
          <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#" />
          <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1" />
          <Reference URI="#Id-6c2377a1-a116-444c-8c1e-101543d5d721">
            <Transforms>
              <Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#" />
            </Transforms>
            <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
            <DigestValue>auFLuvdCM5tbsrruRJUY7h0vf9E=</DigestValue>
          </Reference>
        </SignedInfo>

<SignatureValue>d21v1CT2UxIus64mXbfCxO6xjJ3gbm+cE14HUnlOJmA9QsB5M5L1ric4cvMcOky0hFUDZrhDn5FVUXt
LKJQgV5wqTtmeHi3NT6HNEh1Rrq/VFj/O4/rFGyc4JrzkVxZpqiwRYOee1Empv6iHq3Uf4PIeAv1Tn/qn/b09P+D7FTo=</S
ignatureValue>
        <KeyInfo>
```

```
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#SecurityToken-b7825bf5-1d1c-446c-8f99-88dff075fab8"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-profile-1.0#X509v3" />
          </wsse:SecurityTokenReference>
        </KeyInfo>
      </Signature>
    </wsse:Security>
  </soap:Header>
  <soap:Body wsu:Id="Id-6c2377a1-a116-444c-8c1e-101543d5d721">
    <GetRecipientEsignList xmlns="http://www.docusign.net/API/2.0">
      <UserName>UserName</UserName>
      <SenderEmail>username@email.com</SenderEmail>
      <RecipientEmail>name@email.com</RecipientEmail>
        <AccountId>2184872a-8f6c-4f18-b808-1ec864cec29d</AccountId>
    </GetRecipientEsignList>
  </soap:Body>
</soap:Envelope>
```

Example Usage Diagram

The below diagram depicts In-session signing process flow. Diagram shows one of the basic process and meant only to explain how different systems interact with each other.



Placing 'stick e-tabs' aka Tabs:

The location of Tabs can be specified in one of two ways:

1. Passing the explicit x/y coordinates of each tab in the xml when creating the Envelope. This is appropriate for documents with a standard format, where the signature lines and/or other data items collected by DocuSign tabs are always located in the same place on the document. In this scenario, the xml request describes *where* to locate DocuSign tabs.

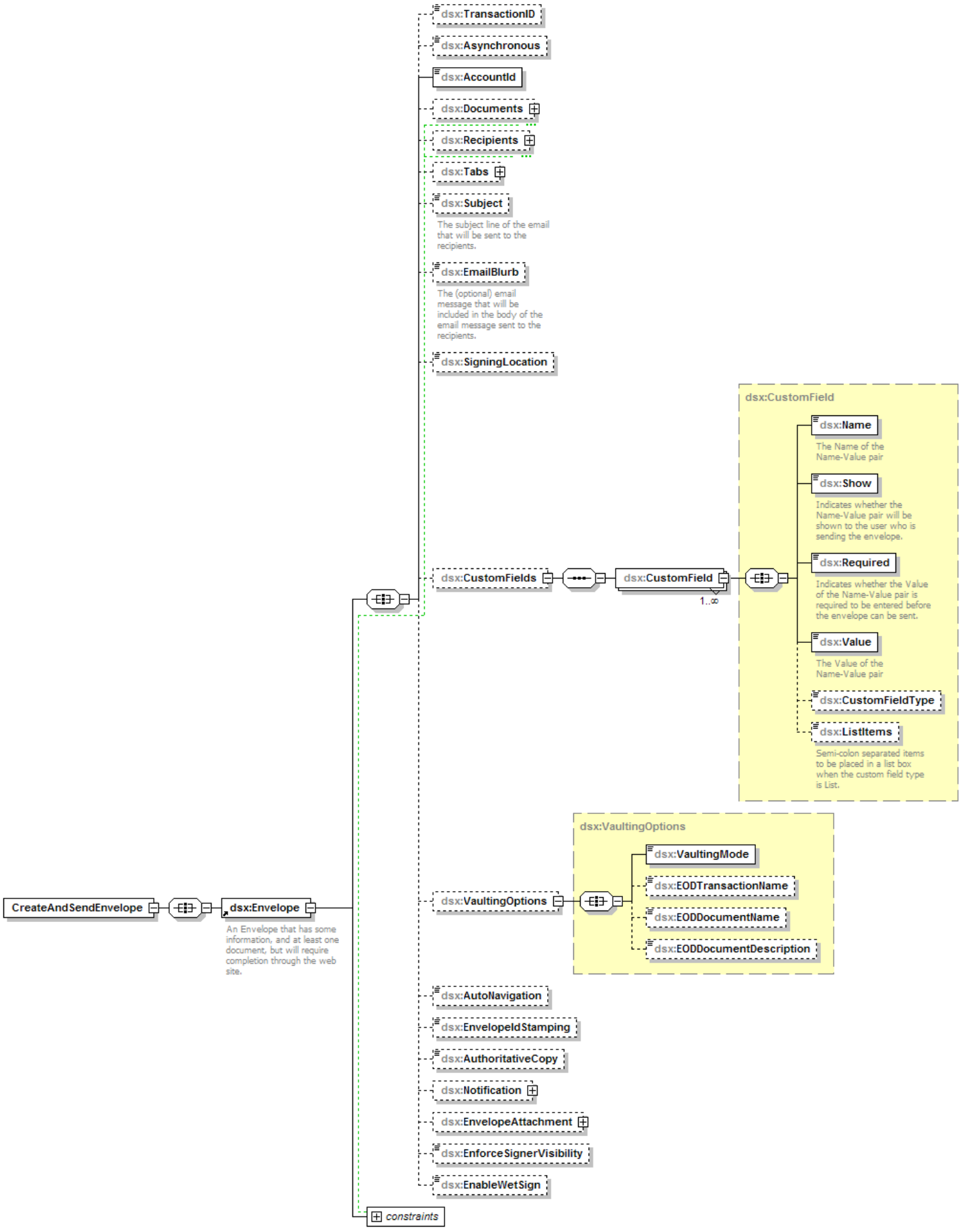
2. Specifying "anchor" strings that exist in the document, against which the tabs should be placed. This is appropriate for documents that have variable content/format, where the tab locations must similarly vary. In this scenario, the xml request describes *how* to locate DocuSign tabs.

Chapter 2: Envelope Creation

This chapter outlines the usage rules and behaviors of the *CreateEnvelope and CreateAndSendEnvelope* call which is used to create envelopes, specify recipients, documents and actions on those documents within that envelope. Note that since schema is similar, only one method is detailed.

CreateAndSendEnvelope

Schema


dsx:TransactionID
dsx:Asynchronous
dsx:AccountId
dsx:Documents
dsx:Recipients
dsx:Tabs
dsx:Subject
The subject line of the email that will be sent to the recipients.
dsx:EmailBlurb
The (optional) email message that will be included in the body of the email message sent to the recipients.
dsx:SigningLocation
dsx:CustomFields
dsx:CustomField
1..∞
dsx:CustomField
dsx:Name
The Name of the Name-Value pair
dsx:Show
Indicates whether the Name-Value pair will be shown to the user who is sending the envelope.
dsx:Required
Indicates whether the Value of the Name-Value pair is required to be entered before the envelope can be sent.
dsx:Value
The Value of the Name-Value pair
dsx:CustomFieldType
dsx:ListItems
Semi-colon separated items to be placed in a list box when the custom field type is List.
dsx:VaultingOptions
dsx:VaultingMode
dsx:EODTransactionName
dsx:EODDocumentName
dsx:EODDocumentDescription
CreateAndSendEnvelope
dsx:Envelope
An Envelope that has some information, and at least one document, but will require completion through the web site.
dsx:AutoNavigation
dsx:EnvelopeIdStamping
dsx:AuthoritativeCopy
dsx:Notification
dsx:EnvelopeAttachment
dsx:EnforceSignerVisibility
dsx:EnableWetSign
constraints

Name	Schema Type	Description
TransactionID	LongString	External ID of this transaction to be used when calling SynchEnvelope. This ID must be unique for a 24-hour period.
Asynchronous	boolean	If true, will queue the envelope for processing and EnvelopeStatus will have a value of 'Processing'. Use SynchEnvelope to determine when the queued envelope has been processed. Additionally, RequestStatus calls will return 'Processing' until completed.
AccountId	DSXId	Account Id of the user who created the envelope
Documents	Document	Complex element contains the details on the documents in the envelope. Please refer the section below.
Recipients	*Recipient*	Specifies the envelope recipients. See the Recipient section below for more information.
Tabs	Tab	Specifies information of the tabs affixed on the documents. Please refer the section below for details.
Subject	LongString	The subject of the email that will be sent to the recipient.
EmailBlurb		Optional element, if specified is included in email notifications to the envelope recipients.
SigningLocation	SigningLocationCode	Specifies the physical location where the signing takes place. It can have two enumeration values; InPerson and Online.
CustomFields	CustomField	Complex element contains a list of names and values. Element can specify if the name value pair needs to be entered before sending the envelope. It can also specify if the name value pair needs to be shown to user who is sending the envelope. Reserved CustomField names: ##SFAccount ##SFContract ##SFOpportunity ##SFCase Reserved field names can be used to link data from DocuSign Connect to Salesforce.
VaultingOptions	VaultingOptions	Specifies that, on document completion, this document is to be sent to an electronic vaulting

		solution. You MUST have an established relationship with the electronic vault and explicitly have vaulting permissions enabled to use this feature. Contact your DocuSign representative for more information.
AutoNavigation	Boolean	Specifies the Auto Navigation feature. If the Boolean is set to true the auto navigation is enabled.
EnvelopeIDStamping	Boolean	Specifies the Envelope Stamping feature. If the Boolean is set to true the Envelope Stamping is enabled.
AuthoritativeCopy	Boolean	Specifies the Authoritative copy feature. If the Boolean is set to true the Authoritative copy feature is enabled.
EnvelopeAttachment	Attachment	It would be possible to send attachments with envelope. This complex element contains data in base64Binary. It also contains label and type for the attachment.
Notification	Notification	Specifies information of the notification options for the envelope. Please refer the section below for details.
EnforceSignerVisibility	Boolean	When true requires that a Signer have a signature or initial on the document or that the document has no signers in order to view it.
EnableWetSign	Boolean	When true, allows the signer to print the document and sign it on paper.

Sample Request XML

```
SOAPAction: "http://www.docusign.net/API/3.0/CreateAndSendEnvelope"

<?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>
    <CreateAndSendEnvelope xmlns="http://www.docusign.net/API/3.0">
      <Envelope>
        <TransactionID>string</TransactionID>
        <Asynchronous>boolean</Asynchronous>
        <AccountId>string</AccountId>
        <Documents>
          <Document>
            <ID>positiveInteger</ID>
            <Name>string</Name>
            <PDFBytes>base64Binary</PDFBytes>
            <Password>string</Password>
            <TransformPdfFields>boolean</TransformPdfFields>
            <FileExtension>string</FileExtension>
```

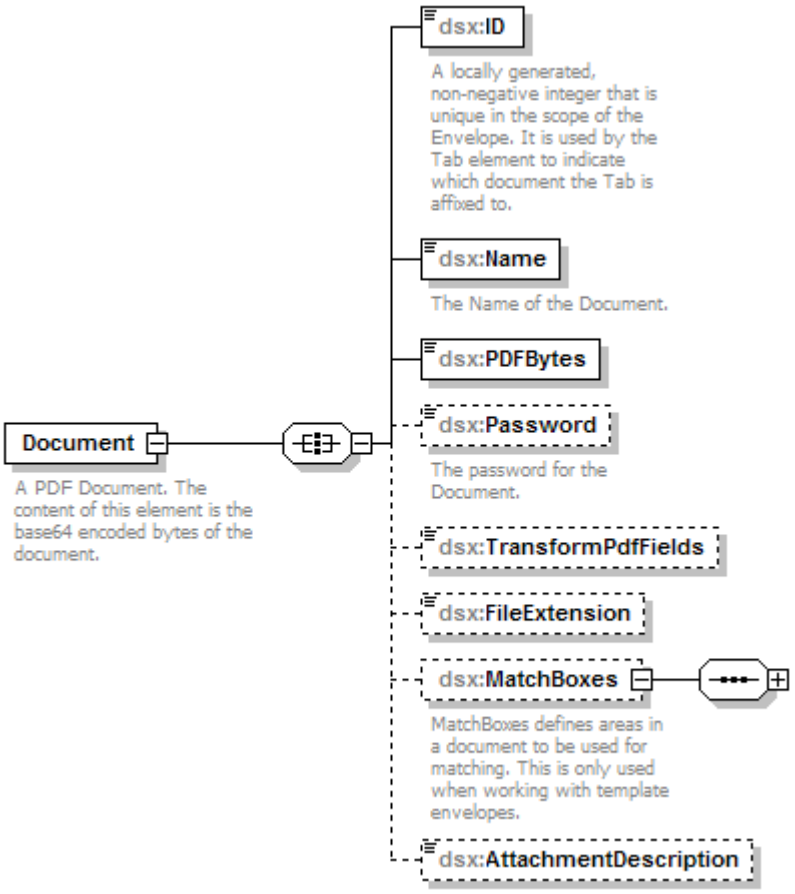
```
    <MatchBoxes xsi:nil="true" />
    <AttachmentDescription>string</AttachmentDescription>
  </Document>
  <Document>
    <ID>positiveInteger</ID>
    <Name>string</Name>
    <PDFBytes>base64Binary</PDFBytes>
    <Password>string</Password>
    <TransformPdfFields>boolean</TransformPdfFields>
    <FileExtension>string</FileExtension>
    <MatchBoxes xsi:nil="true" />
    <AttachmentDescription>string</AttachmentDescription>
  </Document>
</Documents>
<Recipients>
  <Recipient>
    <ID>positiveInteger</ID>
    <UserName>string</UserName>
    <SignerName>string</SignerName>
    <Email>string</Email>
    <Type>Signer or CarbonCopy or CertifiedDelivery or InPersonSigner</Type>
    <AccessCode>string</AccessCode>
    <AddAccessCodeToEmail>boolean</AddAccessCodeToEmail>
    <RequireIDLookup>boolean</RequireIDLookup>
    <IDCheckConfigurationName>string</IDCheckConfigurationName>
    <SignatureInfo xsi:nil="true" />
    <CaptiveInfo xsi:nil="true" />
    <CustomFields xsi:nil="true" />
    <RoutingOrder>unsignedShort</RoutingOrder>
    <IDCheckInformationInput xsi:nil="true" />
    <AutoNavigation>boolean</AutoNavigation>
    <RecipientAttachment xsi:nil="true" />
    <Note>string</Note>
    <RoleName>string</RoleName>
    <TemplateLocked>boolean</TemplateLocked>
    <TemplateRequired>boolean</TemplateRequired>
    <TemplateAccessCodeRequired>boolean</TemplateAccessCodeRequired>
  </Recipient>
  <Recipient>
    <ID>positiveInteger</ID>
    <UserName>string</UserName>
    <SignerName>string</SignerName>
    <Email>string</Email>
    <Type>Signer or CarbonCopy or CertifiedDelivery or InPersonSigner</Type>
    <AccessCode>string</AccessCode>
    <AddAccessCodeToEmail>boolean</AddAccessCodeToEmail>
    <RequireIDLookup>boolean</RequireIDLookup>
    <IDCheckConfigurationName>string</IDCheckConfigurationName>
    <SignatureInfo xsi:nil="true" />
    <CaptiveInfo xsi:nil="true" />
    <CustomFields xsi:nil="true" />
    <RoutingOrder>unsignedShort</RoutingOrder>
    <IDCheckInformationInput xsi:nil="true" />
    <AutoNavigation>boolean</AutoNavigation>
    <RecipientAttachment xsi:nil="true" />
    <Note>string</Note>
    <RoleName>string</RoleName>
    <TemplateLocked>boolean</TemplateLocked>
    <TemplateRequired>boolean</TemplateRequired>
    <TemplateAccessCodeRequired>boolean</TemplateAccessCodeRequired>
  </Recipient>
</Recipients>
<Tabs>
  <Tab>
```

```
            <DocumentID>positiveInteger</DocumentID>
            <RecipientID>positiveInteger</RecipientID>
            <PageNumber>nonNegativeInteger</PageNumber>
            <XPosition>nonNegativeInteger</XPosition>
            <YPosition>nonNegativeInteger</YPosition>
            <ScaleValue>decimal</ScaleValue>
            <AnchorTabItem xsi:nil="true" />
            <Type>InitialHere or SignHere or FullName or Company or Title or DateSigned or
InitialHereOptional or EnvelopeID or Custom or SignerAttachment or SignHereOptional</Type>
            <Name>string</Name>
            <TabLabel>string</TabLabel>
            <Value>string</Value>
            <CustomTabType>Text or Checkbox or Radio or List or Date or Number or SSN or ZIP5 or
ZIP5DASH4 or Email</CustomTabType>
            <CustomTabWidth>int</CustomTabWidth>
            <CustomTabHeight>int</CustomTabHeight>
            <CustomTabRequired>boolean</CustomTabRequired>
            <CustomTabLocked>boolean</CustomTabLocked>
            <CustomTabDisableAutoSize>boolean</CustomTabDisableAutoSize>
            <CustomTabListItems>string</CustomTabListItems>
            <CustomTabListValues>string</CustomTabListValues>
            <CustomTabListSelectedValue>string</CustomTabListSelectedValue>
            <CustomTabRadioGroupName>string</CustomTabRadioGroupName>
            <CustomTabValidationPattern>string</CustomTabValidationPattern>
            <CustomTabValidationMessage>string</CustomTabValidationMessage>
            <TemplateLocked>boolean</TemplateLocked>
            <TemplateRequired>boolean</TemplateRequired>
          </Tab>
          <Tab>
            <DocumentID>positiveInteger</DocumentID>
            <RecipientID>positiveInteger</RecipientID>
            <PageNumber>nonNegativeInteger</PageNumber>
            <XPosition>nonNegativeInteger</XPosition>
            <YPosition>nonNegativeInteger</YPosition>
            <ScaleValue>decimal</ScaleValue>
            <AnchorTabItem xsi:nil="true" />
            <Type>InitialHere or SignHere or FullName or Company or Title or DateSigned or
InitialHereOptional or EnvelopeID or Custom or SignerAttachment or SignHereOptional</Type>
            <Name>string</Name>
            <TabLabel>string</TabLabel>
            <Value>string</Value>
            <CustomTabType>Text or Checkbox or Radio or List or Date or Number or SSN or ZIP5 or
ZIP5DASH4 or Email</CustomTabType>
            <CustomTabWidth>int</CustomTabWidth>
            <CustomTabHeight>int</CustomTabHeight>
            <CustomTabRequired>boolean</CustomTabRequired>
            <CustomTabLocked>boolean</CustomTabLocked>
            <CustomTabDisableAutoSize>boolean</CustomTabDisableAutoSize>
            <CustomTabListItems>string</CustomTabListItems>
            <CustomTabListValues>string</CustomTabListValues>
            <CustomTabListSelectedValue>string</CustomTabListSelectedValue>
            <CustomTabRadioGroupName>string</CustomTabRadioGroupName>
            <CustomTabValidationPattern>string</CustomTabValidationPattern>
            <CustomTabValidationMessage>string</CustomTabValidationMessage>
            <TemplateLocked>boolean</TemplateLocked>
            <TemplateRequired>boolean</TemplateRequired>
          </Tab>
        </Tabs>
        <Subject>string</Subject>
        <EmailBlurb>string</EmailBlurb>
        <SigningLocation>InPerson or Online</SigningLocation>
        <CustomFields>
          <CustomField>
            <Name>string</Name>
```

