



DOTS Email Validation (EV)

Developer's Guide

Version 2.1.0
April 14, 2011

Table of Contents

Introduction	3
Integration	3
Web Service Structure	4
Request Types	5
Request Type Analysis	5
XML Parsing	5
Operation Definitions	6
ValidateEmailFast/Full/NoNames	6
CorrectEmail	8
Errors	9
Frequently Asked Questions	11
Conclusion	13

Introduction

DOTS Email Validation 2 (EV) is an XML web service that provides validity and metadata information about an email address. The service provides common data elements such as syntax validity along with more refined data such as SMTP failures and vulgarity flags.

EV can provide instant data verification to websites or enhancement to contact lists.

Integration

Integrating EV into your application should be easy and straightforward. If you are using a common platform, such as asp, vb, C# .NET, PHP and others, Service Objects may already have sample code built that you can use:

http://www.serviceobjects.com/support/dots_example_code.asp

However, if you are using a common platform that does not already have sample code, you can ask Service Objects to build an example for you. Email support@serviceobjects.com for more details.

Web Service Structure

Web services provide a standard interface to encapsulate tricky business logic. They allow simple integration of applications via the web. Service Objects has followed web services best practices and come up with some of its own standards to ensure that its web services are as easy to integrate, and as accessible as possible.

The host path, or physical location of the web service is here:

<http://trial.serviceobjects.com/ev2/EmailValidation2.asmx>

The location of the WSDL, or Web Service Definition Language document, is here:

<http://trial.serviceobjects.com/ev2/EmailValidation2.asmx?WSDL>

(This is also accessible via the "Service Definition" link.)

The WSDL is an XML document that defines the interaction web service, meaning its inputs, outputs, operations, and the like. Most likely, you will have another tool read this WSDL and make the operations available to you in your application via some type of proxy class. Whenever your utilities or IDE asks for a WSDL path, you can provide this one.

Every web service has *operations* that it offers to subscribers. These operations, also called methods, contain different functionality and return different outputs. EmailValidation2 has three operations:

ValidateEmailFast - **Deprecated: use ValidateEmailFast_v2.**

ValidateEmailFast_v2 - Validates an email address returning all of our standard flags, indicating the quality of the address, as well as other data. This check is always fast and will not perform any real-time SMTP level checks; however, it may still provide SMTP level information via one of our other mechanisms.

ValidateEmailFull - **Deprecated: use ValidateEmailFull_v2.**

ValidateEmailFull_v2 - This operation has the same input/output as ValidateEmailFast_v2. However, if we are unable to provide all of the SMTP level flags, it may try to do real-time SMTP level checks. In this case it may return SMTP check information, but may take several seconds longer.

ValidateEmailFastNoNames - **Deprecated: use ValidateEmailFastNoNames_v2.**

ValidateEmailFastNoNames_v2 - This operation is identical to ValidateEmailFast but is even faster because it omits the PossibleFirstName and PossibleLastName fields.

CorrectEmail - This operation tries to correct an email that is syntactically incorrect, or one that fails to comply with the additional rules implied by its specific domain.

Request Types

EV is a public XML web service that supports SOAP(1.1 and 1.2), POST and GET request methods. A request type is just the type of web (HTTP) request to interact with our web services.

GET - All of the input data is in the query string appended to the url. The response is simple XML.

POST - The input parameters are in the body of the request instead of the query string. The response is simple XML.

SOAP - The input parameters are in an XML SOAP message contained within the body of the request. The response is an XML SOAP message.

Analysis of Request Types

GET is the easiest method to implement by hand because you just set up the URL and query string. It is also easy to debug because you can test the URL + query string in a web browser and see the output.

POST is probably the best method to implement by hand because you do not have to know the specifics of SOAP, and is a little cleaner than passing input parameters in the query string via GET.

SOAP is the best method if you are using a platform that supports SOAP. In many programming environments you can give your Integrated Development Environment (IDE) the URL to the WSDL of a web service (<http://trial.serviceobjects.com/ev2/EmailValidation2.asmx?WSDL>) and it will create a proxy class to help you interact with the web service. In this case you only have to create an instance of the proxy and use its methods. This completely abstracts the programmer from any complications like sending/receiving web requests/responses as well as any XML parsing.