```
        <Show>string</Show>
        <Required>string</Required>
        <Value>string</Value>
         <CustomFieldType>Text or List</CustomFieldType>
         <ListItems>string</ListItems>
      </CustomField>
      <CustomField>
        <Name>string</Name>
        <Show>string</Show>
        <Required>string</Required>
        <Value>string</Value>
         <CustomFieldType>Text or List</CustomFieldType>
         <ListItems>string</ListItems>
      </CustomField>
    </CustomFields>
    <VaultingOptions>
      <VaultingMode>None or EODeStore or EODAuthoritativeCopy</VaultingMode>
      <EODTransactionName>string</EODTransactionName>
      <EODDocumentName>string</EODDocumentName>
      <EODDocumentDescription>string</EODDocumentDescription>
    </VaultingOptions>
    <AutoNavigation>boolean</AutoNavigation>
    <EnvelopeIdStamping>boolean</EnvelopeIdStamping>
    <AuthoritativeCopy>boolean</AuthoritativeCopy>
    <EnvelopeAttachment>
      <Attachment>
        <Data>base64Binary</Data>
        <Label>string</Label>
        <Type>string</Type>
      </Attachment>
      <Attachment>
        <Data>base64Binary</Data>
        <Label>string</Label>
        <Type>string</Type>
      </Attachment>
    </EnvelopeAttachment>
    <EnforceSignerVisibility>boolean</EnforceSignerVisibility>
    <EnableWetSign>boolean</EnableWetSign>
  </Envelope>
 </CreateAndSendEnvelope>
</soap:Body>
</soap:Envelope>
```

Document

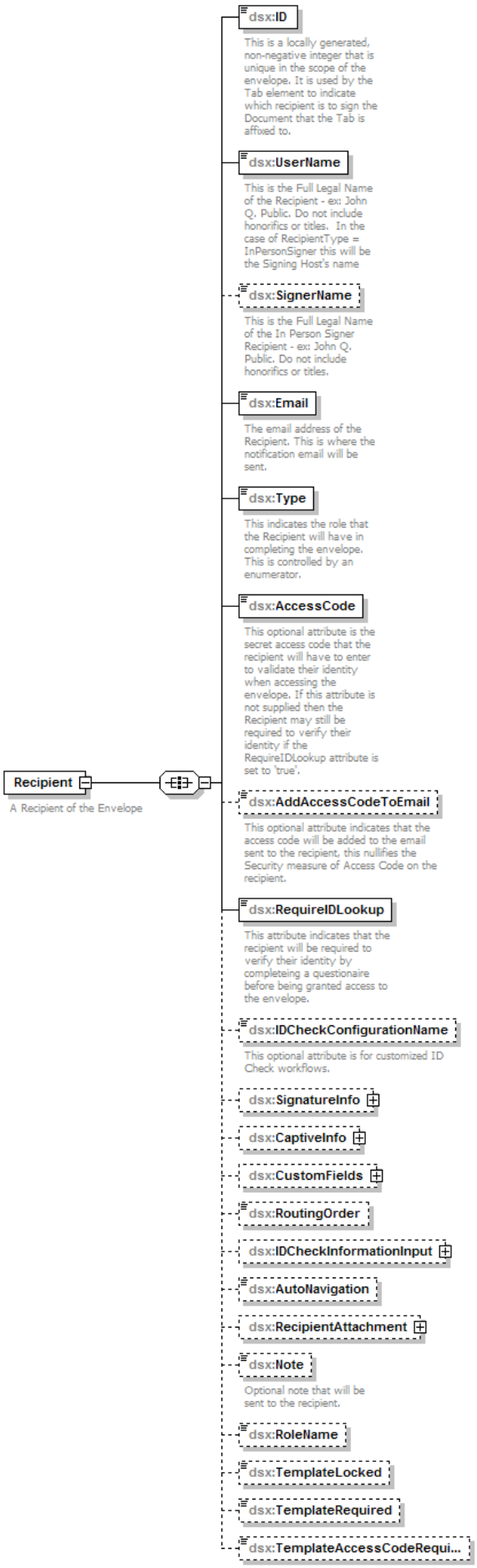
This element contains the details of the documents in the envelope.


dsx:ID
A locally generated, non-negative integer that is unique in the scope of the Envelope. It is used by the Tab element to indicate which document the Tab is affixed to.
dsx:Name
The Name of the Document.
dsx:PDFBytes
dsx:Password
The password for the Document.
Document
A PDF Document. The content of this element is the base64 encoded bytes of the document.
dsx:TransformPdfFields
dsx:FileExtension
dsx:MatchBoxes
MatchBoxes defines areas in a document to be used for matching. This is only used when working with template envelopes.
dsx:AttachmentDescription

Name	Schema Type	Description
ID	LocalId	The unique Id for the document in the specific envelope.
Name	LongString	The name of the PDF document.
PDFBytes	base64Binary	The document byte stream
Password	Password	This will be the password for the document.
TransformPdfFields	Boolean	Optional element. When set to true PDF form field data will be transformed into document tab values when the PDF form field name matches the DocuSign custom tab TabLabel. The resulting PDF form data will also be returned in the PDF meta data when requesting the document PDF.
FileExtension	String	Optional element. File extension of the document. If the document is non-PDF it will be converted to PDF.
MatchBoxes	String	Optional element. Only used when uploading and editing templates. Matchboxes are used to define areas in a document for document matching when creating envelopes. Matchbox schema is defined below in the Manage Templates chapter.
AttachmentDescription	LongString	Optional element. If present, indicates signer will be attaching a document here.

Recipient

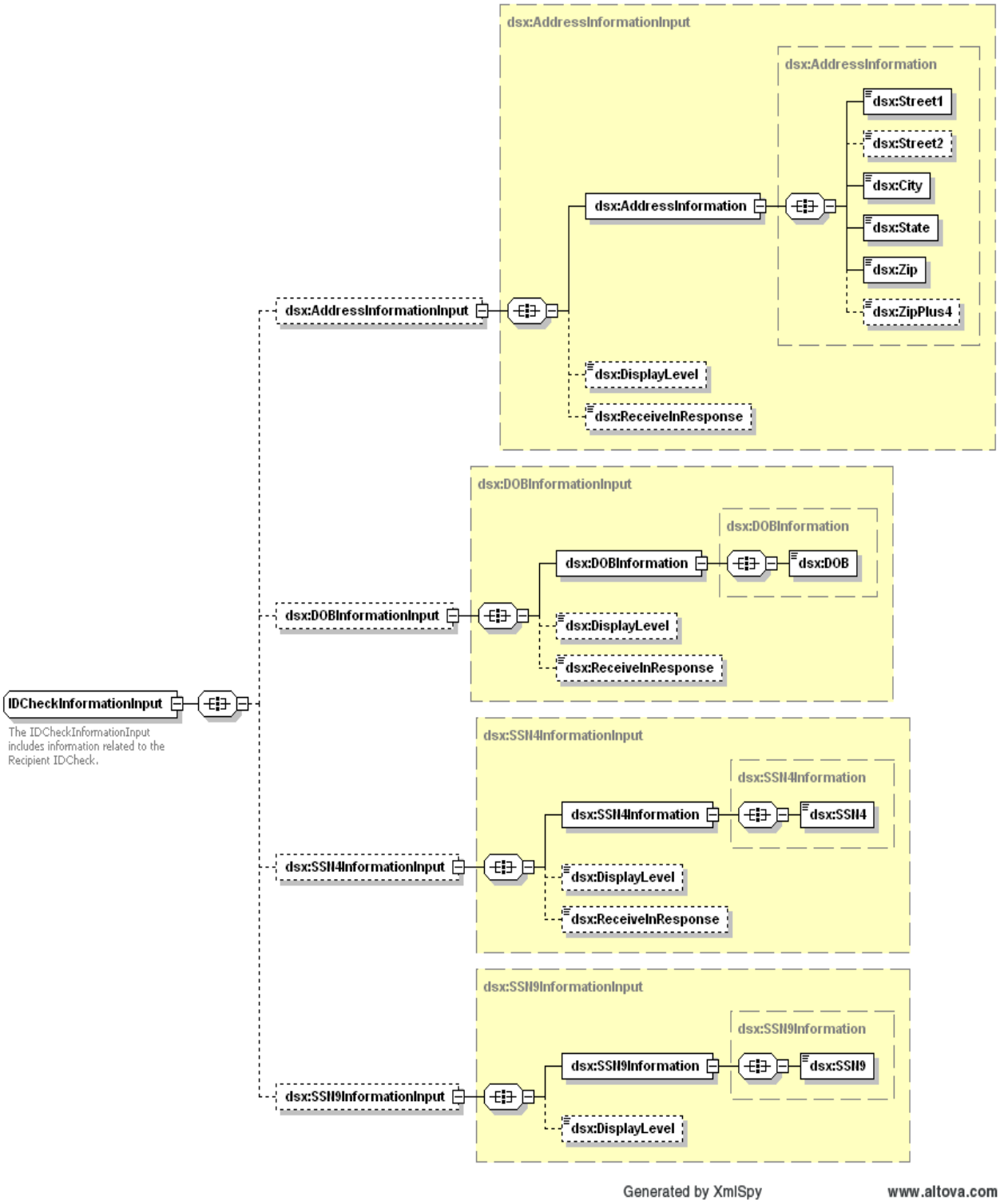
The Recipient element is used to specify envelope recipients.


Recipient
A Recipient of the Envelope
dsx:ID
This is a locally generated, non-negative integer that is unique in the scope of the envelope. It is used by the Tab element to indicate which recipient is to sign the Document that the Tab is affixed to.
dsx:UserName
This is the Full Legal Name of the Recipient - ex: John Q. Public. Do not include honorifics or titles. In the case of RecipientType = InPersonSigner this will be the Signing Host's name
dsx:SignerName
This is the Full Legal Name of the In Person Signer Recipient - ex: John Q. Public. Do not include honorifics or titles.
dsx:Email
The email address of the Recipient. This is where the notification email will be sent.
dsx:Type
This indicates the role that the Recipient will have in completing the envelope. This is controlled by an enumerator.
dsx:AccessCode
This optional attribute is the secret access code that the recipient will have to enter to validate their identity when accessing the envelope. If this attribute is not supplied then the Recipient may still be required to verify their identity if the RequireIDLookup attribute is set to 'true'.
dsx:AddAccessCodeToEmail
This optional attribute indicates that the access code will be added to the email sent to the recipient, this nullifies the Security measure of Access Code on the recipient.
dsx:RequireIDLookup
This attribute indicates that the recipient will be required to verify their identity by completeing a questionaire before being granted access to the envelope.
dsx:IDCheckConfigurationName
This optional attribute is for customized ID Check workflows.
dsx:SignatureInfo
dsx:CaptiveInfo
dsx:CustomFields
dsx:RoutingOrder
dsx:IDCheckInformationInput
dsx:AutoNavigation
dsx:RecipientAttachment
dsx:Note
Optional note that will be sent to the recipient.
dsx:RoleName
dsx:TemplateLocked
dsx:TemplateRequired
dsx:TemplateAccessCodeRequi...

Name	Schema Type	Description
ID	LocalId	Unique for the recipient. It is used by the tab element to indicate which recipient is to sign the Document.
UserName	UserName	Full legal name of the recipient
Email	Email	Email of the recipient. Notification will be sent to this email id
Type	RecipientTypeCode	Specifies the role of the recipient. Role could be Signer recipient, Carbon Copy recipient or Certified Delivery recipient.
AccessCode	LongString	This Optional element specifies the access code a recipient has to enter to validate the identity.
AddAccessCodeToEmail	boolean	This Optional attribute indicates that the access code will be added to the email sent to the recipient; this nullifies the Security measure of Access Code on the recipient.
RequireIDLookup	boolean	If the Boolean is set true then recipients will require to answer questionnaire to validate their identity
IDCheckConfigurationName	LongString	Specify ID check configuration by name. Overrides the default.
SignatureInfo	SignatureInfo	Allows the sender to pre-specify the signature name, signature initials and signature font used in the signature stamp for the recipient.
CaptiveInfo	CaptiveInfo	This specifies if the recipient is embedded or remote. If the ClientUserId is not null then the recipient is embedded.
CustomField	LongString	Allows the sender to send custom data about the recipient. This information is returned in the envelope status but otherwise not used by DocuSign.

RountingOrder	PositiveShort	This element specifies the routing order of the recipient in the envelope
IDCheckInformationInput	*IDCheckInformationInput*	This Complex element contains information related to recipient ID check. It comprises of Address, DOB, SS4 and SS9. Please refer the section below for details.
AutoNavigation	*Boolean*	Specifies the auto navigation setting for recipient.
Subject	LongString	The subject of the email sent to the recipient.
EmailBlurb	*EmailBlurb*	The email message sent to the recipient.
RecipientAttachment	*Attachment*	It would be possible to send attachments with recipients. This complex element contains data in base64Binary. It also contains label and type for the attachment.
Note	String	A note that will be unique to this recipient. This note will be sent to the recipient via the signing email. This note will display in the signing UI near the upper left corner of the document on the signing screen.
RoleName	String	Optional element. Used only when working with template recipients. Role name for the recipient
TemplateLocked	Boolean	Optional element. Used only when working with template recipients. If true, the sender cannot change any attributes of the recipient.
TemplateRequired	Boolean	Optional element. Used only when working with template recipients. If true, the sender may not remove the recipient.
TemplateAccessCodeRequired	Boolean	Optional element. Used only when working with template recipients. If true and TemplateLocked = true, the sender must enter an access code.

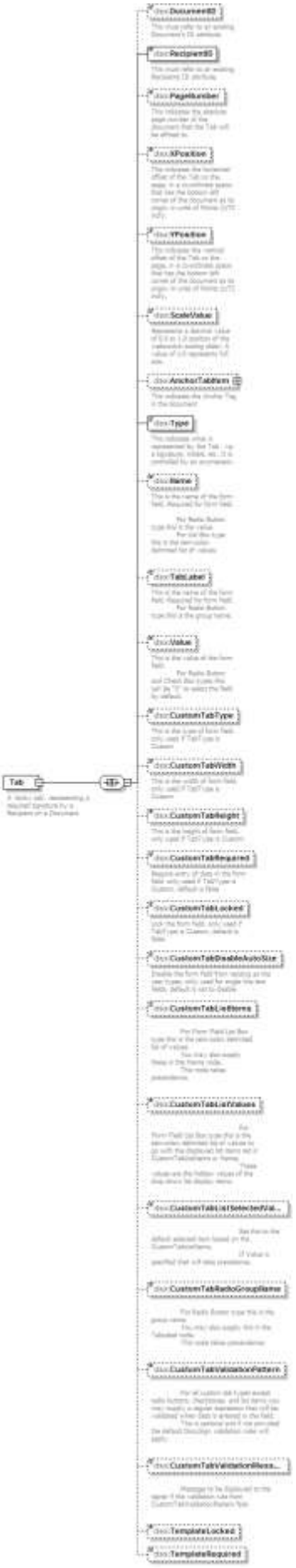
IDCheckInformationInput



Name	Schema Type	Description
AddressInformationInput	AddressInformation	This Complex type contains the following information; Street1, Street2, City, State

		and Zip and ZipPlus4. DisplayLevel specifies the display level for the recipient which is controlled by an enumerator. Display level could be ReadOnly, Editable or DoNotDisplay. Boolean element ReceiveInResponse specifies if the information needs to be returned in the response.
DoBInformationInput	DOBInformation	Contains DOB information; date, Month and Year. This complex element also contains DisplayLevel and ReceiveInResponse elements.
SSN4InformationInput	SSN4Information	The last four digits of the social security number of the recipient. This complex element also contains DisplayLevel and ReceiveInResponse elements.
SSN9InformationInput	SSN9Information	Social security number of the recipient. This element contains the DisplayLevel. SSN9 is never returned in the response.

Tab


Tab
dsx:DocumentID
dsx:RecipientID
dsx:PageNumber
dsx:XPosition
dsx:YPosition
dsx:ScaleValue
dsx:AnchorTabItem
dsx:Type
dsx:TabLabel
dsx:Value
dsx:CustomTabType
dsx:CustomTabWidth
dsx:CustomTabHeight
dsx:CustomTabRequired
dsx:CustomTabLocked
dsx:CustomTabDisableAutoSize
dsx:CustomTabListItems
dsx:CustomTabListValues
dsx:CustomTabListSelectedItem...
dsx:CustomTabRadioGroupName
dsx:CustomTabValidationPattern
dsx:CustomTabValidationMess...
dsx:TemplateLocked
dsx:TemplateRequired

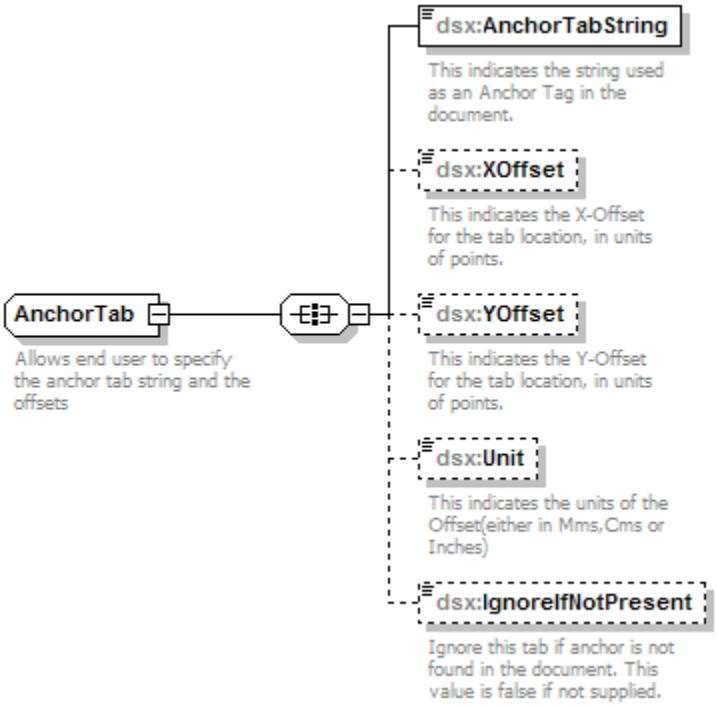
Name	Schema Type	Description
DocumentID	LocalId	This will be the document on which the anchor tab is affixed. Must refer to an existing Document's ID attribute.
RecipientID	LocalId	This specifies the recipient whose signature is needed on the document. Must refer to an existing recipient's ID attribute.
PageNumber	nonNegativeInteger	This specifies the page number where the tab will be affixed.
XPosition	nonNegativeInteger	This indicates the horizontal offset of the tab on the page, in a coordinate space that has left top corner of the document as origin.
YPosition	nonNegativeInteger	This indicates the vertical offset of the tab on the page, in a coordinate space that has left top corner of the document as origin.
ScaleValue	Decimal	Represents a decimal value of 0.0 to 1.0 position of the webcontrol scaling slider. A value of 1.0 represents full size.
AnchorTabItem	AnchorTab	This indicates Anchor tab in the document. See the section below for details
Type	TabTypeCode	It tells what is represented by tab i.e. signature, initial, company, title, date signed, custom etc
Name	Name	This would be the name of the custom tab. Notes: For Type of "Custom" and CustomTabType of "List" the name is a ";" separated list of drop down list items.
TabLabel	TabLabel	This would be the name of the custom tab. Notes: Making custom tab's TabLabel the same will cause the all like fields to update when the user enters data. For Type of "Custom" and CustomTabType of "Radio" make the TabLabels the same for any radio buttons

		you want to group together. The Name value will be what is returned for the radio selected in a group.
Value	Value	This element specifies the value of the custom tab. Notes: For Type of "Custom" and CustomTabType of "List" the Value is the default selected item in the drop down list. For Type of "Custom" and CustomTabType of "Radio" or "Checkbox" place a "X" in this field to check the item by default.
CustomTabType	CustomTabType	It tells what is represented by the tabs that have a TabTypeCode of Custom. Available custom tab types are: Text, Checkbox, Radio, List, Date, Number, SSN, ZIP5, ZIP5DASH4, Email. Notes: List will be presented to the signer as a drop down list on the signing document. The values for a List are set in the Name field as ";" separated items. Checkbox cannot be set as a required field. Number, Date, SSN, ZIP5, ZIP5DASH4, and Email fields will be validated for format before the signer can complete the signing. Text, List, Date, Number, SSN, ZIP5, ZIP5DASH4, and Email characters will be limited by the size of the field set with CustomTabWidth and CustomTabHeight.
CustomTabWidth	Integer	Width of the custom tab.
CustomTabHeight	Integer	Height of the custom tab.
CustomTabRequired	Boolean	Signer is required to fill out the custom tab if true.
CustomTabLocked	Boolean	Signer cannot change the data of the custom tab if true.
CustomTabDisableAutoSize	Boolean	Disables the auto sizing of single line text boxes in the signing screen when the signer enters data. If disabled users will

		only be able enter as much data as the text box can hold. By default this is false. This property only affects single line text boxes.
CustomTabListItems	String	Semicolon separated items used in the dropdown list for a CustomTabType of "List". These values may also be specified in the Name node; however, this node will take precedence.
CustomTabListValues	String	For Form Field List Box type this is the semi-colon delimited list of values to go with the displayed list items set in CustomTabListItems or Name. These values are the hidden values of the drop down list display items.
CustomTabListSelectedValue	String	Set this to the default selected item based on the CustomTabListValues. If Value is specified that will take precedence.
CustomTabRadioGroupName	String	This is the group name for a set of radio buttons. The CustomTabType must be "Radio". This may also be specified in the TabLabel node; however, this node will take precedence.
CustomTabValidationPattern	String	For all custom tab types except radio buttons, checkboxes, and list items you may supply a regular expression that will be validated when data is entered in the field. This is optional and if not provided the default DocuSign validation rules will apply. Javascript RegEx object is used for regular expression validation. Regular expressions must be supported by this object to resolve.
CustomTabValidationMessage	String	Message to be displayed to the signer if the validation rule from CustomTabValidationPattern fails. This is optional and if not provided the default DocuSign message will be displayed
TemplateLocked	Boolean	Optional element. Used only when working with template tabs. If true, the attributes of the tab cannot be changed by the sender.
TemplateRequired	Boolean	Optional element. Used only when

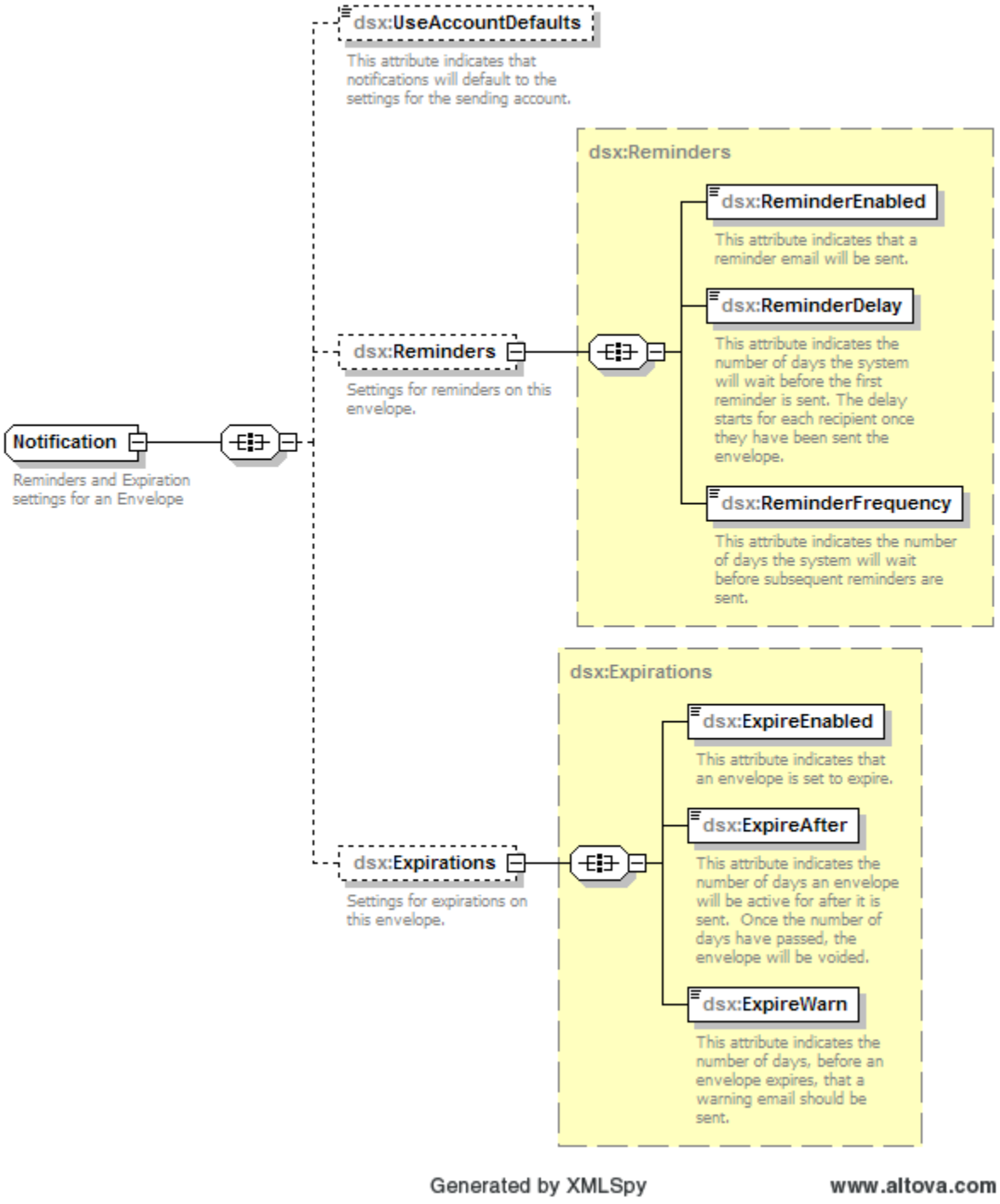
		working with template tabs. If true, the tab cannot be removed by the sender.

Anchor Tab



Name	Schema Type	Description
AnchorTabString	string	Specifies string used as anchor tab in the document.
XOffset	double	This specifies tab location as XOffset position from the anchor tab.
YOffset	double	This specifies tab location as YOffset position from the anchor tab.
Unit	UnitTypeCode	Specifies units of the offset. Unit could be Pixels, Mms, Cms or Inches.
IgnoreIfNotPresent	boolean	Ignore this tab if anchor is not found in the document.

Notification


dsx:UseAccountDefaults
This attribute indicates that notifications will default to the settings for the sending account.
dsx:Reminders
dsx:ReminderEnabled
This attribute indicates that a reminder email will be sent.
dsx:ReminderDelay
This attribute indicates the number of days the system will wait before the first reminder is sent. The delay starts for each recipient once they have been sent the envelope.
dsx:ReminderFrequency
This attribute indicates the number of days the system will wait before subsequent reminders are sent.
dsx:Reminders
Settings for reminders on this envelope.
Notification
Reminders and Expiration settings for an Envelope
dsx:Expirations
dsx:ExpireEnabled
This attribute indicates that an envelope is set to expire.
dsx:ExpireAfter
This attribute indicates the number of days an envelope will be active for after it is sent. Once the number of days have passed, the envelope will be voided.
dsx:ExpireWarn
This attribute indicates the number of days, before an envelope expires, that a warning email should be sent.
dsx:Expirations
Settings for expirations on this envelope.
Generated by XMLSpy
www.altova.com

Name	Schema Type	Description
UseAccountDefaults	boolean	Indicates whether or not the Account defaults for notifications should be used on this envelope.
Reminders	Reminders	ReminderEnabled: Boolean: Whether or not a reminder email will be sent. ReminderDelay: Integer: Number of days AFTER a person receives an envelope, that they will start receiving reminder emails. ReminderFrequency: Integer: Number of days to wait between reminder emails.
Expirations	Expirations	ExpireEnabled: Boolean: Whether or not this envelope is set to expire. ExpireAfter: Integer: Number of days envelope will be active. ExpireWarn: Integer: Number of days before envelope expiration, that a warning email will be sent.

EnvelopeStatus

EnvelopeStatus is the response element for CreateEnvelope and CreateAndSendEnvelope. Please refer to the Envelope Status chapter for the response EnvelopeStatus

Rules for CreateEnvelope and CreateAndSendEnvelope

API user specific rules

- The length of AccountId, document Name should not exceed 100 characters. Document password length should not exceed 50 characters.
- UserName, Email and access code of the recipient should not exceed 100 characters.
- SignatureName in SignatureInfo element should at least have one character. SignatureInitial should have a minimum 1 character and maximum 10 characters.
- Font styles supported for recipient signature are RageItalic, Mistral, BradleyHandITC, KaufmannBT, Freehand575 and LuciaBT.
- Allowed length for State attribute in address information element will be 2 characters. For Zip it will be 5 characters and for ZipPlus4 it will be 4 characters.
- Allowed length for SSN4 will be 4 characters and for SSN9 it will be 9 characters.
- The length of Label and Type for attachment element should not exceed 100 characters.

- CaptiveInfo element is used to specify if the recipient is embedded. A non null ClientUserId implies that the recipient is embedded.

Rules for Exceptions thrown by the API

- AccountId specified while creating an envelope should exist else an exception with error message "Account_Does_Not_Exist_In_System" is thrown.
- DocumentID specified in the tab element should refer to a document in the envelope else exception with error message "Tab_Refers_To_Missing_Document" is thrown.
- RecipientID specified in the tab element should refer to an existing recipients ID attribute. Else exception is thrown with error message "Tab_Refers_To_Missing_Recipient".
- Tab is affixed to a specific page number of the document. So the document specified by DocumentID in the Tab element should have page number specified by PageNumber. Else exception "Tab_Pagenumber_Is_Not_In_Document" is thrown.
- The document element in the CreateEnvelope and CreateAndSendEnvelope should contain the encoded document. Else exception "No_Document_Received" is thrown.
- The Email specified in the recipient element should exist else an exception is thrown with error message "Invalid_Email_Address_For_Recipient".
- Recipient specified by UserName and Email pair in the recipient element must exist. Else exception with error message "Unknown_Envelope_Recipient" is thrown.
- If a signer recipient has any tabs available to them in the envelope they will be required to have at least one signable tab in at least one of the documents. Else exception with error message "Signer_Recipient_Has_No_Signable_Tabs" is thrown. If the signer recipient has no tabs available they will be considered a freeform signer and be required to place the tabs themselves.
- Access code is used to validate recipient identity. User has to enter the access code specified in the recipient element in the CreateEnvelope/CreateAndSendEnvelope to prove his/her identity. Recipient will need to validate his/her identity even if access code is not specified in the recipient element if RequireIDLookup attribute is set to true. But in this case the identity is verified not by entering access code, but by answering a questionnaire correctly. If the user fails to validate his/her identity then an exception is thrown with error message "Recipient_Has_Failed_Security_Check".
- All recipients in an envelope should have a valid routing order else an exception with error message "Recipient_Has_Invalid_RoutingOrder" is thrown.
- There is a provision for creating custom tab. But in case of custom tab both the attributes Name and TabLabel should be specified. If only one of the attribute is specified in the Tab element then an exception "CustomTab_Is_Incomplete" is thrown.
- Account should have API permission to utilize CreateEnvelope and CreateAndSendEnvelope methods. Account needs to have "InSession permission" to send envelopes to InSession recipients. Else an exception with error message "Account_Lacks_Permissions" is thrown.
- AccountID mentioned should not be suspended by DocuSign Administrator; else exception will be thrown with message "Account_Has_Been_Suspended".
- If the recipient Email specified in CreateAndSendEnvelope/CreateEnvelope is already reserved by another then processor throws an exception with error message "Email_Is_Reserved".

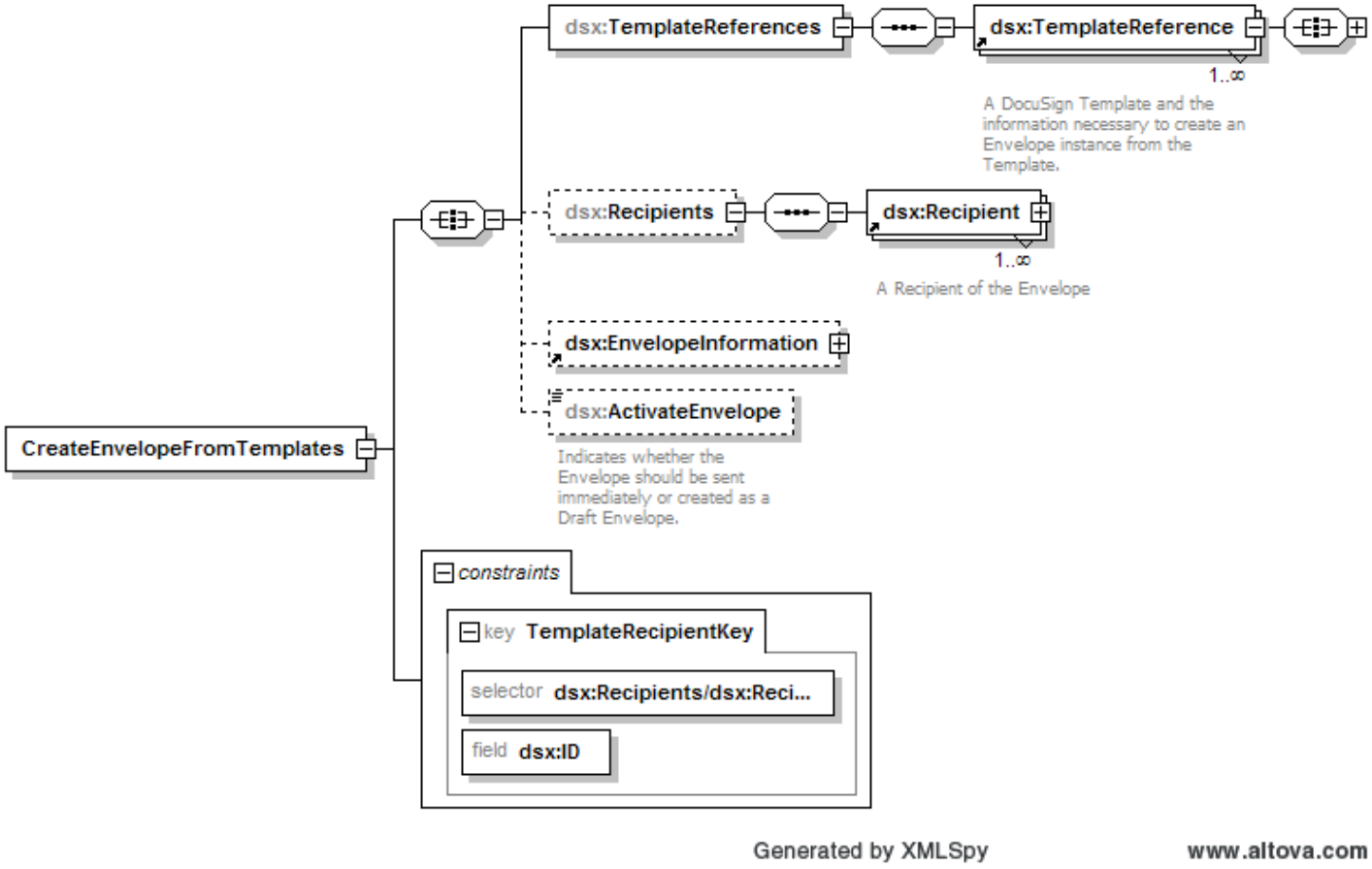
- In CreateEnvelope and CreateAndSend Envelope, the account Id is validated for pricing plan, if the account does not have a pricing plan then "Account_Lacks_Pricing_Plan" exception is thrown
- InSession carbon copy recipients cannot be specified in CreateAndSendEnvelope. If the recipient Type is CC then ClientUserID has to be null else exception "Captive_Carbon_Copy_Recipient_Not_Supported" will be thrown.

CreateEnvelopeFromTemplates

CreateAndSendFromTemplates uses templates as the basis for the definition of a signing ceremony. In contrast to the CreateAndSendEnvelope method, where the relationship of Tabs-to-Documents and Tabs-to-Recipients is passed in the request, CreateAndSendFromTemplates derives its Tab assignments from the pre-configured Template; and provides Recipient information to fulfill the Roles that are specified in the Template.

When provided one or more templates, along with their supplemental Document, Recipient and Envelope information, DocuSign will merge the various Template references into a single Envelope and send to the Recipients.

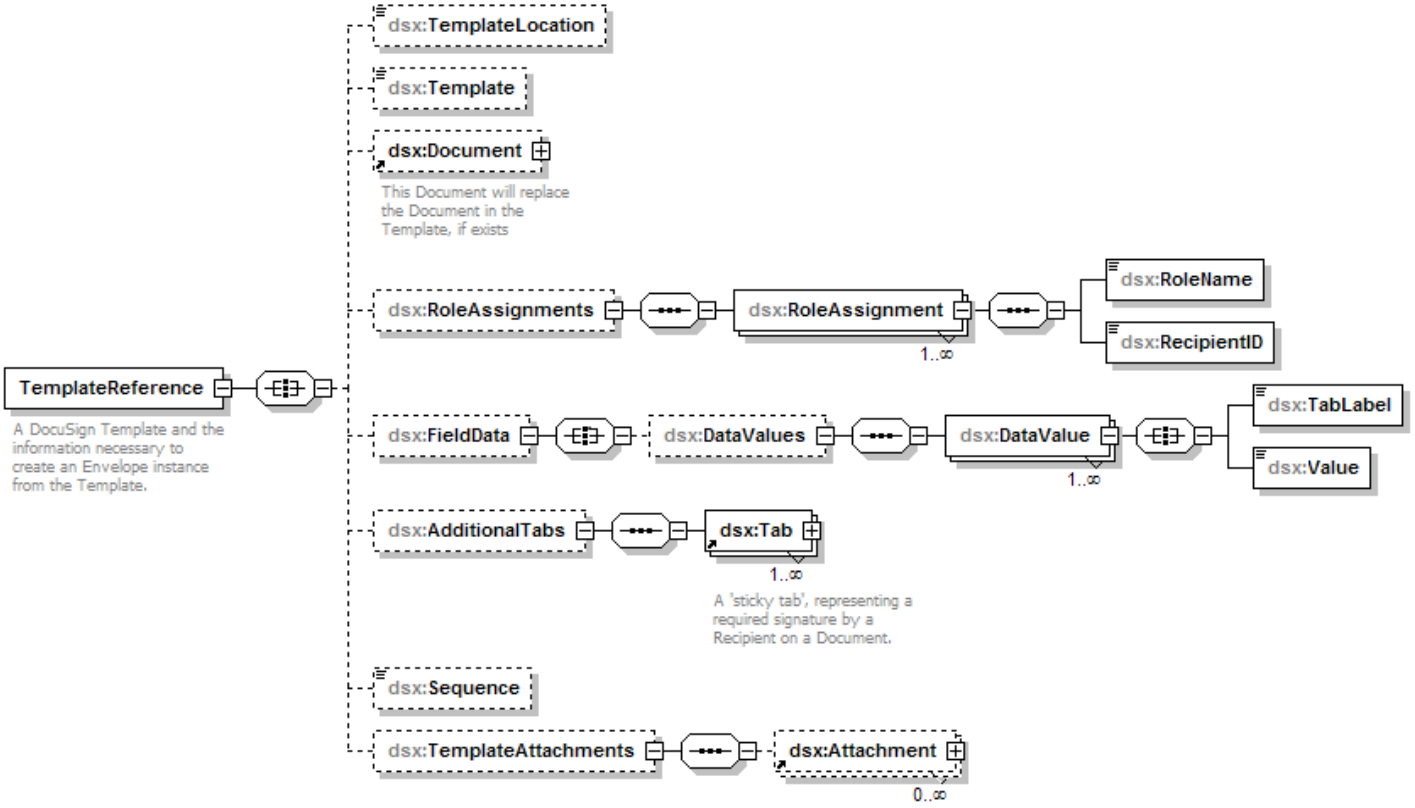
Schema



Name	Schema Type	Description
TemplateReferences	TemplateReference	Specifies the Templates that are included in this call. See the TemplateReference section below for more information.
Recipients	Recipient	Defines the Recipients that are included in this Envelope. These will be referenced from the RoleAssignment elements that are in each TemplateReference. The structure of a Recipient in the CreateAndSendFromTemplates method is similar to Recipients in other methods of the DocuSign Connect API. However, since these properties may also be specified in Templates that are referenced in the request, override rules are in place to determine the order of precedence of values that are specified in multiple places for like fields. For more information on the override rules of Recipient data over the roles specifications in the Template(s), see the override rules below. For Recipient definition refer to the section earlier in this document.
EnvelopeInformation	EnvelopeInformation	The structure and usage of these Envelope-level concepts are similar with their counterparts in other methods of the API. However, since these properties may also be specified in Templates that are referenced in the request, override rules are in place to determine the order of precedence of values that are specified in multiple places for like fields. For more information on the override rules see the EnvelopeInformation section below.