XML Parsing

If you are not using an environment that creates a proxy class for you to use, then you will have to parse XML. If you do have a proxy, then it uses an XML parser behind the scenes for you. Although XML parsing can be done without a parser, most programming environments provide easy access to several standard ones. **We strongly recommend that you take advantage of an XML parser.** These parsers may take a few more minutes to integrate if the developer is not familiar with them, but will give your application an added level of security against improper parsing. Without them it is very difficult, even for skilled programmers to write robust code that can handle all cases of XML properly. Because we have very consistent XML you could get away without this extra precaution, but we suggest you use an XML parser anyway to ensure your application is of the highest quality.

Operation Definitions

This section defines the input, output and behavior of the web services in EV.

Important Note!

SMTP check level data may be unknown due to a real-time timeout, or company policy to limit SMTP requests to a domain. Our system is optimized to get around this limitation, but if you do receive some unknown flags, try the email again a few hours later there is a very good chance we will have acquired the information.

ValidateEmailFast/ValidateEmailFull

These are the main operations for validating an email. They run through several checks to determine if an email is valid. If one step fails, subsequent checks cannot be done because they have been logically eliminated: if the syntax on an email fails, neither the DNS nor the SMTP check is done.

Step 1: Syntax Check

The email is tested to verify that its structure is valid, such as an "@" symbol, a domain, and no odd characters that aren't allowed in email addresses. The rules specific to the domain are also checked. For example, if you input aa@aol.com it will pass the syntax check but fail the domain specific syntax check, and therefore we do not need to continue with the DNS or SMTP level checks.

Step 2: DNS Check

The DNS or domain name check verifies that the domain exists and has a valid MX record in order to direct mail to the correct mail server. If we return unknown SMTP checks, then the DNS check is the next best thing.

Step 3: SMTP Check

Once we obtain the location of the mail server from the MX record of a successful DNS check, we start communicating with the target mail server. No email is actually sent to the email address being verified. This step gives us three pieces of information. It tells us if the server is working, if it will accept any address, and if will it accept this specific address. One difficulty of email validation is the behavior of email servers, known as SMTP servers. Because of the growth of spam and email-mining tools, SMTP servers often respond to requests for information in defensive ways. SMTP servers will respond very slowly to information requests, provide unhelpful data, or sometimes no data at all. We have carefully crafted our system with this in mind, and therefore the data we return is still very accurate. However, sometimes we are still forced to return an unknown result for some of the SMTP level checks.

The ValidateEmailFast and ValidateEmailFast_v2 operations are usually faster but will occasionally return less robust data than the ValidateEmailFull and ValidateEmailFull_v2 operations.

ValidateEmailFast/Full/NoNames v2 Inputs

Name	Type	Description
EmailAddress	String	The email address you wish to validate.
LicenseKey	String	Your license key to use the service. Sign up for a free trial key at www.serviceobjects.com .

ValidateEmailFast/Full/NoNames v2 Outputs

If no errors are encountered an EmailValidateInfo element will be returned with the following information. If there is an error, an Error object will be returned (explained in next section).

Name	Type	Values	Description
EmailAddressIn	String	Varies	Echo of input: EmailAddress
Box	String	Varies	The part to the left of the '@' symbol. Also known as the local part or the username.
Domain	String	Varies	The part to the right of the '@' symbol. The location to which the email address will be sent.
IsRecognizedTLD	Boolean	T/F	Is the top level domain recognized by ICANN.
TopLevelDomain	String	Varies	The part after the rightmost '.'
TopLevelDomainDescription	String	Varies	Description of the Top Level Domain.
IsCCTLD	Boolean	T/F	Indicates if the TLD represents a specific country. For example .us implies United States.
IsSyntaxGood	Boolean	T/F	Indicates if the email is syntactically valid.
IsDomainSpecificSyntaxGood	Boolean	T/F	Indicates if the email is syntactically valid including the additional syntax rules specific to the email's domain.
IsDNSGood	Boolean	T/F	Indicates if the domain is registered and has at least one MX record.
IsSMTPServerGood	String	true false unknown	Indicates if the email's mail server is operational.
IsCatchAllDomain	String	true false unknown	Indicates if the mail server will accept mail to any box in that domain (*@domain).
IsSMTPMailBoxGood	String	true false unknown	Indicates if the mail server will accept mail to the specific box.
IsFree	String	true false unknown	Indicates if the domain of the email is a public-register domain, where users can sign

			up for email accounts for free.
IsEstablished	Boolean	T/F	Indicates if the email is known within large lists of bulk-marketing lists.
IsAlias	Boolean	T/F	Indicates if the email matches alias rules for different sites. For example, any email with a "+" in it at gmail.com is an alias. -OR- Indicates the email is disposable, for example anything@mailinator.com
IsBogus	Boolean	T/F	Indicates if the email is obviously bogus – such as a@a.com , asdf@asdf.com , or bgates@microsoft.com etc.
IsVulgar	Boolean	T/F	Indicates if the email address contains obvious vulgar words.
PossibleFirstName*	String	Varies	A common first name that was found in the email address. It is possible this is the user's first name, but not guaranteed.
PossibleLastName*	String	Varies	A common last name that was found in the email address. It is possible this is the user's last name, but not guaranteed.
IsSMSDomain**	Boolean	T/F	Indicates if the domain is a known Mail-to-SMS Gateway.
IsRole**	Boolean	T/F	Indicates if the email address appears to be a role that is designed to be anonymously managed by one or more persons.
IsGood**	Integer	0 1 2 3 4	An estimate on the validity of the email address. ^[1] 0 = Email is Good 1 = Email is Probably Good 2 = Unknown 3 = Email is Probably Bad 4 = Email is Bad
IsDeliverable**	String	true false unknown	Indicates if the mail server will accept mail for the given email address. ^[2]
IsBusinessAddress**	String	true false unknown	Indicates if the email address appears to be work related.

* **ValidateEmailFastNoNames** does not contain these output fields.

** Made available in version 2 operations.

[1] This value is dependent on the amount data that is available for the address. The service then analyzes the data to estimate the validity of the address.

[2] An **unknown** return value indicates that either there was not enough information available about the recipient's mail server to determine deliverability or the server is a catch-all. In general, catch-all mail servers will always initially accept an email message, even if the email address does not exist.

CorrectEmail

CorrectEmail Inputs

Name	Type	Description
EmailAddress	String	The email address you wish to correct.
LicenseKey	String	Your license key to use the service. Sign up for a free trial key at www.serviceobjects.com .

CorrectEmail Outputs

If no errors are encountered, an EmailCorrectInfo element will be returned with the following. If there is an error, an Error object will be returned (explained in next section).

Name	Type	Values	Description
EmailAddressIn	String	Varies	Echo of input: EmailAddress
EmailAddressOut	String	Varies	The corrected email address.

Errors

Anything that happens during a run of DOTS Email Validation 2 that causes it to be unable to finish its normal processing is an error. If an error occurs, something similar to the following will result instead of the normal/typical output:

Example:

```
<Error>
  <Type>Authorization</Type>
  <TypeCode>1</TypeCode>
  <Desc>Your license key does not work on this service.</Desc>
  <DescCode>8</DescCode>
</Error>
```

There are four error types described below.

Error Types

Type	TypeCode	Billable	Standard Across All Gen2 Web Services
Authorization	1	No	Yes
User Input	2	Yes	No
Service Objects Fatal	3	No	Yes
Domain Specific	4	Yes	No

Error type 1: Authorization

These are standard to all Generation 2 DOTS Web Services.

DescCode	Desc
0	Unknown authorization error.
1	Please provide a valid license key for this web service.
2	The daily allowable number of transactions for this license key has been exceeded.
3	The monthly allowable number of transactions for this license key has been exceeded.
4	The total allowable number of transactions for this license key has been exceeded.
5	There are not enough transactions available. Check your daily/monthly transaction limits.
6	This license key has not yet been activated.
7	This license key has expired.
8	Your license key does not work on this service.

Error type 2: User Input

User Input errors are caused when a user inputs an invalid value or fails to provide a certain input field altogether. If a developer creates a request and mistypes a parameter name, it will probably cause a User Input Error.

DescCode	Desc
1	"You must input an email address in the EmailAddress field."
2	"You must input a license key in the LicenseKey field."

Error type 3: Service Objects Fatal

The Desc will always be the same and the DescCode has no meaning. This is standard to all Generation 2 DOTS Web Services. This is a rare error that signals either a bug in the DOTS Email Validation 2 service, or a Network/Connectivity issue.

DescCode	Desc
1	Unhandled error. Please contact Service Objects.

Error type 4: Domain Specific

Domain specific errors represent the common errors seen in Service Objects services. Domain Specific errors indicate that the service completed successfully but the result is not good.

DescCode	Desc
1	"The email could not be corrected."
2	"The email could not be corrected into the form required by the specific domain."

Frequently Asked Questions

How is EV different from the original EV?