ActivateEnvelope	Boolean	Indicates whether the Envelope should be sent immediately or created as a Draft Envelope. Defaults to "True", where the Envelope will be sent.

TemplateReference

TemplateReferences contains an unbounded array of TemplateReference objects, each a self-contained structure used to satisfy the business rules of a given template, which will be merged into the parent Envelope.

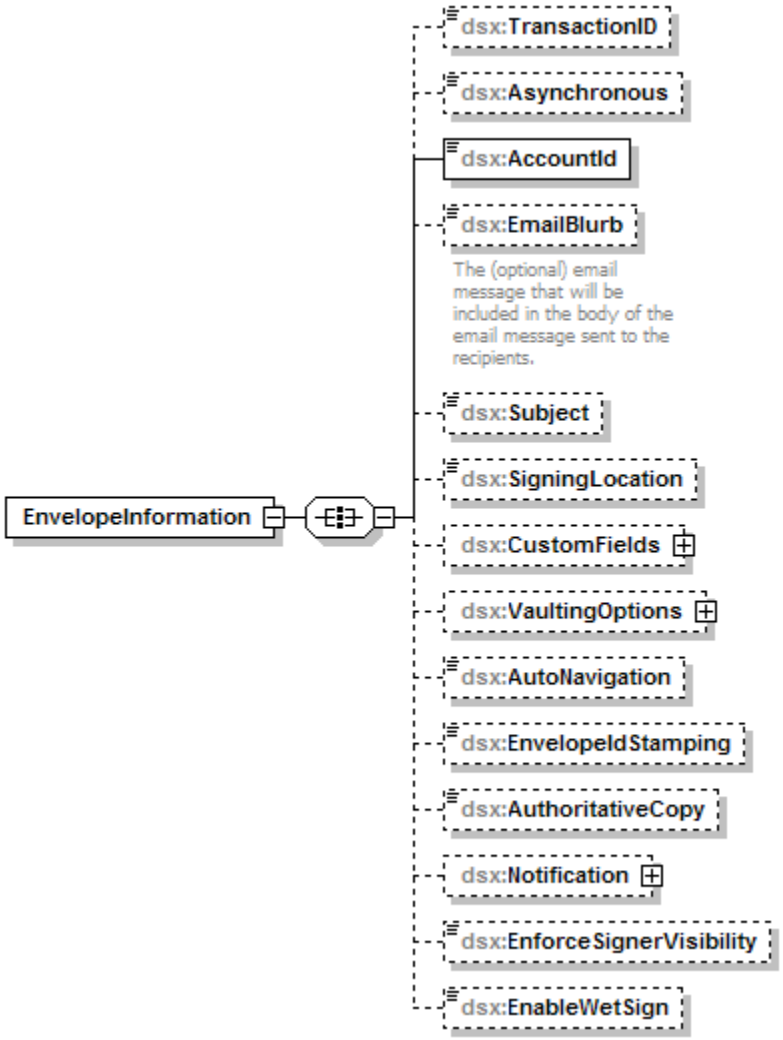

TemplateReference
A DocuSign Template and the information necessary to create an Envelope instance from the Template.
dsx:TemplateLocation
dsx:Template
dsx:Document
This Document will replace the Document in the Template, if exists
dsx:RoleAssignments
dsx:RoleAssignment
1..∞
dsx:RoleName
dsx:RecipientID
dsx:FieldData
dsx:DataValues
dsx:DataValue
1..∞
dsx:TabLabel
dsx:Value
dsx:AdditionalTabs
dsx:Tab
1..∞
A 'sticky tab', representing a required signature by a Recipient on a Document.
dsx:Sequence
dsx:TemplateAttachments
dsx:Attachment
0..∞
Generated by XMLSpy
www.altova.com

Name	Schema Type	Description
TemplateLocation	TemplateLocation	Enumerated list of options that describe where the Template information resides. Available options include: SOAP (default) and PDFMetaData.
Template	String	This element will contain different values, depending on the value of TemplateLocation: "SOAP": Template will include the Template XML in the format created by DocuSign Professional "PDFMetaData": Template can be left blank. It will be disregarded if provided. In this scenario, the Template xml is contained within the xml portion of the PDF document. In either case, the Document pdfBytes can be removed from the Template xml to optimize the payload size.
Document	Document	Refer to Document earlier in this document for details.
RoleAssignments	RoleAssignment	Specifies the RoleAssignments that are included in this call. RoleName – LongString - The name of the Role in the Template to which this Recipient information will be assigned RecipientID – LongString - The Recipient identifier from the Recipients object that will fulfill this Role.
FieldData	FieldData	Provides the rules and/or data values that are used with the Template fields. See the FieldData section below for more information.
AdditionalTabs	Tab	Specifies any Tabs that should be added to a Document. AdditionalTabs enable the request to apply additional Tabs that are not configured in the Template. This would allow for run-time information to be specified as well as the template definitions that are specified at design-time. This could be used with an "open" Template that lacks pre-considered configuration in order to include ad hoc documents to the Envelope. Refer to Tab earlier in this document for details.

Sequence	NonNegativeInteger	Specifies the order that the Document(s) included in the Template should be presented in the Envelope. If not provided, the Documents will be presented in the order of the TemplateReferences.
TemplateAttachments	Attachment	A mechanism to provide the Document object through a SOAP attachment.

EnvelopeInformation

The structure and usage of these Envelope-level concepts are similar with their counterparts in other methods. However, since these properties may also be specified in Templates that are referenced in the request, override rules are in place to determine the order of precedence of values that are specified in multiple places for like fields. Refer to the rules section below for additional information.



All fields in the EnvelopeInformation schema are defined in the Envelope section earlier in this document.

FieldData

FieldData provides information regarding the field data that will be correlated to the fields of the Template

Name	Schema Type	Description
DataValues	DataValue	Provides explicit TabLabel/Calue pairs that should be used for the field values in the Envelope. This value will take precedence over a similarly named PDF field if both are provided.

Rules

CreateAndSendFromTemplates refers to Templates to establish many of the properties of an Envelope. However, the *CreateAndSendFromTemplates* schema also overlaps the Template capabilities, so it is necessary to establish rules of usage and precedence to address incidents of data collision between the values in the SOAP request and the Template(s); or between the Template(s) themselves, since they are logically unrelated to each other. The table below describes the data override rules of the *CreateAndSendFromTemplates* schema.

Section	Element	Description
Recipient	ID	RecipientID is an identifier that is used to correlate the Recipient instance to the RoleAssignment.
Recipient	UserName	If allowable per the Template specification(s), this value will override the corresponding field for any role that this Recipient is assigned to.
Recipient	Email	If allowable per the Template specification(s), this value will override the corresponding field for any role that this Recipient is assigned to.
Recipient	Type	This value cannot override the value specified in the Template(s) for any role that this Recipient is assigned to.
Recipient	AccessCode	If allowable per the Template specification(s), this value will override the corresponding field for any role that this Recipient is assigned to. If conflicting, non-null values are provided in multiple roles to which this Recipient is assigned, an override value must be provided in this structure.
Recipient	RequireIDLookup	If allowable per the Template specifications, this value will override the corresponding field for any role that this Recipient is assigned to. If conflicting values are provided in multiple

		roles to which this Recipient is assigned, the value will resolve to True if it is not specified in this structure.
Recipient	IDCheckConfiguration Name	If allowable per the Template specification(s), this value will override the corresponding field for any role that this Recipient is assigned to. If conflicting, non-null values are provided in multiple roles to which this Recipient is assigned, an override value must be provided in this structure.
Recipient	SignatureInfo .SignatureName .SignatureInitials .FontStyle	If allowable per the Template specification(s), these values will override the corresponding fields for any role that this Recipient is assigned to. If conflicting, non-null values are provided in multiple roles to which this Recipient is assigned, override values must be provided in this structures.
Recipient	CaptiveInfo .ClientUserId	If allowable per the Template specification(s), this value will override the corresponding field for any role that this Recipient is assigned to. If conflicting, non-null values are provided in multiple roles to which this Recipient is assigned, an override value must be provided in this structure.
Recipient	CustomFields	If allowable per the Template specification(s), this value will override a CustomField of the same name for any role that this Recipient is assigned to. If conflicting, non-null values are provided for the same CustomField name in multiple roles to which this Recipient is assigned, an override value must be provided in this structure.
Recipient	RoutingOrder	If allowable per the Template specification(s), this value will override the corresponding field for any role that this Recipient is assigned to. If conflicting, non-zero values are provided in multiple roles to which this Recipient is assigned, an override value must be provided in this structure.
Recipient	IDCheckInformation Input	This information cannot be pre-specified in a template, so the data must be provided in this structure.
Recipient	AutoNavigation	If allowable per the Template specifications, this value will override the corresponding field for any role that this Recipient is assigned to. If conflicting values are provided in multiple

		roles to which this Recipient is assigned, the value will resolve to True if it is not specified in this structure.
Recipient	Attachments	If allowable per the Template specification(s), this value will override an attachment of the same name for any role that this Recipient is assigned to. If values are provided for the same Attachment name in multiple roles to which this Recipient is assigned, an override value must be provided in this structure.
Recipient	Note	If allowable per the Template specification(s), this value will override the corresponding field for any role that this Recipient is assigned to. If conflicting, non-null values are provided in multiple roles to which this Recipient is assigned, an override value must be provided in this structure.
EnvelopeInformation	EmailBlurb	If allowable per the Template specification(s), this value will override the corresponding field if provided in a referenced Template. If conflicting, non-null values are provided in multiple referenced Templates, an override value must be provided in this structure.
EnvelopeInformation	Subject	If allowable per the Template specification(s), this value will override the corresponding field if provided in a referenced Template. If conflicting, non-null values are provided in multiple referenced Templates, an override value must be provided in this structure.
EnvelopeInformation	SigningLocation	If allowable per the Template specification(s), this value will override the corresponding field if provided in a referenced Template. If conflicting, non-null values are provided in multiple referenced Templates, an override value must be provided in this structure.
EnvelopeInformation	CustomFields	If allowable per the Template specification(s), this value will override a CustomField of the same name if provided in a referenced Template. If conflicting, non-null values are provided in multiple referenced Templates for the same CustomField name, an override value must be provided in this structure.
EnvelopeInformation	VaultingOptions	If allowable per the Template

		specification(s), this value will override the corresponding field if provided in a referenced Template. If conflicting, non-null values are provided in multiple referenced Templates, an override value must be provided in this structure.
EnvelopeInformation	AutoNavigation	If allowable per the Template specification(s), this value will override the corresponding field if provided in a referenced Template. If conflicting, non-null values are provided in multiple referenced Templates, the value will resolve to True if it is not specified in this structure.
EnvelopeInformation	EnvelopeIdStamping	If allowable per the Template specification(s), this value will override the corresponding field if provided in a referenced Template. If conflicting, non-null values are provided in multiple referenced Templates, the value will resolve to True if it is not specified in this structure.
EnvelopeInformation	AuthoritativeCopy	If allowable per the Template specification(s), this value will override the corresponding field if provided in a referenced Template. If conflicting, non-null values are provided in multiple referenced Templates, the value will resolve to True if it is not specified in this structure.

Template Execution Rules

The rules below are enforced when sending an Envelope based on a Template specification.

Rule #	Description
1	If a Template role is specified as Optional, remove the signing instructions associated with the role, including all Tabs, if the role is not assigned in the *CreateAndSendFromTemplates* request.
2	If any of the data override rules are violated (indicated by "must" statements in the Data Override descriptions), an exception is returned that describes the violation.

Rules for Mapping Data from PDF Forms

Data will be mapped from the incoming PDF form based on a direct correlation between the field name of the Form and the TabLabel of the DocuSign Tab. Fields will only be mapped if ***TransformPDFFields*** is set to true in the document node. Mapping must be supported for

static XFA-based PDF forms and AcroForms. The order of preference for establishing the form type is:

1. Derived on DocuSign server (transparent to client)
2. Pre-configured in a DocuSign structure (i.e. design-time configuration in a template)
3. Passed in via API request.

Rule #	Description
1	When a PDF form is submitted, the DocuSign mapping component will interrogate the form for any fields and apply the data from those fields to DocuSign Form Fields of the same name.
2	Fields from XFA-based PDF forms will be evaluated by their fully-qualified field name (ex: "form/customer/address/line1[0]")
3	The DocuSign component will map any fields that match (per the requirement above). Any fields that do not have a matching field (i.e. field exists in form but not in Envelope; or field exists in Envelope but not in form) will be disregarded by the mapping component.
4	DocuSign form fields are be limited in scope to the document that they pertain to.
5	Edit rules for the fields must be pre-configured for the DocuSign fields (ex: editable vs. locked, required vs. optional, edit masks). Edit rules will not be derived by the DocuSign mapping component.
6	The following field types are supported for data mapping: text boxes, check boxes, radio buttons, drop-down lists, text areas. Regarding drop-down lists and radio buttons, the data mapping is for the selected value only. The list of available values must be pre-configured in the DocuSign fields.

Error Rules

All errors from CreateAndSendEnvelope and CreateEnvelope apply in addition to the following:

Template_Cannot_Add_Document – error thrown if the template location is set to PDFMetaData and the PDF bytes within the Document node of the template reference are invalid.

Template_Not_Provided – error thrown if the template XML cannot be found in the PDFMetaData or TemplateReference Template node.

Template_Unable_To_Load – error thrown if the template provided cannot be loaded.

Template_Cannot_Add_Document – error thrown if the template and document cannot be loaded.

Template_Unable_To_Load_FieldData – error thrown if the PDF form data exists in the PDF and it cannot be extracted.

Template_Unable_To_Process_FieldData – error thrown if the PDF form data exists and it cannot be mapped properly to the recipients DocuSign Tab data.

Template_Unable_To_Flatten_PDF – error thrown if the PDF form data exists and it cannot be flattened.

Template_RoleSpecified_Does_Not_Exist – error thrown if a recipient is mapped to a role that does not exist in the template.

Template_RecipientID_For_Role_NotFound – error thrown if a recipient ID is mapped to a role and the recipient ID does not exist in the template.

Template_Required_Recip_Not_Satisfied – error thrown if there is a required recipient in the template and no recipient/role mapping satisfied the recipient.

Template_To_Envelope_Error – error thrown if the template provided cannot be transformed to a DocuSign envelope.

Template_Override_EnvelopeInformation_Error – error thrown if any items in the EnvelopeInformation data cannot be applied to the envelope.

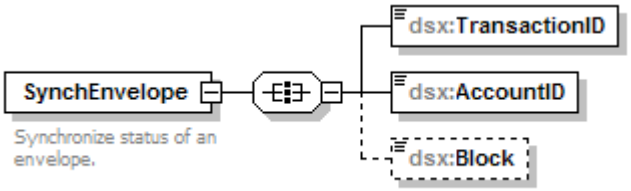
Template_AdditionalTabs_Error – error thrown if an items in the AdditionalTabs data cannot be applied to the envelope.

Template_Override_RecipientData_Error – error thrown if an items in the Recipient data that should override the template data cannot be applied to the envelope.

SynchEnvelope

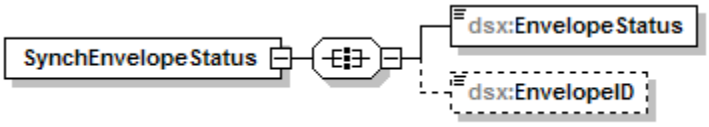
This method is only useful when the 'Asynchronous' flag is set to true on a CreateAndSendEnvelope call. It will determine when the queued envelope has been processed by the system.

Schema



Name	Schema Type	Description
TransactionID	LongString	External ID that is passed in CreateAndSendEnvelope when the 'Asynchronous' flag is set to true.
AccountID	DSXId	Account Id of the user who created the envelope
Block	boolean	If true, this call will wait to return until the queued envelope is done processing. The max time it will wait is 10 minutes.

SynchEnvelopeStatus

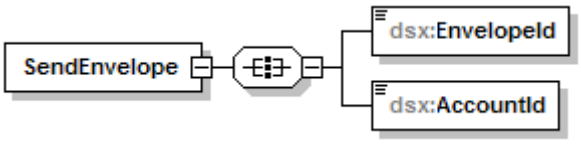


Name	Schema Type	Description
EnvelopeStatus	EnvelopeStatusCode	Status of the envelope at that specific time. Enumeration values could be Voided, Created, Deleted, Sent, Delivered, Signed, Completed, Declined, TimedOut and Processing.
EnvelopeID	LocalId	The unique Id for the envelope.

SendEnvelope

This method is used to send draft envelopes. Refer to CreateAndSendEnvelope method for the sending rules that will be applied before the envelope can be sent.

Schema



Name	Schema Type	Description
EnvelopeId	DSXId	Envelope Id to be sent for signature.
AccountId	DSXId	Account Id of the user who created the envelope

Anchor Based Tagging

This section outlines the usage rules and behaviors of the DocuSign Anchor-Based Tagging feature. Anchor-Based Tagging enables a DocuSign customer to send documents for signature that do not have a fixed layout or format. In these documents, you can not anticipate the absolute location of the tabs when you design your API client application because the tabs must move with text. As an alternative to sending X and Y coordinates for tabs, the DocuSign Signing Service derives anchor tab locations by correlating XML instructions from the SOAP request to data within the document.

Using Anchor Tabs

Anchor tab solutions require a coordinated map between the Anchor data and the document(s) in the CreateEnvelope/CreateAndSendEnvelope SOAP request. When the DocuSign Signing Service receives a request that contains anchor tabs, it searches the document for instances of the AnchorTabString. When found, it places a tab of the specified type for the designated recipient. Tab positions are established by committing the Anchor offset values relative to the lower left point of the AnchorTabString.

When you apply tabs to the document, DocuSign does not remove or replace the AnchorTabString. You can hide codified anchors by using the same font color as the background of the document. While the anchor can be used by DocuSign processes, it will not be visible on the document.

To create an anchor tab:

1. Identify the location in the document by text string. Use either a preexisting text string or add a new one.
   For example, "Borrower's Signature" might already exist in the document. If not, you could add the text string, "BorrowersSignHere".
2. Reference the anchor through the AnchorTabString element of the SOAP request of the CreateEnvelope/CreateAndSendEnvelope request.
3. Determine the offset from the AnchorTabString location to where the tab should be placed. The bottom-left of the AnchorTabString is equivalent to position (0,0), and the bottom-left of the tab graphic is placed relative to that. Positive XOffset values move the tab right on the page and positive Yoffset values move the tab up on the page.
   DocuSign does not currently provide tools to derive the offset values. Determination of the proper offset will likely require some trial-and-error.

Rules for Anchor Tagging

There are both API user specific rules and rules for the exceptions thrown by the processor.

API user specific rules

- The Tab node must contain the RecipientID which refers to an existing recipient and Tab Type (signature, initial etc). Else, the XML validation fails and a Validation error exception is thrown.
- If the Anchor node is initialized, then the AnchorTabString must be specified. Else, the XML validation fails and a Validation error exception is thrown.
- The AnchorTabString specified in the SOAP envelope must be included at least once in the corresponding PDF document of the envelope.
- API users who do not use the anchor-based tabs do not make any changes.
- The user can decide the format of the AnchorTabString. DocuSign tabs are created for each instance of the AnchorTabString within the document, so special care must be taken to establish unique AnchorTabStrings that do not result in unintentional tabs.
- The API user only needs to list the anchor tab in the XML once and map it to a recipient and tab type. The tab is placed for the recipient in each instance that the anchor tab appears in the document(s).
- The API user cannot use the same anchor tab for different recipients for the same document.
- The API user cannot use a text string in an Image PDF.
- The allowed offset units are: pixels, millimeters, centimeters or inches. If not specified, the default unit is pixels.
- X or Y offsets supplied for a tab apply to all instances of the tab in the document.
- To use different offsets at different locations in the document for the same recipient, create multiple, unique anchor tabs.

Rules for Exceptions thrown by the API

- The anchor tab specified by the user must exist in the corresponding document of the envelope. Else, the processor throws an exception with the error message "Anchor_Tab_String_Not_Found". The error message includes the list of anchor tabs that are not found.
- The X and Y offset values must not extend a tab beyond the boundaries of the page. Else, the processor throws an exception with the error message "Invalid_User_Offset". The error message includes the X or Y offset that resulted in the exception.
- The API user must not use the same tab for different recipients for the same document. Nor can the API user use the same tab for different tab types. After the tab is used once, the processor throws an exception with the error "Invalid_Anchor_Tab_String". The error message indicates that the user can not map an anchor tab for different recipients or tab types.

Known Limitations

- The processor cannot search text that is embedded in an image.
- The processor does not support an AnchorTabString embedded in the form of a PDF X-object in the PDF document.

- The processor does not re-flow the text that surrounds the anchor tabs. It is the responsibility of the document author to provide sufficient whitespace to contain the potential width of the ultimate tab value.

Tips and Tricks

The following are tips for effective use of Anchor Based Tagging:

- In order to avoid unintentional conflicts between AnchorTabStrings and the text that naturally exists in documents, establish a codified syntax for AnchorTabStrings that is unlikely to otherwise appear in a document.
- Develop an extensible and consistent syntax that can be used across multiple document types. i.e. [RecipientIdentifier.TabType.TabQualifier]. Examples of this syntax might include [R1.InitialHere] or [CoSign1.Custom.SSN].
- Especially for documents that have variable numbers of tabs or signers (ex: optional Co-signers), author the source document to include hidden anchor tabs for all potential signers/permutations. Then, control the tabs that are actually placed by including/excluding the Anchors in the XML request. This approach allows a single document to be used for all use cases instead of maintaining separate documents for each scenario.

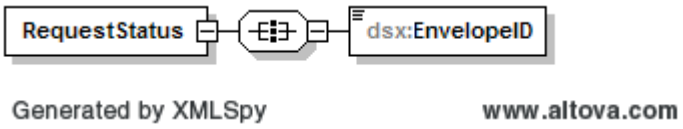
Chapter 3: Envelope Status

The RequestStatus and RequestStatuses methods can be used to query the status of existing envelopes.

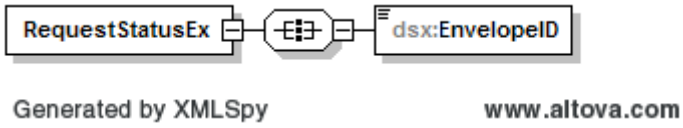
RequestStatus and RequestStatusEx

Schema

RequestStatus



RequestStatusEx



The RequestStatus method takes the element EnvelopeID for the envelope for which the status is requested. This Envelope ID must refer to an envelope created via the Create Envelope/ CreateAndSendEnvelope method call.

Name	Schema Type	Description
EnvelopeID	DSXId	The Envelope ID for which to retrieve status.

Sample Request XML

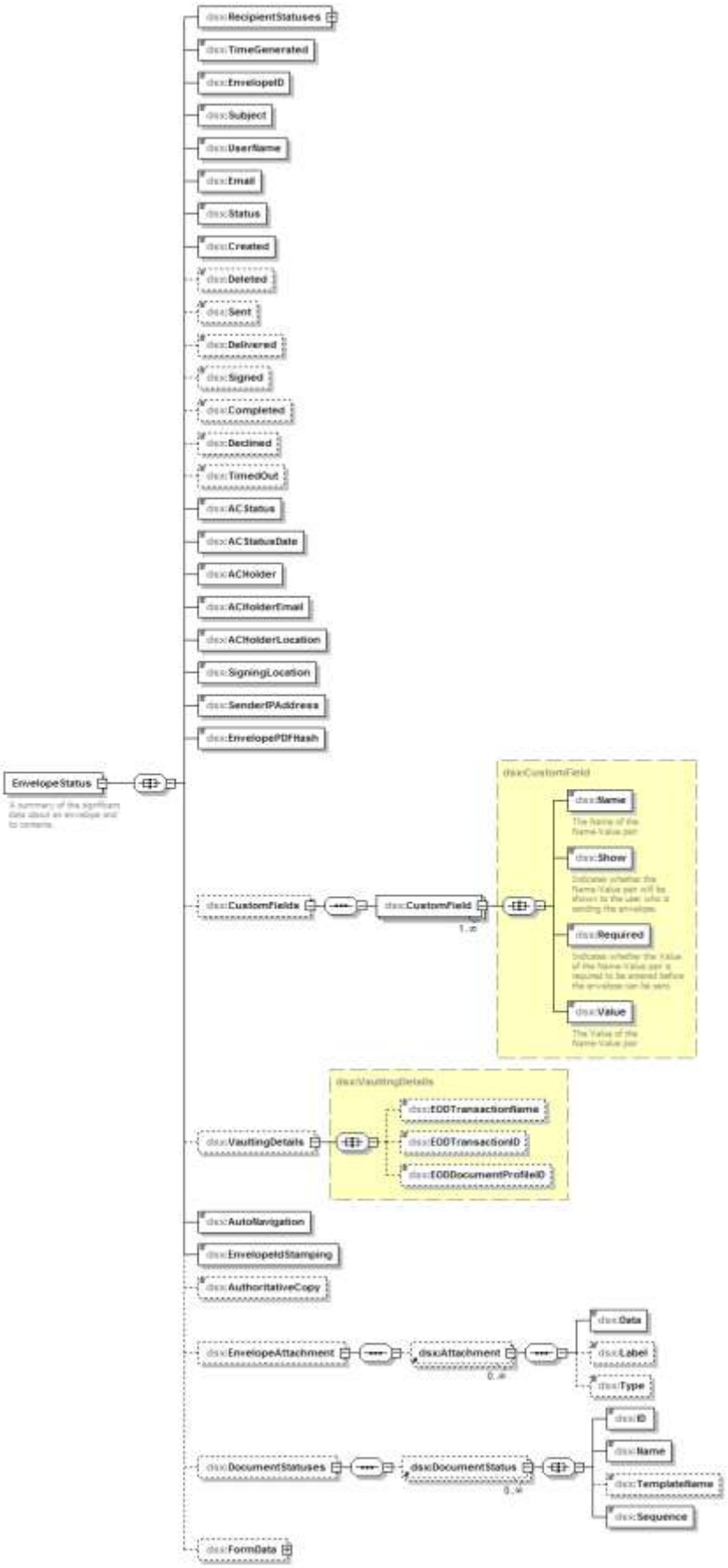
```
SOAPAction: "http://www.docusign.net/API/3.0/RequestStatus"

<?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>
    <RequestStatus xmlns="http://www.docusign.net/API/3.0">
      <EnvelopeID>string</EnvelopeID>
    </RequestStatus>
  </soap:Body>
</soap:Envelope>
```

EnvelopeStatus

Shown below is the schema for envelope status which is the response schema to the RequestStatus.

Schema



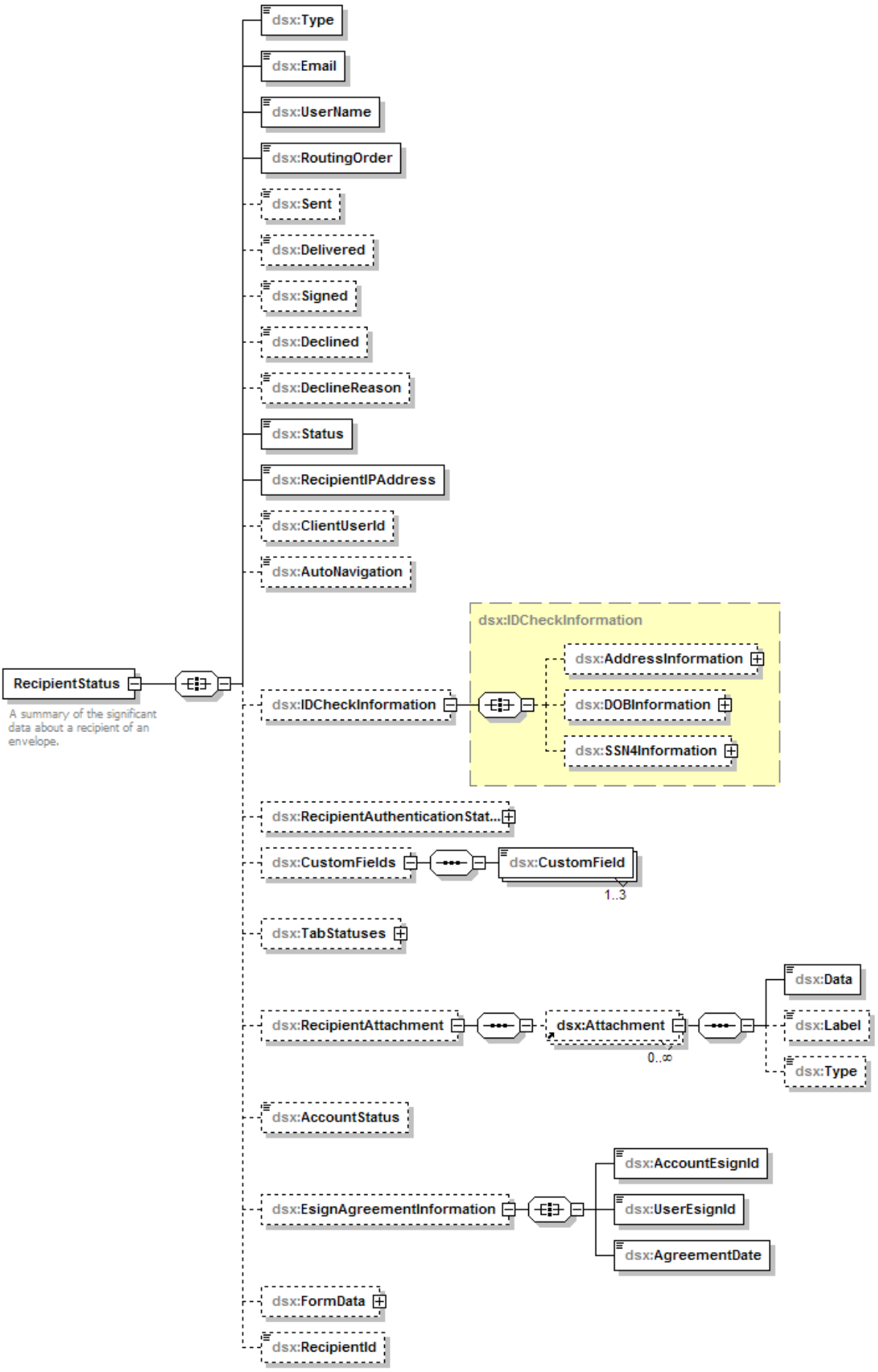
The RequestStatus response contains a snapshot of the status of the envelope at the time the call was issued. It contains the following attributes and elements:

Name	Schema Type	Description
RecipientStatuses	RecipientStatus	Includes the status of each of the envelope recipients. For more information, see the RecipientStatusType section below.
TimeGenerated	dateTime	Specifies the time of creation of the envelope
EnvelopeID	DSXId	The Envelope ID of the envelope who's status is provided.
Subject	string	The envelope subject, as provided by the envelope creator.
UserName	UserName	Contains the user name of the person who created the envelope
Email	Email	Contains the email of the person who created the envelope
Status	EnvelopeStatusCode	Status of the envelope at that specific time. Enumeration values could be Voided, Created, Deleted, Sent, Delivered, Signed, Completed, Declined and TimedOut. The <any> element has been included to enable document author to extend his/her document with elements not specified by the schema.
Created	dateTime	Date and time when the envelope was created.
Deleted	dateTime	Date and time when the envelope was deleted.
Sent	dateTime	Date and time when the envelope was sent.
Delivered	dateTime	Last time the envelope was viewed by a signer.
Signed	dateTime	Last time a signer signed the document.
Completed	dateTime	Date and time when the envelope is completed.
Declined	dateTime	Date and time when the envelope was declined by the recipient.
TimedOut	dateTime	Date and time with the envelope is timed out

ACStatus	string	Returns any one of the following Authoritative copy status; DepositPending, Deposited, DepositedEO or DepositFailed
ACStatusDate	dateTime	The time at with authoritative copy was created
ACHolder	string	Contains the user name of the authoritative copy holder.
ACHolderEmail	LongString	Contains the email of the authoritative copy holder.
ACHolderLocation	string	Contains the location of the authoritative copy holder.
SigningLocation	SigningLocationCode	Specifies the physical location of the signer. Enumeration values are InPerson and Online
SenderIPaddress	string	IP address of the user who sent the envelope
EnvelopePDFHash	string	DocuSign's internal hash of the encrypted document.
CustomField	CustomField	Each custom field defined for the envelope is returned. In addition to the name and value of the field, it will include information as to whether the field was required to be provided. Also it mentions if it is shown in the DocuSign user interface.
VaultingDetails	VaultingDetails	This complex type contains information about EOD Transaction name, transaction ID and document profile ID.
EnvelopeStamping	boolean	Specifies if the envelope stamping feature is enabled or not.
AuthoritativeCopy	boolean	Return value true if the authoritative copy is created else returns false.
EnvelopeAttachment	Attachment	It would be possible to return attachments with envelope. This complex element contains data in base64Binary. It also contains label and type for the attachment.
DocumentStatuses	DocumentStatus	Only returned when calling RequestStatusEx and RequestStatusesEx.

		DocumentStatus includes: ID – this is the ID from the source system. Name – name of the document. TemplateName – template the document originated from if the envelope was created using CreateEnvelopeFromTemplates. Sequence – order in which the document appears in the envelope.
FormData	FormData	Reserved for future use. Will only be returned with the GetStatusInDocuSignConnectFormat API. This will return all the DocuSign Secure Fields and DocuSign PowerForms fields as XML.

RecipientStatus


RecipientStatus
A summary of the significant data about a recipient of an envelope.
dsx:Type
dsx:Email
dsx:UserName
dsx:RoutingOrder
dsx:Sent
dsx:Delivered
dsx:Signed
dsx:Declined
dsx:DeclineReason
dsx:Status
dsx:RecipientIPAddress
dsx:ClientUserId
dsx:AutoNavigation
dsx:IDCheckInformation
dsx:IDCheckInformation
dsx:AddressInformation
dsx:DOBInformation
dsx:SSN4Information
dsx:RecipientAuthenticationStat...
dsx:CustomFields
dsx:CustomField
1..3
dsx:TabStatuses
dsx:RecipientAttachment
dsx:Attachment
0..∞
dsx:Data
dsx:Label
dsx:Type
dsx:AccountStatus
dsx:EsignAgreementInformation
dsx:AccountEsignId
dsx:UserEsignId
dsx:AgreementDate
dsx:FormData
dsx:RecipientId

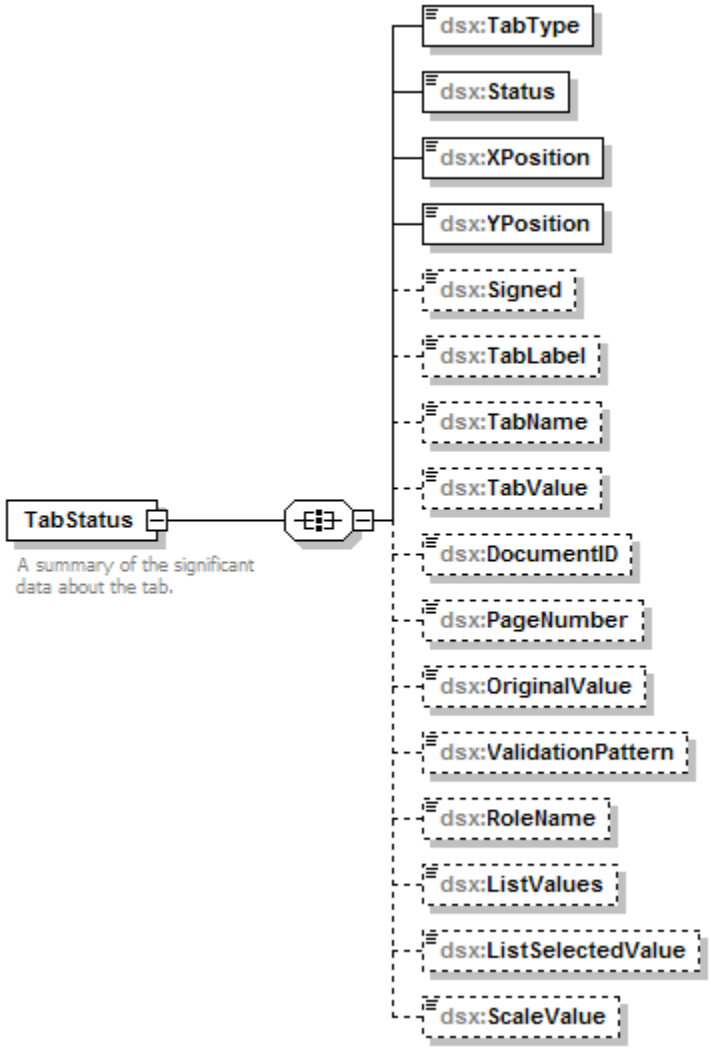
Name	Schema Type	Description
Type	RecipientTypeCode	This element controlled by enumerator tells the role of the recipient. It could be Signer, CarbonCopy, CertifiedDelivery or InPersonSigner. For InPersonSigner: The Email element should be that of the "Signing Host" (the person who will receive the email and assist the signer with the in-person signing process. The UserName element should be that of the "Signing Host" (the Email and UserName combination must match an active DocuSign user).
Email	Email	The recipient's email address.
UserName	UserName	User name of recipient.
RoutingOrder	PositiveShort	Routing order of the recipient in the envelope.
Sent	dateTime	Date and time when the envelope was sent.
Delivered	dateTime	Date and time when the envelope was viewed by the recipient.
Signed	dateTime	Date and time when the envelope was signed by the recipient.
Declined	dateTime	Date and time when the envelope was declined by the recipient.
DeclineReason	string	The reason given by the recipient while declining the envelope.
Status	RecipientStatusCode	Specifies the status of the recipient at the time of request. It could be Created, Sent, Delivered, Signed, Declined, Completed or FaxPending.
Recipient IPAddress	string	The IP address of the recipient's most recent access of this envelope.
ClientUserID	LongString	Tells if the recipient is captive or

		not. If the ClientUserId is not null then the recipient is captive.
CustomField	LongString	Each custom field defined for the envelope is returned. In addition to the name and value of the field, it will include information as to whether the field was required to be provided. Also it mentions if it is shown in the DocuSign user interface.
AutoNavigation	boolean	If returns true then autonavigation feature is enabled for the recipient.
IDCheckInformation	IDCheckInformationInput	Returns the following information about the recipient; Address (i.e. Street1, Street2, City, State, Zip and ZipPlus4), DOB and last four digits of SSN.
RecipientAuthenticationStatus	AuthenticationStatus	Returns the details about recipient authentication. It includes information regarding the IdCheck, Access Code and IDLookUp results. Tells if the status of the above check is passed or failed and also does a timestamp.
TabStatuses	TabStatus	Returns the status of the tabs. For details see below.
RecipientAttachment	*Attachment*	This complex element contains data in base64Binary. It also contains label and type for the attachment.
AccountStatus	string	Only returned when calling RequestStatusEx and RequestStatusesEx. Returns the status of the recipient's account.
EsignAgreementInformation	EsignAgreementInformation	Only returned when calling RequestStatusEx and RequestStatusesEx. Only returned if the recipient agreed. Returns the agreement information of the recipient. Returned items: AccountEsignID – account agreement ID. AgreementDate – date accepted. UserEsignID – user agreement ID.

FormData	FormData	Reserved for future use. Will only be returned with the GetStatusInDocuSignConnectFormat API. This will return all the DocuSign Secure Fields for the recipient.
RecipientId	string	Only returned when calling RequestStatusEx and RequestStatusesEx. Returns the DocuSign recipient Id mapped to the recipient.

TabStatus

The TabStatus element contains information on tabs in the document.



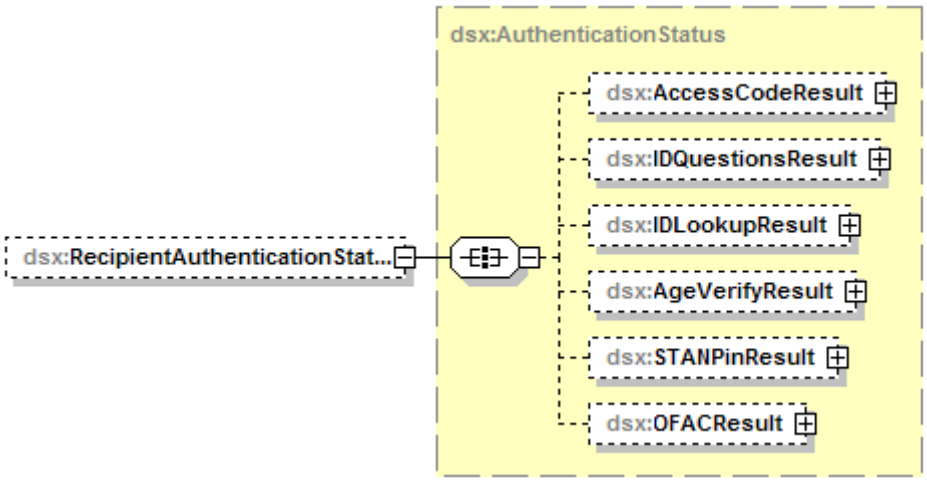
Name	Schema Type	Description

TabType	TabTypeCode	Returns the tab type. This element is enumerator controlled and can have following values; InitialHere, SignHere, FullName, Company, Title, DateSigned, InitialHereOptional, EnvelopeID, Custom, SignerAttachment and SignHereOptional.
Status	string	The status of individual tabs. It can have the enumeration values; Active, Signed, Declined and N/A
XPosition/YPosition	double	X position and Y position of the field. The X and Y positions originate at the upper left of the page and indicate the bottom left-most point of the field. For text fields, this is the upper left of the first letter of the text string. For signature fields, this is the upper left of the first letter of the signature stamp and does NOT include signature stamp image decoration which can surround the signature text.
Signed	dateTime	Date and time when the document was signed by the recipient.
TabLabel	string	This would be the label of the custom tab.
TabName	string	This would be the name of the custom tab.
TabValue	string	This element contains the value of the custom tab.
DocumentID	PositiveInteger	Only returned when calling RequestStatusEx and RequestStatusesEx. Returns the originating document ID.
PageNumber	PositiveInteger	Only returned when calling RequestStatusEx and RequestStatusesEx. Returns the page number the tab is on.
OriginalValue	string	Only returned when calling RequestStatusEx and RequestStatusesEx. Original value before signing.
ValidationPattern	string	Only returned when calling RequestStatusEx and RequestStatusesEx. Validation pattern that was originally passed in, if any.
RoleName	string	Only returned when calling RequestStatusEx and RequestStatusesEx. Only returned when the envelope originated from CreateEnvelopeFromTemplates. RoleName that this tab is associated with in the originating template.
ListValues	string	Only returned when calling RequestStatusEx and RequestStatusesEx. Returns the list values passed in when the envelope was created.
ListSelectedValue	String	Only returned when calling RequestStatusEx and RequestStatusesEx. Returns the selected value from

		ListValues.
ScaleValue	decimal	Represents a decimal value of 0.0 to 1.0 position of the webcontrol scaling slider. A value of 1.0 represents full size.

AuthenticationStatus

The AuthenticationStatus element contains information on the recipient authentication status.



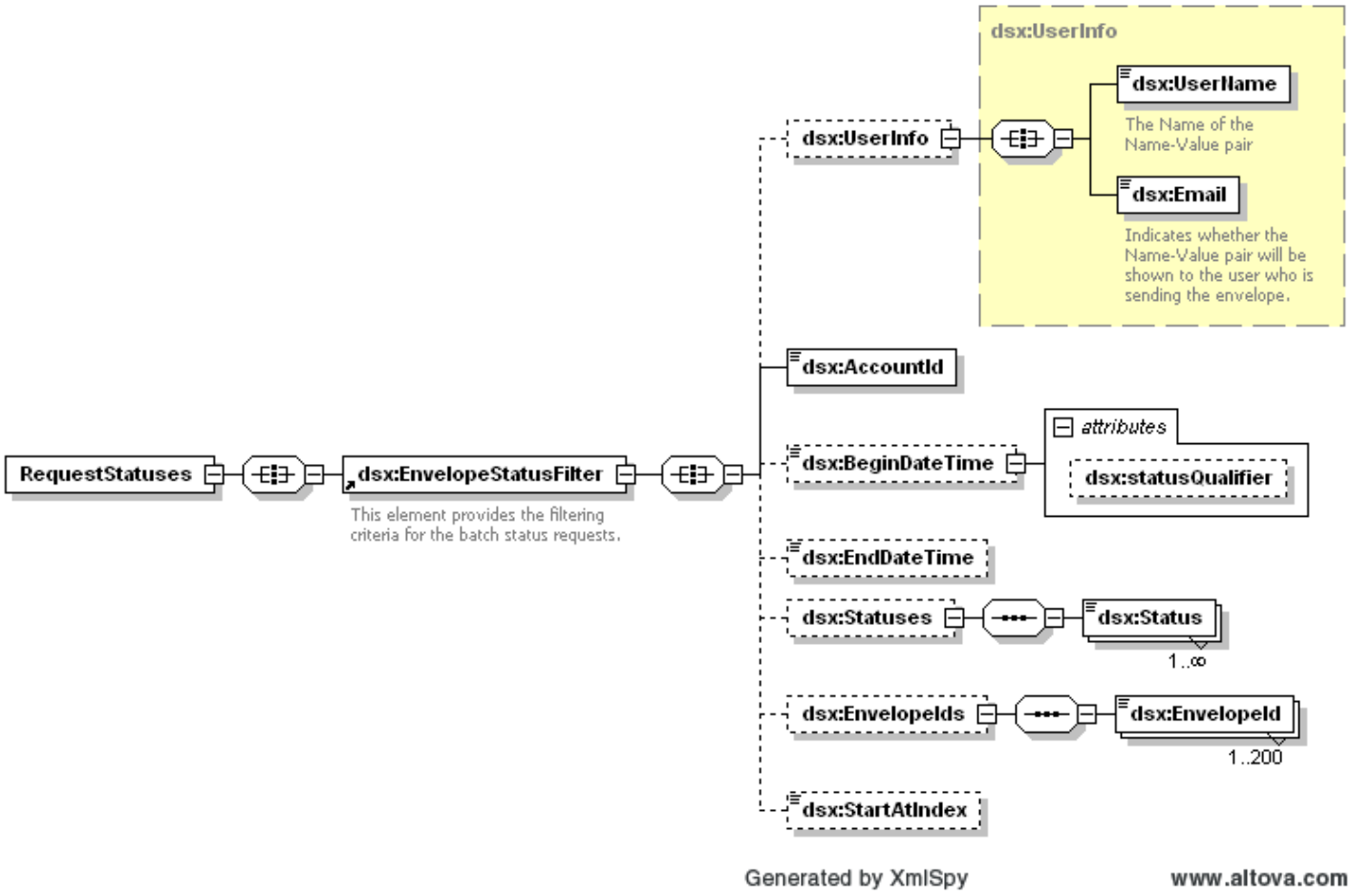
Name	Schema Type	Description
AccessCodeResult	EventResult	Returns the status and time of the access.
IDQuestionsResult	EventResult	Returns the status and time of the ID check questions.
IDLookupResult	EventResult	Returns the status and time of when the recipients was located in the ID check system.
AgeVerifyResult	EventResult	Returns the status and time of the age verification check.
STANPinResult	EventResult	Returns the status and time from the Student Authentication Network (STAN) check.
OFACResult	EventResult	Returns the status and time from the Office of Foreign Asset Control (OFAC) check.

RequestStatuses And RequestStatusesEx

This method is used to request the status of multiple envelopes in a single call. Up to 200 envelopes can be retrieved within a single call. If more results are available, subsequent calls can be used to retrieve the next 200, etc.

Schema

RequestStatuses schema:



RequestStatusesEx schema (EnvelopeStatusFilter is the same as above):

Name	Schema Type	Description
UserInfo	UserInfo	This would be the username Email pair of the person whose envelope details are requested.
AccountId	DSXId	Account Id for which envelope status is requested.
BeginDateTime	dateTime	Specifies the start date of the date range for which envelopes are requested. Has a status qualifier attribute which specifies envelope with what status needs to be requested.

EndDateTime	dateTime	Specifies the end date of the date range for which envelopes are requested.
Statuses	EnvelopeStatusCode	Specifies the status of the envelope at the time of request. This is enumerator controlled.
EnvelopeIds	DSXId	Specifies the envelope Ids of envelopes for which status needs to be known.
StartAtIndex	nonNegativeInteger	If envelope status result sets exceed 200 envelopes, this element can be used to specify that the method should return envelopes at the specified index. The first index is 0, and should be used in the first call to RequestStatuses. Typically this number is a multiple of 200.

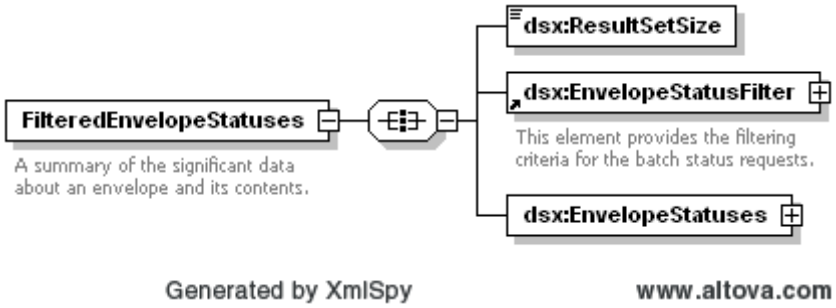
Sample Request XML