EV is one of our Gen2 services. It has more consistent and documented error handling mechanism and was developed with a newer framework. EV is much more reliable and should always return its findings within a few seconds. EV is much faster and smarter than the original EV. It uses a combination of systems to get as much detailed information, especially SMTP level checks, as possible. The system has background processes that are constantly gathering information. If we were not able to get a piece of information on the first request, it is likely a background process will grab that information and have it ready for you next time. EV combines all the functionality of its predecessor and more into just two new validation operations. There is also a completely new operation, CorrectEmail, which tries to correct addresses that are syntactically bad.

Why do I almost always get SMTP level info for most email addresses, but seldom get it for aol.com and hotmail.com addresses?

In order to get the SMTP box level check we must hit the mail server real-time, or somehow already know that it is good. The problem is that we have a important company policy that prevents us from making too many requests, within a certain time period, to the same mail server. So even though we make many requests to the common mail servers like aol.com, we do not always have enough available requests at that time to do the SMTP level checks.

I ran the same email twice with the ValidateEmailFast operation. The second request returned SMTP level information but the first time it did not. What happened?

Our system has background processes that constantly monitor new requests. We did not have the information on the first request, but the background process saw the request, and gathered the information so that it was instantly available the next time you requested it. Note: This does not mean we will always be able to do this, especially with common domains where we may have exceeded the maximum number of requests.

I have a bunch of emails that I absolutely must have the SMTP level information for, what should I do?

You should run them first against our ValidateEmailFull operation. Then wait an hour or so and take the ones that had unknown values in the SMTP level checks and run them against our ValidateEmailFast operation. This should get nearly all of them. Continue this until all the emails have data for the SMTP level checks. The more you submit an email, the better the chances are that our background processes will be able to process it. Our system improves with use.

How fast is the service?

The ValidateEmailFast operation should typically take under one second but on occasion may take a couple seconds. The ValidateEmailFull operation could take anywhere from under a second to around five seconds. The CorrectEmail operation should always take less than a second.

My email address fails EV's SMTP level checks, but I know it is a valid email address.

Your mail server may be down, or extremely slow. If we cannot get a response from the server there is nothing we can do but assume it is not valid. You can also send us the email address at support@serviceobjects.com and we can double check that the service is working properly.

Where do you get your information?

We get our data from a variety of public and proprietary sources. We also do real-time checks, as well as have background processes that are constantly getting new data.

Does your service hammer popular mail servers like aol.com with real-time SMTP requests? Do you run the risk of getting blocked from any mail servers?

We have a company policy to limit the number of requests per time period to a specific mail server. So to answer the question, yes we hit common mail servers often but never more than our limit. Our limit modestly set so that we do not overly disturb any mail servers. Because we carefully limit and monitor the requests to all mail servers we are very unlikely to ever get blocked from any mail server. In the unlikely event that we do get blocked, we have backup plans in place to insure that our service will continue to return top quality data.

I'm not technical, but I have a list of email addresses I want to validate. Can you run my list for me?

Yes! We run EV batches for users quite often, and would be happy to run your list once we discuss your needs. We offer a free trial batch for up to a hundred emails. Please contact sales@serviceobjects.com for details.

The sample code is giving strange errors or is crashing!

Most likely, the sample code cannot connect to Service Objects. Many environments will not allow you to connect out on port 80, or will clip out XML data from these requests/responses. The easiest way to check for this is to open a browser on the machine running the sample code. In your browser, navigate to: <http://trial.serviceobjects.com/ev2/emailvalidation2.aspx>. Then test one of the operations with your trial key. If you get a browser error, or get no data back, then the sample code isn't able to connect, either. Contact your systems administrator to resolve why you are not able to connect to Service Objects.

Why are email addresses for Hotmail, Yahoo, and other domains commonly marked as a one, "Probably Good", even though I know they are valid?

Email providers like Hotmail and Yahoo have catch-all domains. They will accept requests for any email address, even for addresses that do not exist. It is difficult to be certain on whether an email address with a catch-all domain is truly valid or not. This uncertainty is generally the reason why an email address will be marked as either one (Probably Good) or three (Probably Bad).

Conclusion

Service Objects is pleased to offer you a free trial of DOTS Email Validation 2.

Sign up today for a free trial at:

http://www.serviceobjects.com/products/dots_email_validation.asp

Technical questions or concerns can be directed to support@serviceobjects.com.

If you are interested in purchasing DOTS Email Validation 2, please contact:

sales@serviceobjects.com.

We welcome your feedback! Please do not hesitate to let us know what you think of our web services, documentation, or customer support.

www.serviceobjects.com