```
SOAPAction: "http://www.docusign.net/API/3.0/RequestStatuses"

<?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>
    <RequestStatuses xmlns="http://www.docusign.net/API/3.0">
      <EnvelopeStatusFilter>
        <UserInfo>
          <UserName>string</UserName>
          <Email>string</Email>
        </UserInfo>
        <AccountId>string</AccountId>
        <BeginDateTime d5p1:statusQualifier="string" xmlns:d5p1="http://www.docusign.net/API/3.0" />
        <EndDateTime>dateTime</EndDateTime>
        <Statuses>
          <Status>Any or Voided or Created or Deleted or Sent or Delivered or Signed or Completed or Declined or
TimedOut</Status>
          <Status>Any or Voided or Created or Deleted or Sent or Delivered or Signed or Completed or Declined or
TimedOut</Status>
        </Statuses>
        <EnvelopeIds>
          <EnvelopeId>string</EnvelopeId>
          <EnvelopeId>string</EnvelopeId>
        </EnvelopeIds>
        <StartAtIndex>nonNegativeInteger</StartAtIndex>
        <ACStatus>Unknown or Original or Transferred or AuthoritativeCopy or AuthoritativeCopyExportPending or
AuthoritativeCopyExported or DepositPending or Deposited or DepositedEO or DepositFailed</ACStatus>
      </EnvelopeStatusFilter>
    </RequestStatuses>
  </soap:Body>
</soap:Envelope>
```

FilteredEnvelopeStatuses

Schema



Name	Schema Type	Description
ResultSetSize	int	This element contains the total number of envelopes in the total result set. If this number exceeds 200 then at most 200 results is returned in this request.
EnvelopeStatusFilter	EnvelopeStatusFilter	This element contains the filter criteria exactly as passed in the request. For details please refer above
EnvelopeStatus	EnvelopeStatus	A EnvelopeStatus element (exactly as defined by the EnvelopeStatus method response above) is returned for each of the envelopes in the result set, up to a maximum size of 200 elements.

Rules for RequestStatus, RequestStatuses, RequestStatusEx, RequestStatusesEx, EnvelopeStatus and FilteredEnvelopeStatuses

API user specific rules

- The EnvelopeID, UserName and Email specified in the RequestStatus, EnvelopeStatus RecipientStatus, and RequestStatuses methods shall not exceed 100 characters.
- AccountID specified in RequestStatuses should not exceed 100 characters.

Rules for Exceptions thrown by the API

- An envelope having the specified Envelop*eID* must exist in the system. Else, an exception with the error message "Envelope_Does_Not_Exist" is thrown.
- To receive the envelope status the envelope should not be void. Else an exception with error "Envelope_Has_Been_Voided" is thrown.
- In RequestStatus method User defined by UserName and Email pair should have the permission to request for envelope status. Only sender recipient or a user in the same account as sender with account wide permission will be able to request for the status. Else an exception is thrown with error message "User_Lacks_Permissions".
- In the RequestStatuses method, user defined by UserName and Email pair should belong to the account specified by AccountID. Else an exception with error message "User_Does_Not_Belong_To_Specified_Account" thrown.

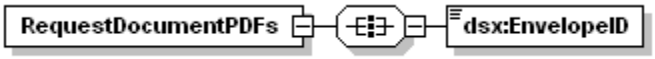
- In the RequestStatuses method the envelopes for which the status is requested should owned by user in the specified AccountID. Else an exception is thrown with error message “Account_Not_Authorized_For_Envelope”.

Chapter 4: Retrieving Documents

This chapter describes the methods which can be used to retrieve envelope documents from the DocuSign system. DocuSign maintains a separate certificate document with each envelope document that contains important origination and landmark events pertinent to the document. RequestDocumentsPDFs and RequestPDF methods are used to request documents. DocumentPDFs and EnvelopePDF are the response methods

RequestDocumentPDFs

Schema



Name	Schema Type	Description
EnvelopeID	DSXId	This is the envelope ID of the envelope, as returned by the CreateEnvelope/CreateAndSendEnvelope. Only documents within the specified envelope are returned.

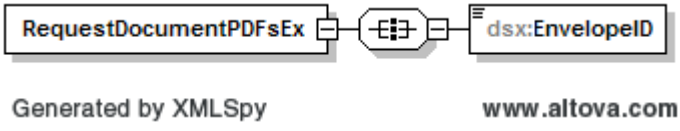
Sample Request XML

```
SOAPAction: "http://www.docusign.net/API/3.0/RequestDocumentPDFs"

<?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>
    <RequestDocumentPDFs xmlns="http://www.docusign.net/API/3.0">
      <EnvelopeID>string</EnvelopeID>
    </RequestDocumentPDFs>
  </soap:Body>
</soap:Envelope>
```

RequestDocumentPDFsEx

Schema



Name	Schema Type	Description

EnvelopeID	DSXId	This is the envelope ID of the envelope, as returned by the CreateEnvelope/CreateAndSendEnvelope. Only documents within the specified envelope are returned.

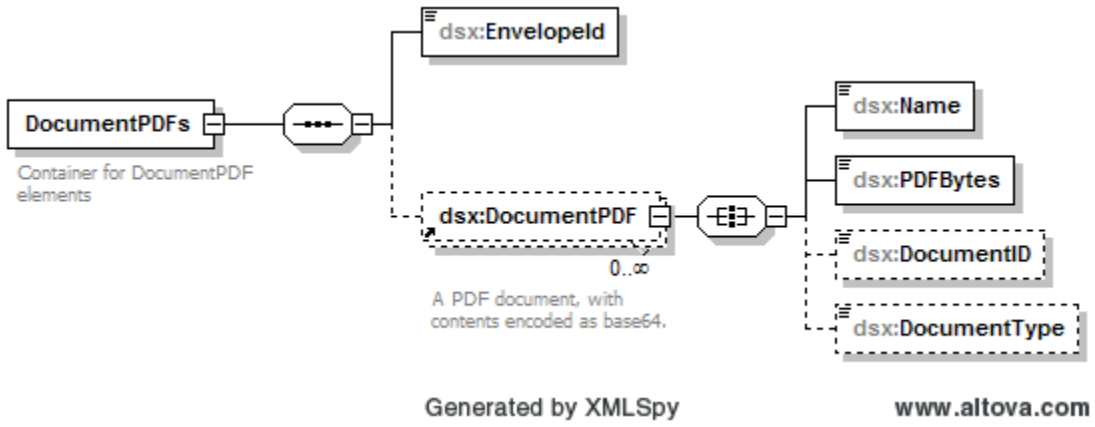
Sample Request XML

SOAPAction: "http://www.docusign.net/API/3.0/RequestDocumentPDFs"

```
<?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>
    <RequestDocumentPDFsEx xmlns="http://www.docusign.net/API/3.0">
      <EnvelopeID>string</EnvelopeID>
    </RequestDocumentPDFsEx>
  </soap:Body>
</soap:Envelope>
```

DocumentPDFs

Schema



Name	Schema Type	Description
EnvelopeID	DSXId	Returns EnvelopeID of the envelope.
DocumentPDF	DocumentPDF	This complex type contains: Name - name of the document. PDFBytes – the document byte stream. DocumentID – the originating document ID. Only returned when calling RequestDocumentPDFsEx. DocumentType – type of the document. Only returned when calling RequestDocumentPDFsEx.

RequestPDF

This method returns all of the documents combined into a single, contiguous PDF. Additionally it returns the Signing Certificate PDF document, which details the specific attributes of the participants and landmark events of the signing transaction.

Schema

Name	Schema Type	Description
EnvelopeID	DSXId	This is the envelope ID of the envelope, as returned by the CreateEnvelope/CreateAndSendEnvelope.

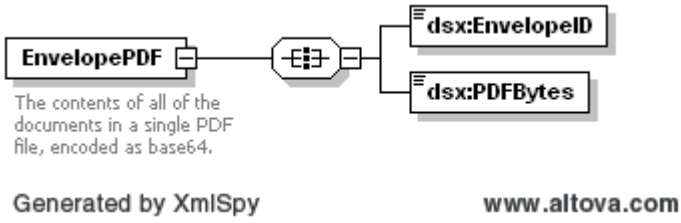
Sample Request XML

```
SOAPAction: "http://www.docusign.net/API/3.0/RequestPDF"

<?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>
    <RequestPDF xmlns="http://www.docusign.net/API/3.0">
      <EnvelopeID>string</EnvelopeID>
    </RequestPDF>
  </soap:Body>
</soap:Envelope>
```

EnvelopePDF

Schema



Name	Schema Type	Description
EnvelopeID	DSXId	Returns EnvelopeID of the envelope.
PDF	base64Binary	The document bytestream

Usage Rules

- In the methods mentioned in this chapter, length of the EnvelopeID must not exceed 100 characters. Else, the XML validation fails and the processor throws a Validation error.
- In the request methods envelope with specified Id must exist. Else an exception "Envelope_Does_Not_Exist" is thrown.
- In the DocumentPDF method returns all the PDF documents in the envelope as base64binary with document names. But DocumentPDF element is optional. If the envelope with mentioned EnvelopeID is in the draft stage then Name or PDFBytes is not returned.
- DocumentPDFs also return certificate document with PDF documents in the envelope
- Only sender or recipient can retrieve the documents else exception, "User_Not_Envelope_Sender_Or_Recipient" will be thrown.
- You may have your account setup by DocuSign to include DocuSign PDF meta data in the returned PDF bytes.

  The envelope level PDF will contain:

  Tracking information that contains the Envelope ID.

  EnvelopeStatus XML.

  The document level PDF will contain:

  Tracking information that contains the Envelope ID and originating Document ID.

  EnvelopeStatus XML with the tab information removed that does not pertain to the document.

  Document PDF form data if the document originated with TransformPDFFields enabled.

  The envelope summary document will contain:

  Tracking information that contains the Envelope ID.

  EnvelopeStatus XML with all tab information removed.

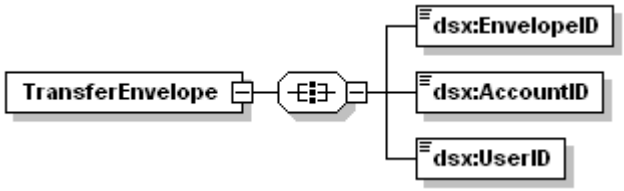
Chapter 5: Transferring Envelopes

This chapter describes the TransferEnvelope method, which envelope owners can use to transfer ownership of existing, completed envelopes to other DocuSign system members. TransferEnvelopeStatus is the response method.

TransferEnvelope

The TransferEnvelope method effectively transfers all documents in the specified envelope to the new owner.

Schema



Name	Schema Type	Description
EnvelopeID	DSXId	This is the envelope ID of the envelope, as specified by the request.
AccountID	DSXId	This specifies the account ID of the person to whom ownership is transferred.
UserID	DSXId	The user ID of the person to whom the envelope ownership is being transferred.

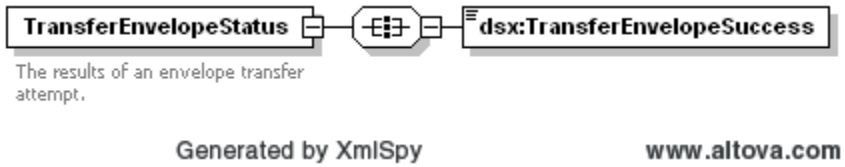
Sampe Request XML

```
SOAPAction: "http://www.docusign.net/API/3.0/TransferEnvelope"

<?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>
    <TransferEnvelope xmlns="http://www.docusign.net/API/3.0">
      <EnvelopeID>string</EnvelopeID>
      <AccountID>string</AccountID>
      <UserID>string</UserID>
    </TransferEnvelope>
  </soap:Body>
</soap:Envelope>
```

TransferEnvelopeStatus

Schema



Name	Schema Type	Description
TransferEnvelopeSuccess	boolean	This element is true if the transfer succeeded.

Usage Rules

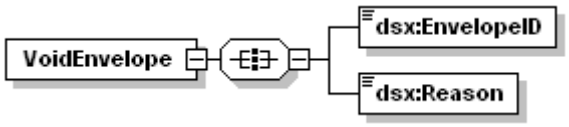
- The length of any of the EnvelopeID, AccountID, and UserID must not exceed 100 characters. Else, the XML validation fails and the processor throws a Validation error.
- An envelope having the specified Envelop*eID* must exist in the system. Else, an exception with the error message "Envelope_Does_Not_Exist" is thrown.
- AccountID must exist in the system; else exception "Account_Does_Not_Exist_In_System" will be thrown.
- UserID must exist in the system; else exception "User_Does_Not_Exist_In_System" will be thrown.
- Transfer of envelope will be allowed only if the envelope status is complete. Else "Envelope_Is_Incomplete" will be thrown.
- Only sender would be able to transfer the ownership to someone else. Else error "User_Not_Envelope_Sender" will be thrown
- User cannot transfer the ownership of the envelope to self. If a user tries to transfer envelope to self "Envelope_Transferee_Already_Owns_Envelope" exception will be thrown.
- If Authoritative Copy status is set then envelope cannot be transferred. It tried to transfer the resulting error will be "Envelope_Cannot_Transfer_Invalid_ACStatus".

Chapter 6: Voiding Envelopes

This chapter describes the methods available to void envelopes. Only incomplete envelopes can be voided. VoidEnvelope method is used to void an envelope and VoidEnvelopeStatus is the response method.

VoidEnvelope

Schema



Name	Schema Type	Description
EnvelopeID	DSXId	This is the envelope ID of the envelope which needs to be voided.
Reason	Reason	The reason for voiding the envelope. Envelope recipients, when notified of the void, are shown this reason.

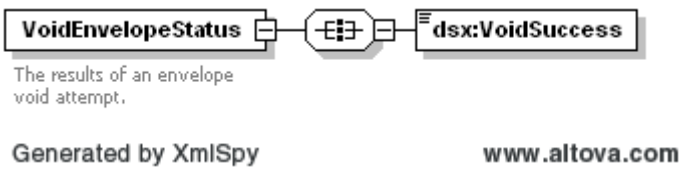
Sample Request XML

```
SOAPAction: "http://www.docusign.net/API/3.0/VoidEnvelope"

<?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>
    <VoidEnvelope xmlns="http://www.docusign.net/API/3.0">
      <EnvelopeID>string</EnvelopeID>
      <Reason>string</Reason>
    </VoidEnvelope>
  </soap:Body>
</soap:Envelope>
```

VoidEnvelopeStatus

Schema



Name	Schema Type	Description

VoidSuccess	boolean	Set to true if the void operation succeeded.

Usage Rules

- The length of any of the EnvelopeID must not exceed 100 characters. Else, the XML validation fails and the processor throws a Validation error.
- The length of reason should not exceed 200 characters and should have a minimum of 1 character.
- EnvelopeID specified in the VoidEnvelope request should exist else an exception is thrown with error message "Envelope_Does_Not_Exist" .
- Only envelopes in the 'Sent' or 'Delivered' states may be voided. Else "Envelope_Cannot_Void_Invalid_State" exception will be thrown.
- Only the sender of envelope can void the envelope. If a user other than sender tries to void an envelope then exception "User_Not_Envelope_Sender" will be thrown.

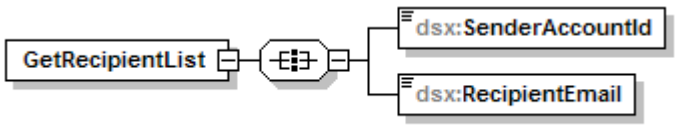
Chapter 7: Recipients

This chapter describes the method to retrieve those recipients which have created signing accounts.

GetRecipientList

Senders can use this method to determine which recipients are available at the given email address.

Schema



Name	Schema Type	Description
SenderAccountId	DSXId	Sender's account Id
RecipientEmail	Email	Recipient's email address.

Sample Request XML

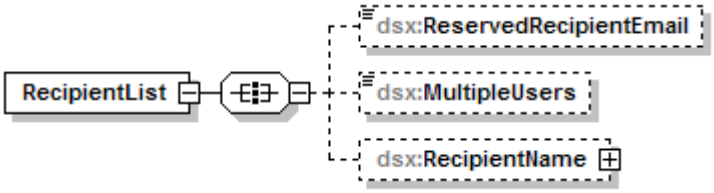
```
SOAPAction: "http://www.docusign.net/API/3.0/GetRecipientList"

<?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>
    <GetRecipientList xmlns="http://www.docusign.net/API/3.0">
      <SenderAccountId>string</SenderAccountId>
      <RecipientEmail>string</RecipientEmail>
    </GetRecipientList>
  </soap:Body>
</soap:Envelope>
```

RecipientList

This element returns a list of RecipientRecords for a designated recipient email. For each record, the return Boolean value indicates whether or not an Esign agreement exists between the sender and that particular recipient.

Schema



Name	Schema Type	Description
RecipientName	LongString	List of recipient Signature Names available to sign at this email.
ReservedRecipientEmail	boolean	Whether or not the user has reserved this email to disallow new recipient names at this email address.
MultipleUsers	boolean	If true, more than one physical person is using this email address to sign with.

Usage Rules

- The length of SenderAccountId and RecipientEmail must not exceed 100 characters.
- If no active signatures are available at the given email address, then no records will be returned.

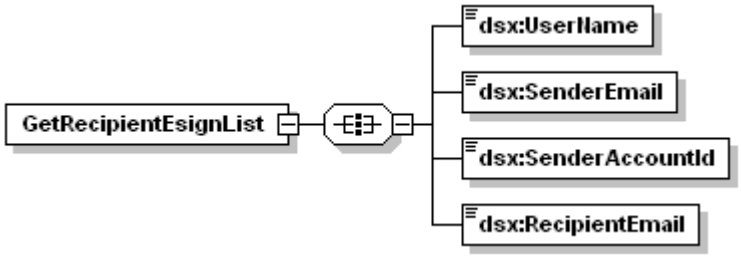
Chapter 8: ESign Agreements

This chapter describes the method to retrieve those recipients which have accepted an ESign agreement with the invoker's account.

GetRecipientESignList

Senders can use this method to determine if an ESign agreement preexists between the sender and recipient.

Schema



Name	Schema Type	Description
UserName	UserName	Sender's name.
SenderEmail	Email	Sender's email address.
SenderAccountId	DSXId	Sender's account Id
RecipientEmail	Email	Recipient's email address.

Sample Request XML

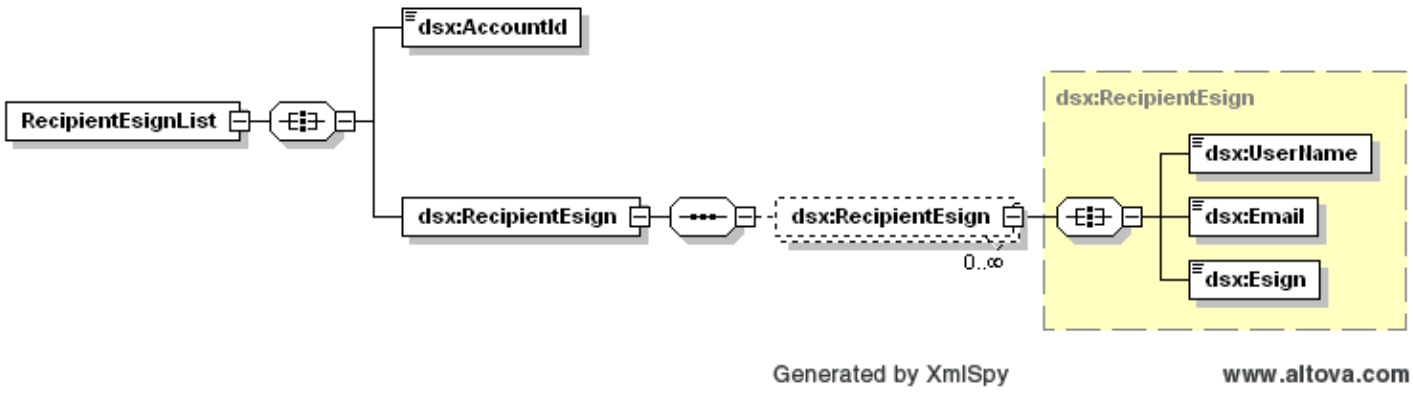
```
SOAPAction: "http://www.docusign.net/API/3.0/GetRecipientEsignList"

<?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>
    <GetRecipientEsignList xmlns="http://www.docusign.net/API/3.0">
      <UserName>string</UserName>
      <SenderEmail>string</SenderEmail>
      <SenderAccountId>string</SenderAccountId>
      <RecipientEmail>string</RecipientEmail>
    </GetRecipientEsignList>
  </soap:Body>
</soap:Envelope>
```

RecipientEsignList

This element returns a list of RecipientEsignRecords for a designated sender and recipient combination. For each record, the return Boolean value indicates whether or not an Esign agreement exists between the sender and that particular recipient.

Schema



Name	Schema Type	Description
AccountId	DSXId	The account holding the ESign agreement.
UserName	UserName	The recipient's name.
Email	Email	The recipient's email.
ESign	boolean	If true, an Esign agreement exists between the sender and the recipient

Usage Rules

- The UserName should be valid.
- The SenderEmail should be valid; else "Invalid_Email_Address_For_Sender" execution will be thrown.
- The length of UserName, SenderEmail, SenderAccountId and RecipientEmail must not exceed 100 characters.
- User defined by UserName and SenderEmail pair belong to SenderAccountId specified, else "User_Does_Not_Belong_To_Specified_Account" exception is thrown.

Chapter 9: Retrieve Membership Features

GetAccountMembershipFeaturesList

GetAccountMembershipFeaturesList is used to retrieve the features available to the membership.

Schema

Name	Schema Type	Description
AccountID	DSXId	Account Id of the user whose membership permissions for optional features are requested.

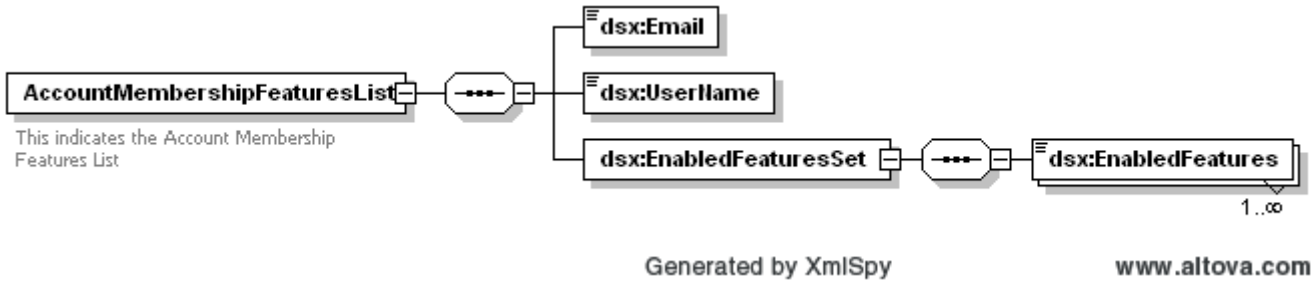
Sample Request XML

```
SOAPAction: "http://www.docusign.net/API/3.0/GetAccountMembershipFeaturesList"

<?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>
    <GetAccountMembershipFeaturesList xmlns="http://www.docusign.net/API/3.0">
      <AccountId>string</AccountId>
    </GetAccountMembershipFeaturesList>
  </soap:Body>
</soap:Envelope>
```

AccountMembershipFeaturesList

Schema



Name	Schema Type	Description

Email	Email	Email of the user whose membership feature list is requested.
UserName	UserName	UserName of the user whose membership feature list is requested.
EnabledFeaturesSet	string	Returns the set of DocuSign Professional features in membership.

Usage Rules

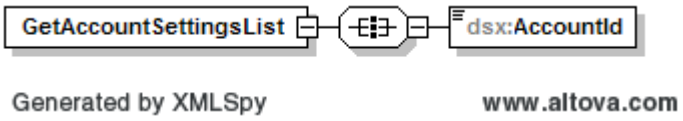
- The length of AccountId, Email and UserName must not exceed 100 characters.
- The EnabledFeatures would be unbounded. Values for this will be DocuSignProfessional, eOriginalVaultResponse, SequentialSigningAPI, SequentialSigningUI, TransactionPoint, CanSetDocumentSignerVisibility and AccountForcesSignerVisibility.

Chapter 10: Retrieve Account Features

GetAccountSettingsList

GetAccountSettingsList is used to retrieve the settings available to the account.

Schema



Name	Schema Type	Description
AccountID	DSXId	AccountId of the account settings to be returned.

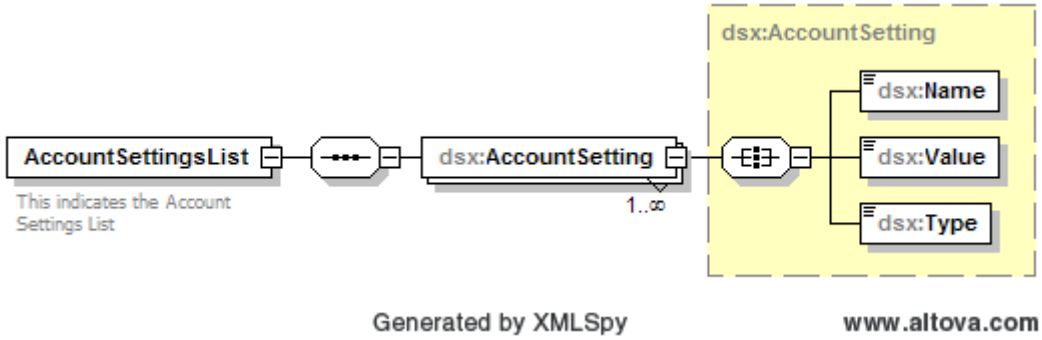
Sample Request XML

```
SOAPAction: "http://www.docusign.net/API/3.0/GetAccountSettingsList "

<?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>
    <GetAccountSettingsList xmlns="http://www.docusign.net/API/3.0">
      <AccountId>string</AccountId>
    </ GetAccountSettingsList >
  </soap:Body>
</soap:Envelope>
```

AccountSettingsList

Schema



Name	Schema Type	Description

AccountSetting	AccountSetting	Name: string: the name of the account setting. Value: string: the value of the account setting. Type: string: the type of the account setting (ie. Integer, Boolean, String)

Usage Rules

- The length of AccountId, must not exceed 100 characters.
- The AccountSetting is unbounded. Current "Name" values are: UserOverrideEnabled, ReminderEnabled, ReminderFrequency, ReminderDelay, ExpireEnabled, ExpireAfter, and ExpireWarnBuffer.

Chapter 11: Embedded Signing

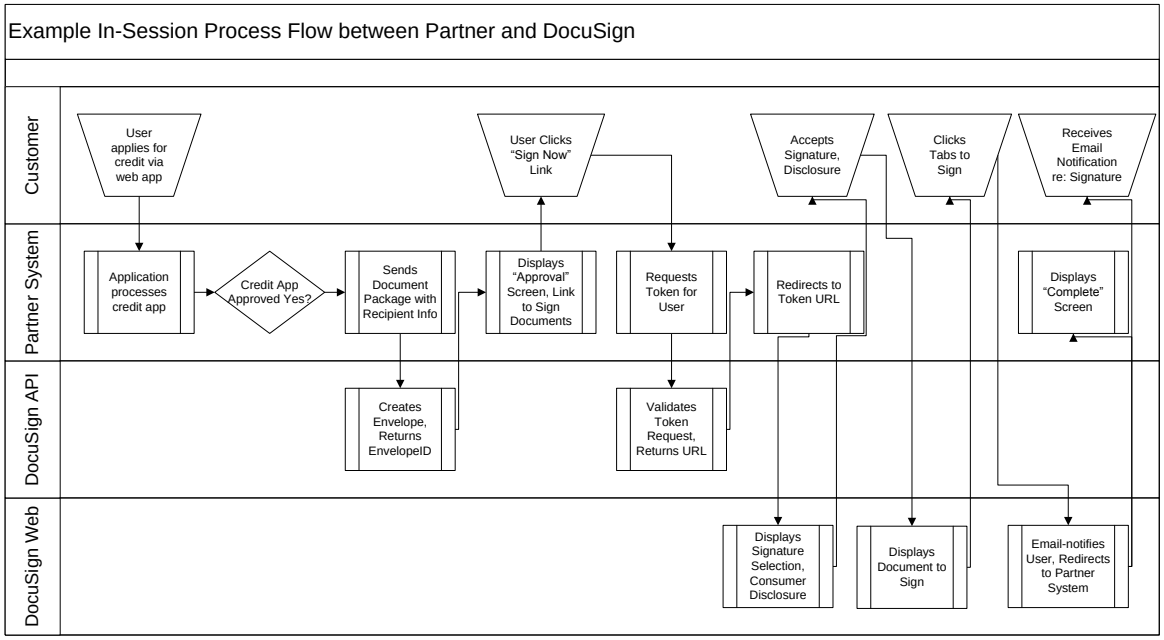
This chapter presents the principles and methods involved in implementing a DocuSign Connect API integration using the Embedded Signing interaction pattern for recipients. In this section, we specify methods, processes and exceptions that deviate from typical API integration concepts.

Embedded Signing is a departure from the default "Remote Signer" pattern, in which DocuSign brokers the communication with envelope Recipients via system-generated emails containing links to activate the viewing and signing process. Embedded Signing, in contrast, enables a API client application to maintain its connection with envelope Recipients by incorporating the DocuSign Signing Service directly into its process flow. The functional result is a more fluid document transaction and more transparent feature extension of the client application.

This tighter integration with DocuSign imposes additional technical and functional burdens on the client application, including addressing security, legal, and user experience requirements that are engineered into the DocuSign application in the Remote Signer pattern.

Embedded Signing Functional Process Flow

This overview of a typical embedded document-signing transaction should familiarize DocuSign Connect API users with terms and process. It is not a faithful reproduction of how the process is implemented on the web site or in the API. In this use case, a web user applies for credit from an online lending source that provides an instant credit decision and also electronic signature services.



Captive Recipients

The Embedded Signing pattern introduces a new class of recipients referred to as Captive Recipients. *Captive* refers to the exclusive relationship between the recipient and the sending account. In contrast, non-Captive Recipients are DocuSign-global entities that can have

recipient relationships with an unlimited number of senders. Characteristics of Captive Recipients are that they:

- Are identified by their FirstName, LastName, Email, sending account, and optionally a sender-provided ClientUserID.
- Have an exclusive relationship with their sending account. If another DocuSign customer has sent documents to the same person, either as a Remote or Captive Recipient, the recipient information is not related in any way.
- Can access DocuSign content only via their sending account's application; they cannot login directly to DocuSign.
- Do not have access to the DocuSign Member Console.
- Do not receive email notifications from DocuSign, except in cases specifically called out in this document.
- Cannot be aware that they have an account in DocuSign.

Captive Recipients can co-exist with Remote Recipients on a single envelope, but they cannot switch modalities after they are created. A Captive Recipient cannot be converted to a Remote Recipient at any point in the process. This is important to recognize during the client solution design phase, as DocuSign must know how a Recipient will interact with the client application at the time the envelope is created.

DocuSign Integration

Three distinct operational areas characterize Embedded Signing API integration:

- **Pre-DocuSign Operations –** Operations within the client application relating to creating the Envelope and navigating the Captive Recipient into the DocuSign-hosted pages.
- **DocuSign Operations –** Operations include displaying Consumer disclosure, Signature display and selection.
- **Post-DocuSign Operations –** Client-hosted pages that the Captive Recipient is redirected to upon reaching a terminal state in the DocuSign user experience.

Pre-DocuSign Operations

Creating the envelope

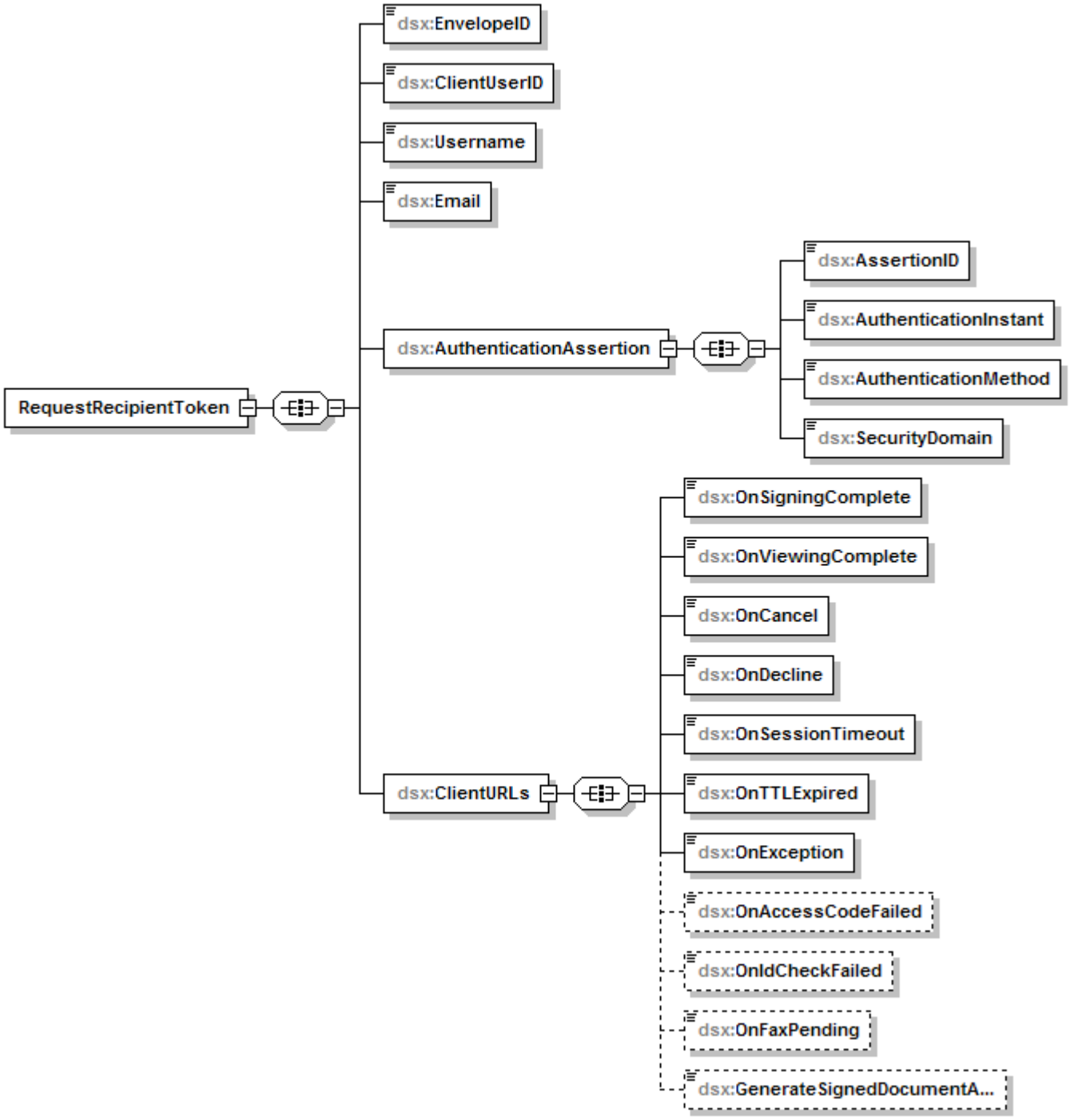
Creating envelops for Embedded Recipients parallels the Remote Recipient model. But sender needs to specify if the recipient is Remote or Embedded. Sender can do this by using ClientUserId. A non null ClientUserId implies that user is Embedded. A single envelope can contain both Embedded and Remote Recipients.

Requesting Recipient Tokens

DocuSign constrains access to the Embedded signature process via short-lived Recipient Tokens. Successful RequestRecipientToken requests return an URL that will invoke the DocuSign signing wizard for a particular envelope and Embedded Recipient. The following section describes the RequestRecipientToken method interface and the behaviors of Recipient Tokens.

The RequestRecipientToken method interface is divided into three sections: Entity Identification, Authentication Assertions and InSessionCallBackURL.

Schema



Entity Identification

The following parameters identify the Envelope and Captive Recipient for which the Recipient Token is requested:

- **EnvelopeID** – The DocuSign-generated value contained in the EnvelopeStatus object returned from the CreateAndSendEnvelope method. This parameter identifies the Envelope.
- **ClientUserID** – The value that specifies if the recipient is Remote or Embedded
- **UserName –** Specifies the username of the recipient

- **Email** - Specifies the Email of the recipient.

The Username and Email identifies a unique recipient and ClientUserID specifies if the recipient is Embedded or Remote. All the three parameter together identifies a Captive Recipient. All values must exist and correlate to a valid Envelope/Embedded Recipient entity relationship in order for the RequestRecipientToken request to succeed.

Authentication Assertions

DocuSign acts as a Single Sign-On consumer to the API client application in the Embedded Signing model. That is, DocuSign trusts that the client application authenticated the end-user sufficiently to establish their identity and offer documents to sign or view. This is critical to the seamless integration between the two applications, and it is necessary since DocuSign has no interaction with the end-user prior to displaying the documents. However, even though the trust relationship does not include a DocuSign-verification of the user identity, DocuSign requires the client application to assert that it has authenticated the end-user through some means. This assertion is communicated through the following parameters:

- **AuthenticationID** – A unique identifier of the authentication event executed by the client application.
- **AuthenticationInstant** – The date/time that the end-user was authenticated.
- **AuthenticationMethod** – The convention used to authenticate the end-user.
- **SecurityDomain** – The domain to which the user authenticated.

An entry is added into the Security Level section of the DocuSign Signing Certificate that reflects the "SecurityDomain – AuthenticationMethod" used to verify the user identity. Additional identity verification within DocuSign is optionally available through the ID Check and Access Code features.

ClientURLs

Client application will have the option of setting Client URLs to be called on specific DocuSign events.

When DocuSign redirects the Recipient to the Embedded client application, it appends a URL parameter "event" to the CallbackURL to indicate which terminal state the Recipient reached. The valid values are listed in Chapter 21: Embedded Callback Event Codes

.

For ClientURL embedded client should provide ClientURLs for nine events as in Appendix A.

XML Signing

Embedded clients must sign the body of the RequestRecipientToken message with a valid X.509 certificate. This process supports the Single Sign-On trust relationship between DocuSign and the client application. Client organizations must provide DocuSign with the certificate's Common Name (CN) during the implementation phase so that DocuSign can validate the XML signature. Supported certificate authorities are VeriSign and Thawte.

Expiration

- Recipient Tokens expire five minutes after they are issued by DocuSign. If a Recipient Token URL is invoked after it is expired, the user is re-directed to the Callback URL specified in the RequestRecipientToken request with event code TTLExpired.
- Recipient Tokens expire upon being successfully invoked.
- Active Recipient Tokens expire if the envelope is voided.
- Five minutes is the default "Time to Live" for Recipient tokens. This is a configurable setting.

Signing vs. Viewing Documents

Recipient Tokens must be requested (and returned) in order to sign or to view envelope documents. The DocuSign web application renders the view that is appropriate to the state of the envelope for the Recipient. The logic is as follows:

- If Recipient has uncompleted signing actions and the Recipient has not signed the envelope documents yet, DocuSign will present documents in Sign mode.
- If Recipient has uncompleted signing actions and the Recipient has already signed the envelope documents, DocuSign will present documents in View mode.
- If Recipient has completed all signing actions, DocuSign will present documents in View mode.
- If Recipient has uncompleted signing actions and the envelope has been declined by another Recipient, then the remaining recipients will not be able to view the envelope.
- If the sender has voided the envelope no recipient will be able to view or sign document.

DocuSign Operations

Once the authentication is complete user is directed to DocuSign-hosted pages. Here user will be able to view Consumer Disclosure, Select signature and complete signing by clicking on Tabs. When signing is complete notification email is sent and user is directed to partner system.

Post-DocuSign Landing Pages

The client application must create, host, and manage landing pages for each of the events that are detailed in the "InSessionCallBackURL" topic. The DocuSign web application will re-direct Recipients to the respective URLs upon reaching a terminal state in the signing/viewing process.

Addenda

Additional Features and Behaviors

This section describes new behaviors that are specific to Embedded Recipients, as well as product enhancements that are currently only available for Embedded Recipients.

- Embedded Recipients will receive an email on behalf of the sending account after they have completed the signing or viewing process. This is intended as a fraud-detection measure and is inherent to the Embedded process. The text of the email is configurable by the sending account.

- This is the only DocuSign-generated email that Embedded recipients will receive. They will not receive emails inviting them to sign, reporting that the Envelope has been voided, or informing them of Envelope completion because the Embedded client application is responsible for these communication tasks.
- Embedded clients have the ability to pass in default SignatureInfo (Signature Name, Signature Initials, and Signature Font), streamlining the user set-up process.
- Embedded clients have the ability to establish Envelope-specific SignatureInfo that is not necessarily the same as the original Recipient Username or the original signature created by the user. This allows the signature to conform to the Recipient's printed name on a document, specifically in cases where it is represented differently from one document to the next (i.e., initials are included in one Envelope instance and not the next). The SignatureInfo, however, cannot vary in separate documents within a single Envelope.
- The Consumer Disclosure agreement is consolidated into the "About You" page for Embedded Recipients. This action streamlines the signing process in relation to the Remote Recipient convention of a separate Consumer Disclosure page.
- Embedded clients have the ability to create their own branded Help content.

Suppressed Features/Behaviors

This section describes features and behaviors that are characteristic of Remote Recipients but not applicable to Embedded Recipients

- Envelope drafts cannot be created for an Envelope that contains a Captive Recipient. A CreateEnvelope request that contains a Captive Recipient will result in a SOAP fault. Envelopes with Captive Recipients must originate via the CreateAndSendEnvelope.
- Carbon-Copy Recipients cannot interact as Embedded Recipients.

  Embedded Recipients are not presented with the option to create a Password in DocuSign. Consequently, Captive Recipients cannot enter the DocuSign web application via the login screen at https://www.docusign.net/member/memberlogin.aspx. Captive Recipients can only enter DocuSign via client applications using Recipient Tokens.
- Embedded Recipients do not see the DocuSign Member Console.
- Embedded Recipients do not receive DocuSign-generated email notification for any events except for the Signing/Viewing confirmation upon completion of the respective task, described in the Additional Features/Benefits section.

Legal Considerations

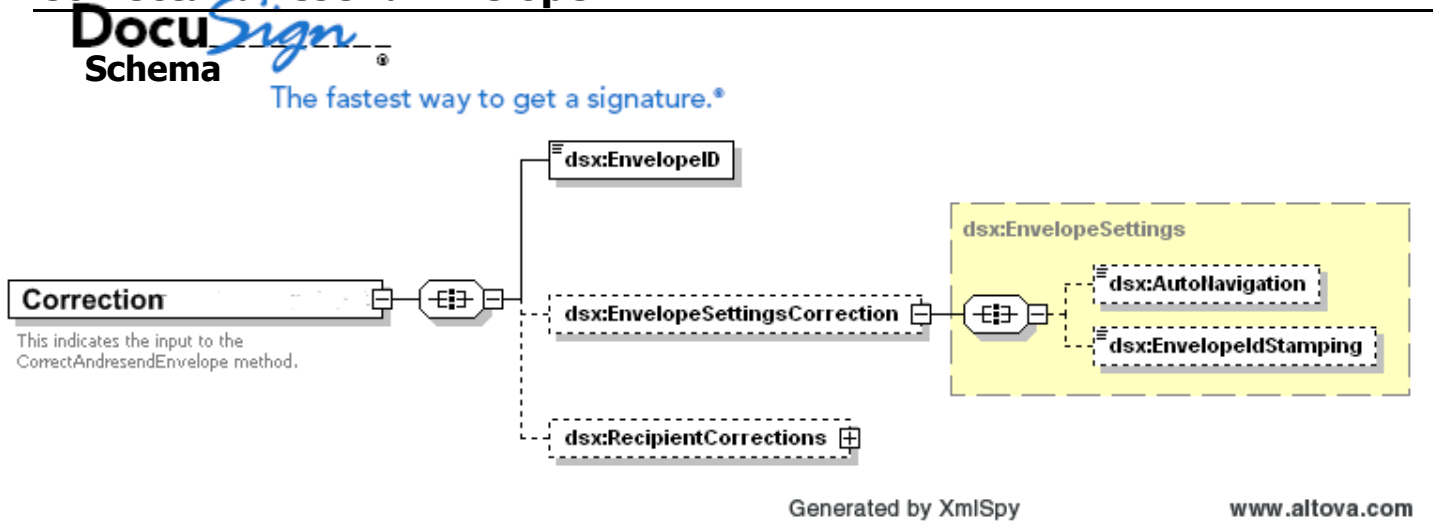
The DocuSign Remote Recipient process flow has been designed to comply with the federal Electronic Signatures in Global and National Commerce Act (ESIGN) and the model form of the Uniform Electronic Transaction Act (UETA). Most of the elements of the signing process such as authentication, consumer disclosure, and document delivery are performed within the DocuSign environment when executing the standard Remote Signing process. However, DocuSign's Embedded signing process provides the Client the flexibility for performing, within the Client's environment, certain authentication and signing process steps typically performed within the DocuSign environment. Accordingly, while DocuSign will work with the Client to recommend an overall business process for electronic signatures, DocuSign does not assume responsibility or liability for the steps taken outside of the DocuSign Environment and the effect such steps could have on the enforceability of electronic records signed electronically utilizing an Embedded signing process. Additional provisions dealing with an allocation of risk in the

Embedded signing process are contained in the relevant sections of the DocuSign Terms of Use for the Embedded process.

Chapter 12: Correct and Resend Envelopes

This chapter presents the principles and methods involved in implementing a DocuSign Connect API integration using the *CorrectAndResendEnvelope* and RecipientCorrectionStatus API method. The *CorrectAndResendEnvelope* API method enables API users to modify attributes of envelope recipients through the DocuSign web service, as well as resend Envelope activation emails. Through this API apart from changing recipient attributes settings like auto navigation, Envelope Stamping and authoritative copy can be changed.

CorrectandResendEnvelope



Name	Schema Type	Description
Envelope ID	DSXId	Envelope ID of the envelope which is corrected through API call.
EnvelopeSettingCorrection	*EnvelopeSettingCorrection*	Complex element consists of two elements *AutoNavigation and EnvelopeIdStamping. AutoNavigation is a Boolean element and i*If set to true, auto navigation feature will be enabled. *EnvelopeIdStamping* if set to true, Envelope Stamping feature will be enabled.
RecipientCorrections	RecipientCorrection	Contains all the attributes of the

701 5th Avenue, Suite 4500
Seattle, WA 98104
tel 206.219.0200
fax 206.622.9736
www.docusign.com

		recipient that could be corrected. Refer RecipientCorrection below for more details

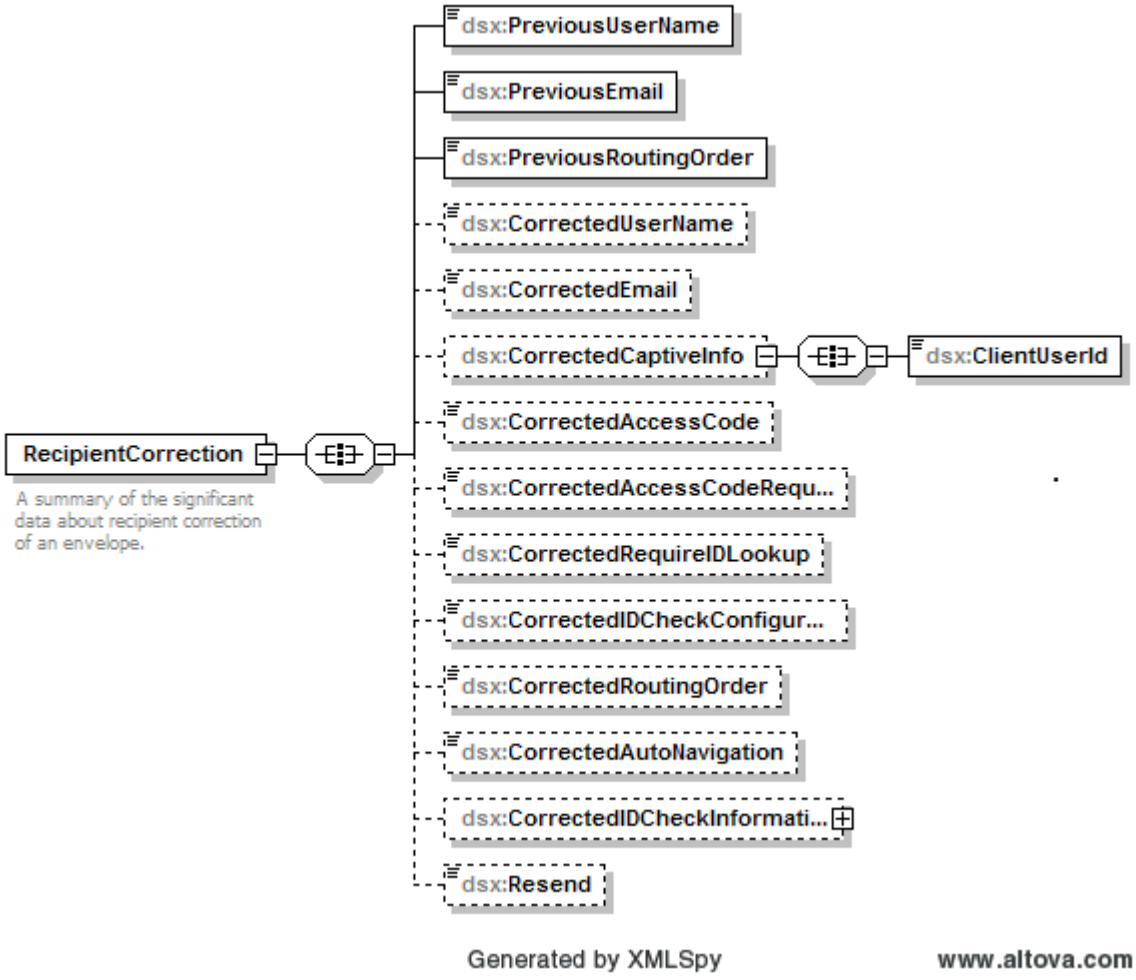
Sample Request XML:

SOAPAction: "http://www.docusign.net/API/3.0/CorrectAndResendEnvelope"

```
<?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>
    <CorrectAndResendEnvelope xmlns="http://www.docusign.net/API/3.0">
      <Correction>
        <EnvelopeID>string</EnvelopeID>
        <EnvelopeSettingsCorrection>
          <AutoNavigation>boolean</AutoNavigation>
          <EnvelopeIdStamping>boolean</EnvelopeIdStamping>
        </EnvelopeSettingsCorrection>
        <RecipientCorrections>
          <RecipientCorrection>
            <PreviousUserName>string</PreviousUserName>
            <PreviousEmail>string</PreviousEmail>
            <PreviousRoutingOrder>unsignedShort</PreviousRoutingOrder>
            <CorrectedUserName>string</CorrectedUserName>
            <CorrectedEmail>string</CorrectedEmail>
            <CorrectedCaptiveInfo xsi:nil="true" />
            <CorrectedAccessCode>string</CorrectedAccessCode>
            <CorrectedAccessCodeRequired>boolean</CorrectedAccessCodeRequired>
            <CorrectedRequireIDLookup>boolean</CorrectedRequireIDLookup>
            <CorrectedIDCheckConfigurationName xsi:nil=”true”>string</ CorrectedIDCheckConfigurationName >
            <CorrectedRoutingOrder>unsignedShort</CorrectedRoutingOrder>
            <CorrectedAutoNavigation>boolean</CorrectedAutoNavigation>
            <CorrectedIDCheckInformationInput xsi:nil="true" />
            <Resend>boolean</Resend>
          </RecipientCorrection>
          <RecipientCorrection>
            <PreviousUserName>string</PreviousUserName>
            <PreviousEmail>string</PreviousEmail>
            <PreviousRoutingOrder>unsignedShort</PreviousRoutingOrder>
            <CorrectedUserName>string</CorrectedUserName>
            <CorrectedEmail>string</CorrectedEmail>
            <CorrectedCaptiveInfo xsi:nil="true" />
            <CorrectedAccessCode>string</CorrectedAccessCode>
            <CorrectedAccessCodeRequired>boolean</CorrectedAccessCodeRequired>
            <CorrectedRequireIDLookup>boolean</CorrectedRequireIDLookup>
            <CorrectedRoutingOrder>unsignedShort</CorrectedRoutingOrder>
            <CorrectedAutoNavigation>boolean</CorrectedAutoNavigation>
            <CorrectedIDCheckInformationInput xsi:nil="true" />
            <Resend>boolean</Resend>
          </RecipientCorrection>
        </RecipientCorrections>
      </Correction>
    </CorrectAndResendEnvelope>
  </soap:Body>
</soap:Envelope>
```

RecipientCorrection

Schema

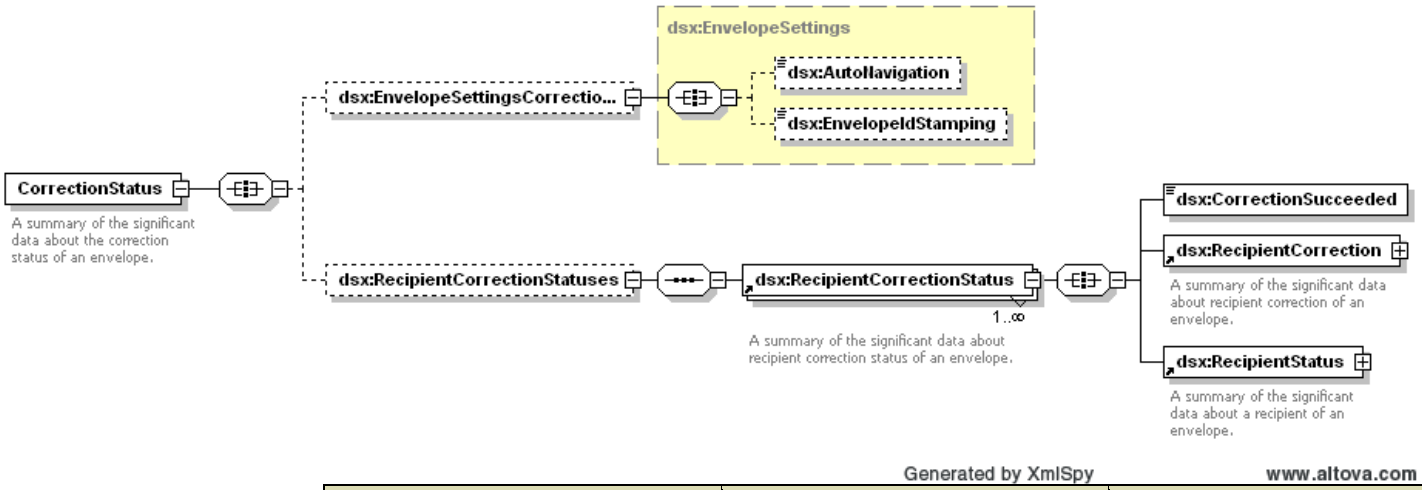


Name	Schema Type	Description
PreviousUsername	UserName	User name of the recipient specified in the envelope which needs correction.
PreviousEmail	Email	Email of the recipient specified in the envelope which needs correction.
PreviousRoutingOrder	PositiveShort	Routing order of the recipient specified in the envelope which needs correction.
CorrectedUsername	UserName	Corrected user name of the recipient.
CorrectedEmail	Email	Corrected Email of the recipient.
CorrectedCaptiveInfo	CorrectedCaptiveInfo	Specifies if the recipient is captive or not. If the ClientUserId is not null then the recipient is captive.
CorrectedAccessCode	String	Corrected access for the recipients
CorrectedAccessCodeRequired	Boolean	If set to true, then access code check will be required for the recipient.
CorrectedRequireIDLookup	Boolean	If set to true, then ID check will be required for the recipient.
CorrectedIDCheckConfigurationName	String	Corrected ID check configuration by name.
CorrectedRoutingOrder	PositiveShort	Corrected routing order of the recipient.
CorrectedAutoNavigation	Boolean	Corrected setting for the auto navigation feature. If Boolean is set to true auto navigation feature will be enabled for the recipient

CorrectedIDCheckInformationInput	*CorrectedIDCheck*InformationInput	Specified the corrected IDCheck information. For details please refer *IDCheckInformationInput under CreateAndSend method*
Resend	Boolean	Specifies if the email notifying correction needs to be sent to the recipient.

CorrectionStatus

Schema



Name	Schema Type	Description
EnvelopeSettingsCorrection	EnvelopeSettingsCorrection	Consists of AutoNavigation and EnvelopeIdStamping. A true value indicates that the feature is enabled.
RecipientCorrectionStatuses	*RecipientCorrectionStatuses*	Please check the details below.

RecipientCorrectionStatus

Name	Schema Type	Description
CorrectionSuccededed	boolean	Return true if the envelope correction is successful
RecipientCorrection	*RecipientCorrection*	Returns the corrected information. For more details check the *RecipientCorrection* section above
RecipientStatus	*RecipientStatus*	Returns current status of the recipient. For more details check RecipientStatus section in above.

Rules for *CorrectAndResendEnvelope*

API user specific rules

- The length of any of the email addresses (specified as *CorrectedEmail*) must not exceed 100 characters. Else, the XML validation fails and the processor throws a Validation error.
- The email addresses (specified as *CorrectedEmail*) must conform to the appropriate email format. Else, the XML validation fails and the processor throws a Validation error.
- The length of any of the EnvelopeID must not exceed 100 characters. Else, the XML validation fails and the processor throws a Validation error.
- If the *CorrectedAccessCode* field is not null and not empty then regardless of whether the value of *CorrectedAccessCode*Required is 'true' or 'false' the *CorrectedAccessCode* is considered as the new Access Code for the specified recipient.
- If the *CorrectedAccessCode* field is null or empty:
  - If the value of *CorrectedAccessCodeRequired* is 'false', the existing Access Code for the specified recipient is cancelled.
  - If the *CorrectedAccessCodeRequired* is 'true' then the existing access code is left as it is.

Rules for Exceptions thrown by the API

- An envelope having the specified Envelop*eID* must exist in the system. Else, an exception with the error message "Envelope_Does_Not_Exist" is thrown.
- The status of the envelope with the specified Envelop*eID* must be 'Sent' or 'Delivered'. Else, an exception with the error message "Envelope_Cannot_Correct_Invalid_State" is thrown.
- The user making the API calls must have permissions to correct the Envelope. Else, an exception with the error message "User_Lacks_Permissions" is thrown. The API user can correct the Envelope only if the user:
  - Is the sender of the envelope

- User specified by PreviousUserName and PreviousEmail should be the owner of the envelope. Else an exception with error message "User_Not_Envelope_Sender" is thrown.
- The envelope with the specified Envelop*eID* must not have duplicate recipients. Else an exception is thrown with the error message "Envelope_Has_Duplicate_Recipients". An envelope with duplicate recipients cannot be corrected, because if a correction is specified for a user who exists as more than one recipient, then it raises an ambiguity on which instance of the user to be corrected.
- More than one correction must not be specified for a given recipient as it raises an ambiguity on which correction to apply. Else, an exception with the error message "Correction_Has_Duplicate_Recipients" thrown.
- The *CorrectedUserName* of any correction (after trimming of leading and following spaces) must not be empty. Else an exception with the error message "Correction_Has_A_Blank_Username" is thrown.
- The final set of recipients of the envelope (assuming that the corrections that succeed are applied) must not contain duplicates. Else an exception is thrown with the error message "Correction_Results_In_Duplicate_Recipients".
- Recipient specified by CorrectedUserName and CorrectedEmail should exist else exception with error message "Recipient_Not_Found_For_Correction" is thrown.
- Correcting Carbon copy recipient is not allowed. If tried the exception with error message "Captive_Carbon_Copy_Recipient_Not_Supported" will be thrown.
- If the corrected Email specified has already been reserved by some other account then an exception with error message "Corrected_Email_Is_Reserved" is thrown.

Rules for return value of the API

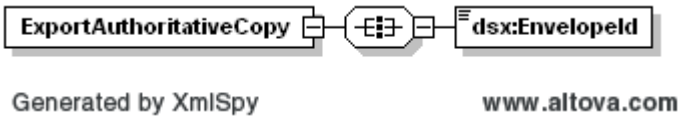
- The status of a recipient (for whom a correction is specified) must not be 'Signed', 'Declined' or 'Completed'. Else, correction is not permitted for this recipient and *CorrectionSucceeded* value is set to 'false'. In other words a *CorrectionSucceeded* value of 'false' indicates that the correction could not be applied, as the status of the recipient does not permit corrections.
- The *RecipientCorrection* object in the *CorrectAndResendEnvelope* the same as the *RecipientCorrection* passed as the argument to the API.

Chapter 13: Export Authoritative Copy Envelopes

This chapter presents the principles and methods involved in implementing a DocuSign Connect API integration using the *ExportAuthoritativeCopy* and *AcknowledgeAuthoritativeCopyExport* API methods. The *ExportAuthoritativeCopy* API method enables API users to extract Authoritative Copy envelopes from DocuSign. The *AcknowledgeAuthoritativeCopyExport* API method is used to indicate success of the extract call and to obtain the key to unlock the envelope's documents.

ExportAuthoritativeCopy

Schema



Name	Schema Type	Description
Envelope ID	DSXId	Envelope ID of the envelope which is to be exported.

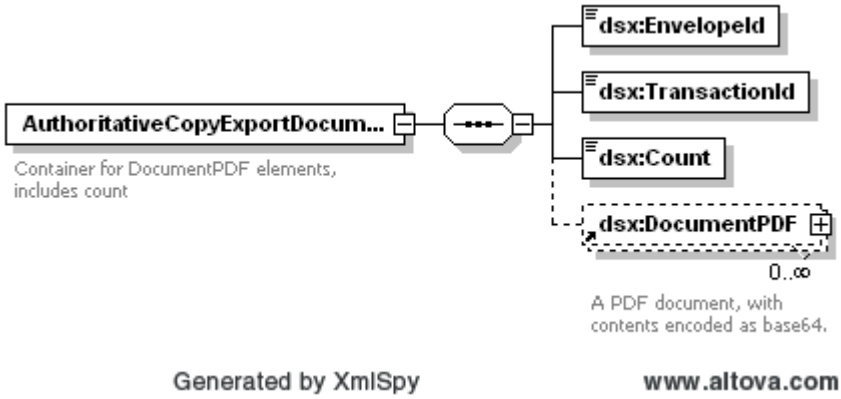
Sampe Request XML:

```
SOAPAction: "http://www.docusign.net/API/3.0/ExportAuthoritativeCopy"

<?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>
    <ExportAuthoritativeCopy xmlns="http://www.docusign.net/API/3.0">
      <EnvelopeId>string</EnvelopeId>
    </ExportAuthoritativeCopy>
  </soap:Body>
</soap:Envelope>
```

AuthoritativeCopyExportDocuments

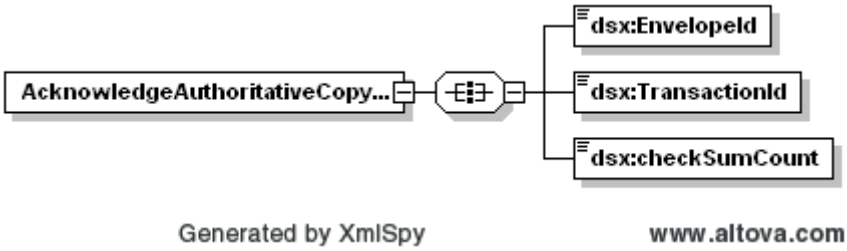
Schema



Name	Schema Type	Description
EnvelopeId	DSXId	Envelope ID of the envelope which is to be exported.
TransactionId	DSXId	Transaction ID for this export process.
Count	Int	Number of documents in the DocumentPDF array.
DocumentPDF	DocumentPDF	Array of encrypted document PDFs in the envelope.

AcknowledgeAuthoritativeCopyExport

Schema



Name	Schema Type	Description
EnvelopeId	DSXId	Envelope ID of the envelope which is to be

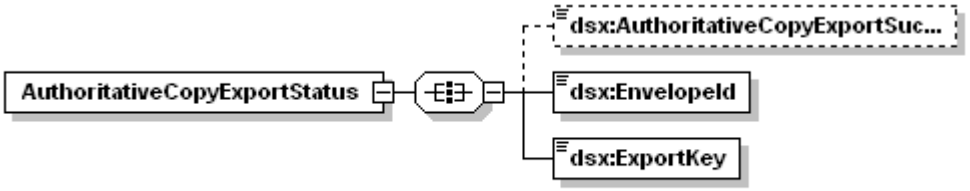
		exported.
TransactionId	DSXId	Transaction ID for this export process. Obtained from the ExportAuthoritativeCopy API call.
checkSumHash	base64Binary	Total hash of all the documents returned in the ExportAuthoritativeCopy API call.

Sampe Request XML:

```
SOAPAction: "http://www.docusign.net/API/3.0/AcknowledgeAuthoritativeCopyExport"

<?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>
    <AcknowledgeAuthoritativeCopyExport xmlns="http://www.docusign.net/API/3.0">
      <EnvelopeId>string</EnvelopeId>
      <TransactionId>string</TransactionId>
      <checkSumHash>base64Binary</checkSumHash>
    </AcknowledgeAuthoritativeCopyExport>
  </soap:Body>
</soap:Envelope>
```

AuthoritativeCopyExportStatus



Name	Schema Type	Description
AuthoritativeCopyExportSuccess	boolean	Return true if the envelope export is successful
EnvelopeId	DSXId	Envelope ID of the envelope which is to be exported.
ExportKey	DSXId	Key used to decrypt the Rijndael encrypted documents returned in the

		ExportAuthoritativeCopy API call. Encryption mode is CBC. Encryption padding is PKCS7.

Rules for exporting Authoritative Copy envelopes

API user specific rules

- The envelope ID passed to the API must be valid.
- The envelope ID passed to the API must be in the Authoritative Copy status in DocuSign.
- The user and account must be configured properly by DocuSign in order to enable this API.

Rules for Exceptions thrown by the ExportAuthoritativeCopy API

- Envelope_Does_Not_Exist – invalid envelope ID passed.
- User_Lacks_Permissions - The user making the API calls must have permissions to export Authoritative Copy Documents.
- Envelope_AC_Export_Completed – the envelope being requested has already been exported from DocuSign.
- Envelope_AC_Export_Not_AC_Copy – the envelope being requested is not an Authoritative Copy.

Rules for Exceptions thrown by the AcknowledgeAuthoritativeCopyExport API

- Envelope_Does_Not_Exist – invalid envelope ID passed.
- User_Lacks_Permissions - The user making the API calls must have permissions to export Authoritative Copy Documents.
- Envelope_AC_Export_Invalid_Status – envelope is not in the valid export status.
- Envelope_AC_Export_Invalid_TransID – transaction ID passed is invalid.

Rules for return value of the AcknowledgeAuthoritativeCopyExport API

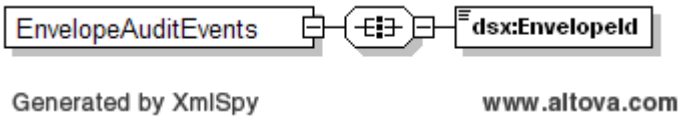
- AuthoritativeCopyExportSuccess – True if export was successfully completed.
- ExportKey – key to decrypt the Rijndael encrypted documents. This key is only provided if AuthoritativeCopyExportSuccess is true. Encryption mode is CBC. Encryption padding is PKCS7.

Chapter 14: Accessing Envelope Events

This chapter presents the principles and methods involved in implementing a DocuSign Connect API integration using the *EnvelopeAuditEvent*. The *EnvelopeAuditEvent* API method enables API users to request the events performed on an envelope. The event are returned in an XML document.

EnvelopeAuditEvents

Schema



Name	Schema Type	Description
EnvelopeId	DSXId	Envelope ID of the envelope to return the events for.

Sample Request XML:

```
SOAPAction: "http://www.docusign.net/API/3.0/EnvelopeAuditEvents"

<?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>
    <EnvelopeAuditEvents xmlns="http://www.docusign.net/API/3.0">
      <EnvelopeId>string</EnvelopeId>
    </EnvelopeAuditEvents>
  </soap:Body>
</soap:Envelope>
```

Rules for accessing envelope events.

API user specific rules

- The envelope ID passed to the API must be valid.
- The envelope ID passed to the API must be accessible by the API user.
- The user and account must be configured properly by DocuSign in order to enable this API.

Rules for Exceptions thrown by the EnvelopeAuditEvents API

- Envelope_Does_Not_Exist – invalid envelope ID passed.

- User_Lacks_Permissions - The user making the API calls must have permissions to export Authoritative Copy Documents.

Return XML for EnvelopeAuditEvents API

- Below is a sample of the XML returned:

```
<DocuSignEnvelopeEvents EnvelopeId="xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx" xmlns="">
    <Event>
    <logTime>2008-02-12T16:52:00.6430736-08:00</logTime>
    <Source>Web</Source>
            <UserName>User name here</UserName>
    <Action>Registered</Action>
    <Message>The envelope was created by SS UI and API</Message>
    <EnvelopeStatus>Created</EnvelopeStatus>
    <ClientIPAddress>127.0.0.1</ClientIPAddress>
    <Information />
    </Event>
    <Event>
    <logTime>2008-02-12T16:52:52.0519556-08:00</logTime>
    <Source>Web</Source>
    <UserName>The User</UserName>
    <Action>Sent Invitations</Action>
    <Message>The User sent an invitation to User email[the.user@docusign.com]</</Message>
    <EnvelopeStatus>Sent</EnvelopeStatus>
    <ClientIPAddress>127.0.0.1</ClientIPAddress>
    <Information>User email[the.user@docusign.com]</Information>
    </Event>
</DocuSignEnvelopeEvents>
```

Chapter 15: Ping the service

This chapter presents the principles and methods involved in implementing a DocuSign Connect API integration using the *Ping*. The *Ping* API method enables API users make a simple call to the API service to determine its active state.

Ping

Schema

Sample Request XML:

```
SOAPAction: "http://www.docusign.net/API/3.0/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:Body>
    <Ping xmlns="http://www.docusign.net/API/3.0" />
  </soap:Body>
</soap:Envelope>
```

Return XML

This method simply returns “true” in the result XML if the calling application was able to reach it.

Chapter 16: GetStatusInDocuSignConnectFormat

GetStatusInDocuSignConnectFormat API is reserved for future use.

GetStatusInDocuSignConnectFormat

Schema

Name	Schema Type	Description
EnvelopeId	DSXId	Envelope ID of the envelope to return the status for.

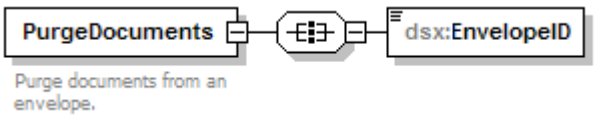
This method has been reserved for future use

Chapter 17: PurgeDocuments

This method can be used to purge completed documents from the DocuSign system. The envelope information for the document will remain.

PurgeDocuments

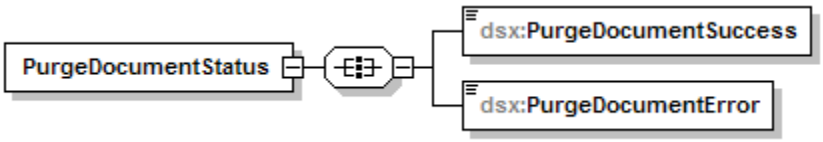
Schema



Name	Schema Type	Description
EnvelopeId	DSXId	Envelope ID to purge the documents from.

This method returns PurgeDocumentStatus upon execution completion.

PurgeDocumentStatus



Name	Schema Type	Description
PurgeDocumentSuccess	boolean	True if successful.
PurgeDocumentError	String	If PurgeDocumentSuccess is false an error description is returned.

PurgeDocuments rules and exceptions

- Envelope_Does_Not_Exist – invalid envelope ID passed. This exception will be thrown.

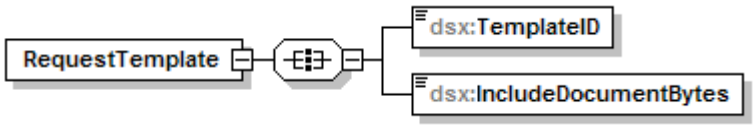
- User_Lacks_Permissions - The user making the API calls must have permissions to export Authoritative Copy Documents. This exception will be thrown.
- If PurgeDocumentSuccess returns false one of the following error conditions occurred:
  - Only the sender may remove the envelope documents.
  - Envelope is an authoritative copy, documents cannot be removed.
  - Envelope is not complete, documents cannot be removed.

Chapter 18: Manage Templates

These methods can be used to manage server side templates in the DocuSign system.

RequestTemplate

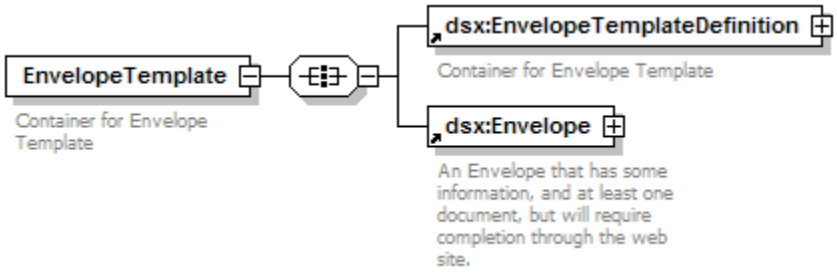
Schema



Name	Schema Type	Description
TemplateId	DSXId	Template ID to return.
IncludeDocumentBytes	Boolean	If true, include the document bytes of the template within the Envelope object returned.

Retrieves a specific template from the server. This method returns EnvelopeTemplate upon execution completion.

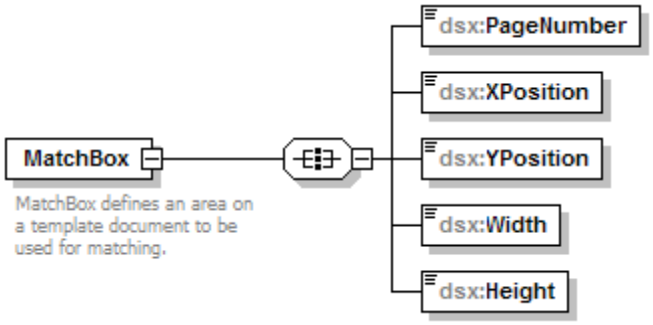
EnvelopeTemplate



Name	Schema Type	Description

EnvelopeTemplateDefinition	EnvelopeTemplateDefinition	Defines the attributes of a template. See below.
Envelope	Envelope	Template is contained within an Envelope object defined earlier in this document.

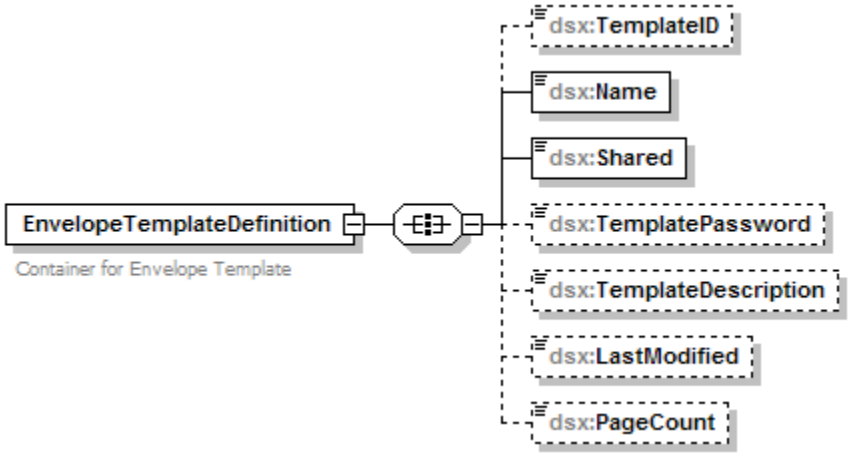
MatchBox



Name	Schema Type	Description
PageNumber	Integer	Page to put the matchbox on.
XPosition	Integer	X position of the matchbox.
YPosition	Integer	Y position of the matchbox.
Width	Integer	Width of the matchbox.
Height	Integer	Height of the matchbox.

Matchboxes can be supplied with the document structure within the Envelope. They are used in the matching process when documents are uploaded for creating templates.

EnvelopeTemplateDefinition



Name	Schema Type	Description
TemplateID	DSXId	Unique identifier of the template.
Name	String	Name of the template.
Shared	Boolean	True if shared with the account.
TemplatePassword	String	Password if the template is locked. NOTE: This will only be returned if the requesting user is the creator of the template.
TemplateDescription	String	Description of the template.
LastModified	DateTime	Last time the template was modified.
PageCount	Integer	Number of pages in document of the template.

RequestTemplate rules and exceptions

- Account_Lacks_Permissions – account does not have permissions to use this API.
- User_Lacks_Permissions – user does not have permissions to use this API.
- Template_Unable_To_Load – template ID provided was invalid.

RequestTemplates

Schema

Name	Schema Type	Description
AccountId	DSXId	Account to request the templates for. If left blank the API user's default account will be used.

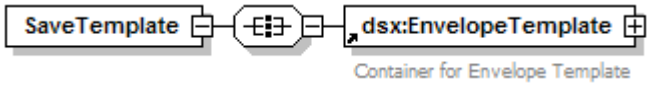
This method returns a list of server side templates available for this account. The list format is in EnvelopeTemplateDefinition described above.

RequestTemplates rules and exceptions

- Account_Lacks_Permissions – account does not have permissions to use this API.
- User_Lacks_Permissions – user does not have permissions to use this API.

SaveTemplate

Schema



Name	Schema Type	Description
EnvelopeTemplate	EnvelopeTemplate	Template to save. EnvelopeTemplate object was described earlier.

This method returns SaveTemplateResult.

Input to SaveTemplate is the EnvelopeTemplate object described earlier.

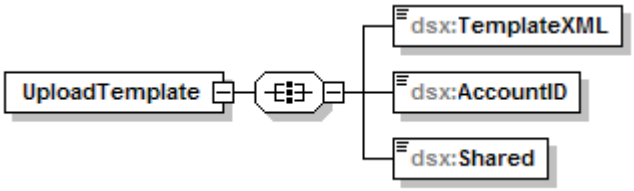
SaveTemplate rules and exceptions

- Account_Lacks_Permissions – account does not have permissions to use this API.

- User_Lacks_Permissions – user does not have permissions to use this API or the user does not have permissions to manage templates for the account.
- If the template ID is not provided in EnvelopeTemplate a new template will be created.

UploadTemplate

Schema



Name	Schema Type	Description
TemplateXML	String	XML data to be transformed into a DocuSign server side template. NOTE: only DocuSign Professional template XML is supported.
AccountID	DSXId	Account ID to add the template to. If blank, the API user's default account will be used.
Shared	Boolean	True if the template should be shared with the other users of the account.

This method returns SaveTemplateResult.

NOTE: this method currently only supports DocuSign Professional Template XML.

UploadTemplate rules and exceptions

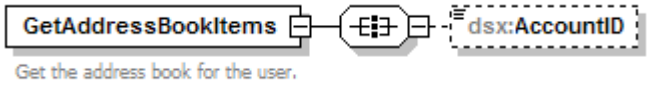
- Account_Lacks_Permissions – account does not have permissions to use this API.
- User_Lacks_Permissions – user does not have permissions to use this API or the user does not have permissions to manage templates for the account.

Chapter 19: Manage Address Book

These methods can be used to manage your server side address book in the DocuSign system. You will be able to retrieve, insert, update, and remove address book entries.

GetAddressBookItems

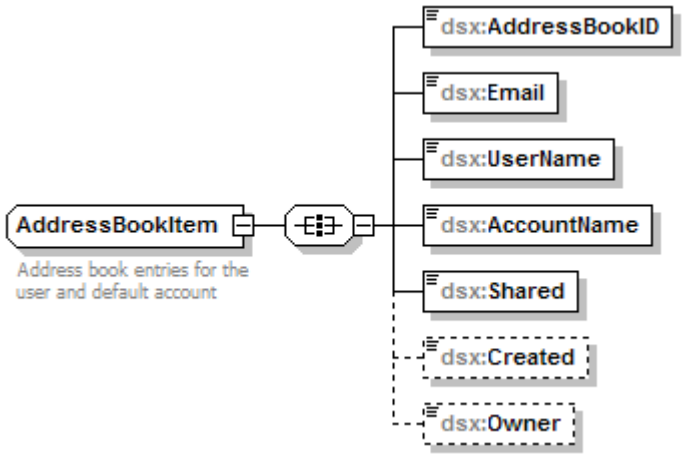
Schema



Name	Schema Type	Description
AccountID	DSXId	

Retrieves your address book and account shared address book. This method returns an array of AddressBookItem objects.

AddressBookItem



Name	Schema Type	Description
AddressBookID	DSXId	Unique identifier of the address book item.
Email	String	Email associated with the address book item.
UserName	String	Name associated with the address book item.

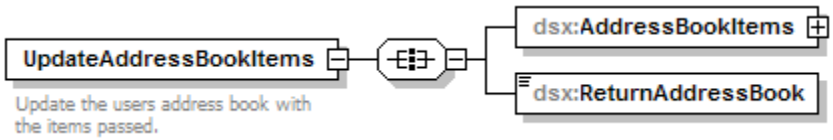
AccountName	String	Account name associated with the address book item.
Shared	Boolean	True if the address book item is shared with the API user's account.
Created	DateTime	Date the address book item was created.
Owner	Boolean	True if the API user account is the owner of the address book item.

GetAddressBookItems rules and exceptions

- Account_Lacks_Permissions – account does not have permissions to use this API.
- User_Lacks_Permissions – user does not have permissions to use this API.
- Address book items that are passed to this API that the user does not own will not be removed.

UpdateAddressBookItems

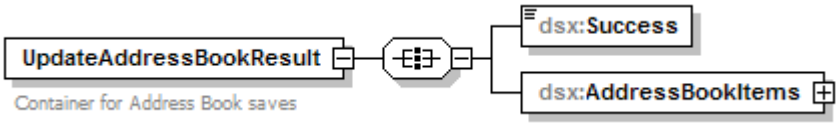
Schema



Name	Schema Type	Description
AddressBookItems	AddressBookItem Array	Array of address book items to update or insert.
ReturnAddressBook	Boolean	If true, return the address book item list in the UpdateAddressBookResult object.

Updates and inserts all the specified items passed to your address book. This method returns an UpdateAddressBookResult object.

UpdateAddressBookResult



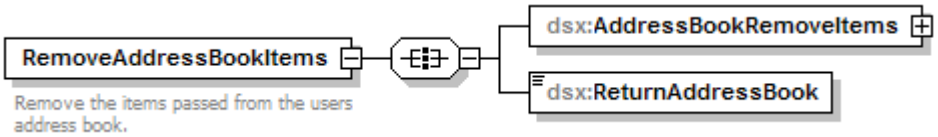
Name	Schema Type	Description
Success	Boolean	True if successful.
AddressBookItems	AddressBookItem Array	Array of AddressBookItem after the updates are completed. AddressBookItem object described earlier.

UpdateAddressBookItems rules and exceptions

- Account_Lacks_Permissions – account does not have permissions to use this API.
- User_Lacks_Permissions – user does not have permissions to use this API.
- Address book items that are passed to this API that the user does not own will not be removed.
- If the address book ID is left blank in the AddressBookItem obect a new address book entry will be added.
- If the AddressBookItem share is changed and the user does not have address book sharing privileges, the address book item share will not be updated.
- AddressBook_Email_Invalid – if there is an invalid email in the address book items passed to the API this error will be returned and no updates will be done.

RemoveAddressBookItems

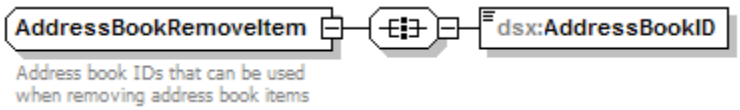
Schema



Name	Schema Type	Description
AddressBookRemoveItems	AddressBookRemoveItem Array	Array of address book item IDs to remove.
ReturnAddressBook	Boolean	If true, return the address book item list in the UpdateAddressBookResult object.

Removes all the specified items passed from your address book. This method returns an UpdateAddressBookResult object described earlier.

AddressBookRemoveItem



Name	Schema Type	Description
AddressBookID	DSXId	Address book ID to remove.

RemoveAddressBookItems rules and exceptions

- Account_Lacks_Permissions – account does not have permissions to use this API.
- User_Lacks_Permissions – user does not have permissions to use this API.
- Address book items that are passed to this API that the user does not own will not be removed.
- Invalid address book IDs will be ignored.

Chapter 20: Authentication Token

These methods can be used to get a onetime use URL with an authentication token to launch the DocuSign member system.

GetAuthenticationToken

Schema

Name	Schema Type	Description
GoToEnvelopeID	string	Optional envelope ID. If provided the URL returned will send the user directly to viewing the envelope. If left blank the user will be taken to the member console.

This method returns a string that is the URL to access the DocuSign member system. This URL can be placed in a browser and you will be automatically logged into the member system.

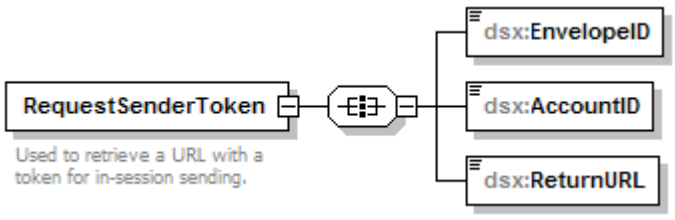
GetAuthenticationToken rules and exceptions

- User_Does_Not_Exist_In_System – API user requesting the authentication token URL does not have access to the system.
- Account_Lacks_Permissions – API user requesting the authentication token URL does not have valid API permissions.

RequestSenderToken

This method is used to get a onetime use login token that allows the user to be placed into the DocuSign sending wizard. Upon sending completion the user is returned to the return URL provided by the API application.

Schema



Name	Schema Type	Description

EnvelopeID	string	Optional envelope ID. If provided the user will be placed into the envelope in process. If left blank the user will be placed into a new envelope for sending.
AccountID	String	DocuSign account ID to map the envelope to.
ReturnURL	String	URL to send the user to upon completion of the sending process. The URL will have an event passed to it as a query parameter. The parameter will be named "event", see "In-session sending events" below for description of the valid events. The DocuSing Envelope Id will also be returned in the "envelopeId" parameter.

In-session sending events

Events returned on the ReturnURL as a query string parameter. The event is sent in the parameter named "event". The ReturnURL is called with the event on any completion state of the envelope send. Valid events can be found in the schema type InSessionSendEvent in the DocuSign API XSD.

Event	**Description**
Send	User sends the envelope.
Save	User saves a draft of the envelope.
Cancel	User cancels the sending transaction. No envelope ID returned.
Error	Error performing the send.
SessionEnd	Sending session ended before the user completed.

RequestSenderToken rules and exceptions

- User_Does_Not_Exist_In_System – API user requesting the sending token URL does not have access to the system.
- Account_Does_Not_Exist_In_System – Account is not valid.
- Account_Lacks_Permissions – API user requesting the sending token URL does not have valid API permissions.
- User_Lacks_Membership – user is not a valid member of the account.
- User_Not_Envelope_Sender – user is not a valid sender in the account.

- Envelope_Does_Not_Exist – if envelope ID is provided, envelope ID does not exist.
- Account_Not_Authorized_For_Envelope – if envelope ID is provided, account is not valid for the envelope.

Chapter 21: Embedded Callback Event Codes

When DocuSign redirects the *Recipient* to the Embedded client application, it will append a URL parameter "event" to the *CallbackURL* to indicate which terminal state the *Recipient* reached. The valid values are:

Event	Description
OnSigningComplete	The Recipient successfully completed the signing the document.
OnViewingComplete	The Recipient successfully viewed the document.
OnCancel	The Recipient cancelled the document while viewing the envelope.
OnDecline	The recipient declined the envelope.
OnSessionTimeout	The recipient session has timed out.
OnTTLExpired	If a Recipient Token URL is invoked after it is expired.
OnAccessCodeFailed	The Recipient failed to provide the correct Access Code.
OnIdCheckFailed	The Recipient failed to pass the ID Check authentication questions.
OnException	If an exception is thrown.
OnFaxPending	If a faxed in document is pending.

Asynchronous Document Generation

The following value can be set to cause the system to report that the envelope has been completed before the actual final documents have been generated.

Name	Schema Type	Description
GenerateSignedDocumentAsynch	boolean	Optional flag. If true, the final document generation will happen after the envelope status has been set to 'Completed'. Default is false.

Chapter 22: Credential API

This chapter covers the methods involved in implementing a DocuSign Credential API. The DocuSign Credential API can be used to authenticate with an email and password to acquire your API credentials. The credentials can then be used to access the DocuSign Connect API methods described earlier. If you have an integrators key from DocuSign you can place it in front of the email argument bracketed, "[" and "]".

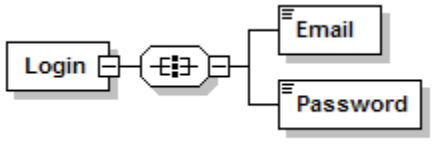
The DocuSign Credential API exposes the following methods:

- **Login** – Allows you to authenticate with an email and password to retrieve your API credentials required to access the DocuSign Connect API.
- **Ping** – Allows you to validate that the Credential API is alive.

Login

The Login method can be used to authenticate and retrieve DocuSign Connect API credentials based on an email and password. This method will return an array of valid DocuSign Connect API accounts that match the email and password provided.

Schema



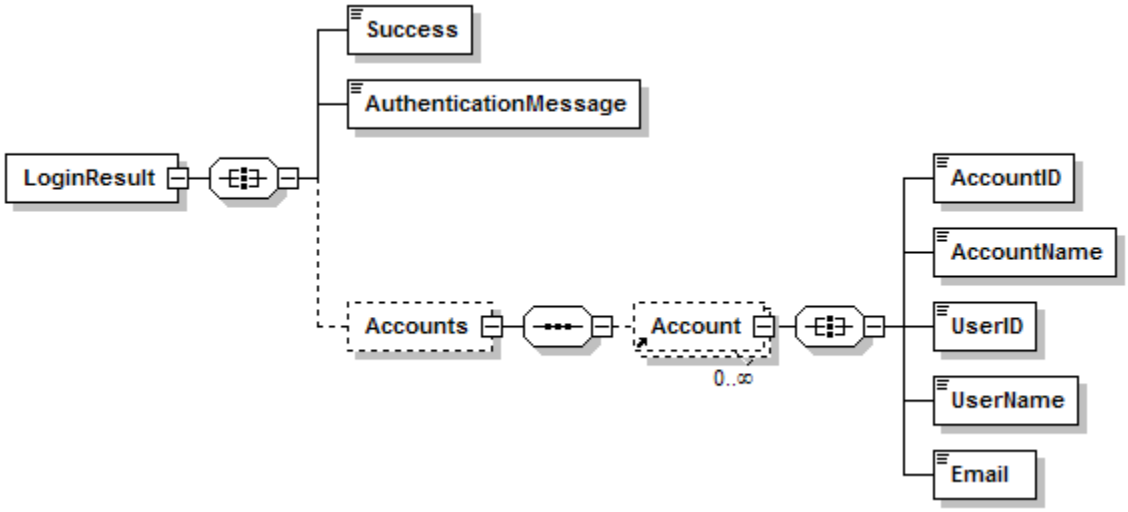
Name	Schema Type	Description
Email	string	Email to authenticate with. Place your bracketed integrators key in front of the email if you have received one from DocuSign. Sample: [integrator key]email
Password	string	Password to authenticate with.

Sample Request XML:
SOAPAction: "http://www.docusign.net/API/Credential/Login"

```
<?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>
    <Login xmlns="http://www.docusign.net/API/Credential">
      <Email>[Optional Integrator Key]string</Email>
      <Password>string</Password>
    </Login>
  </soap:Body>
</soap:Envelope>
```

LoginResult



Name	Schema Type	Description
AccountID	DSXId	Account ID of the valid DocuSign Connect API user.
AccountName	String	Account Name of the valid DocuSign Connect API user.
UserID	DSXId	User ID of the valid DocuSign Connect API user.
UserName	String	Username of the valid DocuSign Connect API user.
Email	String	Email of the valid DocuSign Connect API user.

Ping

Ping can be used to do a quick check to see if the DocuSign Credential service is available.

Schema

Sample Request XML:

SOAPAction: http://www.docusign.net/API/Credential/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:Body>
    <Ping xmlns="http://www.docusign.net/API/Credential" />
  </soap:Body>
</soap:Envelope>
```

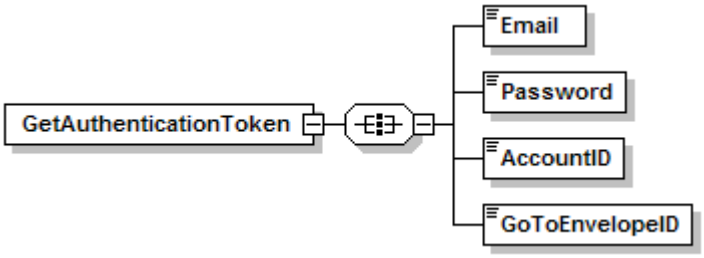
PingResult

Returns true if successful.

GetAuthenticationToken

This method can be used to get a onetime use URL with an authentication token to launch the DocuSign member system.

Schema



Name	Schema Type	Description
Email	string	Email to authenticate with. Place your bracketed integrators key in front of the email if you have received one from DocuSign. Sample: [integrator key]email
Password	string	Password to authenticate

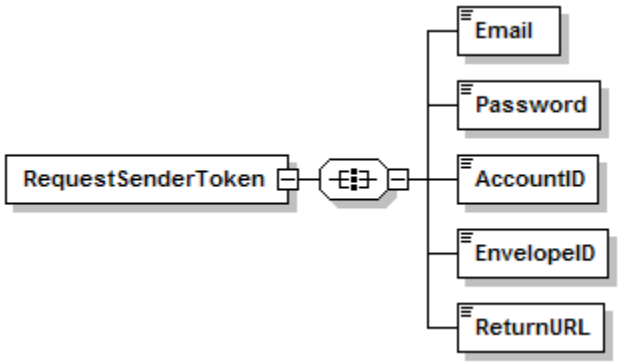
		with.
AccountID	string	Optional DocuSign account ID for the user. If left blank the default account ID will be used.
GoToEnvelopeID	String	Optional envelope ID. If provided the URL returned will send the user directly to viewing the envelope. If left blank the user will be taken to the member console.

This method returns a string that is the URL to access the DocuSign member system. This URL can be placed in a browser and you will be automatically logged into the member system.

RequestSenderToken

This method is used to get a onetime use login token that allows the user to be placed into the DocuSign sending wizard. Upon sending completion the user is returned to the return URL provided by the API application.

Schema



Name	Schema Type	Description
Email	string	Email to authenticate with. Place your bracketed integrators key in front of the email if you have received one from DocuSign. Sample:

		[integrator key]email
Password	string	Password to authenticate with.
EnvelopeID	string	Optional envelope ID. If provided the user will be placed into the envelope in process. If left blank the user will be placed into a new envelope for sending.
AccountID	String	DocuSign account ID to map the envelope to.
ReturnURL	String	URL to send the user to upon completion of the sending process. The URL will have an event passed to it as a query parameter. The parameter will be named "event", see "In-session sending events" below for description of the valid events. The DocuSing Envelope Id will also be returned in the "envelopeId" parameter.

In-session sending events

Events returned on the ReturnURL as a query string parameter. The event is sent in the parameter named "event". The ReturnURL is called with the event on any completion state of the envelope send. Valid events can be found in the schema type InSessionSendEvent in the DocuSign API XSD.

Event	**Description**
Send	User sends the envelope.
Save	User saves a draft of the envelope.
Cancel	User cancels the sending transaction. No envelope ID returned.
Error	Error performing the send.
SessionEnd	Sending session ended before the user completed.

RequestSenderToken rules and exceptions

- User_Does_Not_Exist_In_System – API user requesting the sending token URL does not have access to the system.
- Account_Does_Not_Exist_In_System – Account is not valid.
- Account_Lacks_Permissions – Account does not have valid permissions.

- User_Lacks_Permissions – user does not have valid permissions.
- User_Authentication_Failed – user/password is not valid.