Visma EasyCruit Web Services API Guide

Acronyms and Definitions

The purpose of this section is to define terminology used in, and as an aid to understanding, this article.

Acronym	Description
WS	Web Service, is interoperable software system designed to support Machine to Machine interaction over a network.
XML	Extensible Markup Language, is a general-purpose markup language. It is classified as an extensible language because it allows its users to define their own tags. Its primary purpose is to facilitate the sharing of structured data across different information systems.
SOAP	Simple Object Access protocol is a protocol for exchanging XML-based messages over computer networks, normally using HTTP/HTTPS.
WSDL	Web Services Description Language, web service description file that is being used to consume web services in applications while communicating with the web service.
XSD	XML Schema Definition, An XSD defines a type of XML document in terms of constraints upon what elements and attributes may appear their relationship to each other, what types of data may be in them, and other things.
MIME	Base64 MIME (Multipurpose Internet Mail Extensions) specification, lists "base64" as one of several binary-to-text encoding schemes. MIME's base64 encoding is based on that of the RFC 1421 version of PEM: it uses the same 64-character alphabet and encoding mechanism as PEM, and uses the "=" symbol for output padding in the same way. Base64 is a positional notation using a base of 64. It is the largest power-of two bases that can be represented using single printable ASCII characters. This has led to its use as a transfer encoding for e-mail among other things. For more information about MIME Base64, RFC 2045 - http://tools.ietf.org/html/rfc2045

XML Character entity encoding	A character entity reference is a placeholder that represents that entity. It consists of the entity's name preceded by an ampersand ("&") and followed by a semicolon (";"). XML has five predeclared entities: • & & ampersand • < < less than • > > greater than • ' ' apostrophe • " " quotation mark
UTF-8	UTF-8 (8-bit UCS/Unicode Transformation Format) is a variable-length character encoding for Unicode.

Visma EasyCruit Web Service

This section describes the EasyCruit WS interface. It contains a general discussion of why and how you may want to use this service as well as a detailed reference section enabling you to tie together this functionality for your specific use case.

Overview

The Visma EasyCruit web application provides a customer access to their data via a WS interface. This interface can be used by customers to programmatically retrieve their data. The data can then be further processed or imported into archiving, ERP or HRM systems.

The Visma EasyCruit WS interface is exposed in a WSDL file. This means a software development kit may be used to automatically generate code to interact with the interface.

Accessing this interface involves sending a SOAP message containing an XML document over HTTPS to the Visma EasyCruit system. The secure HTTPS protocol guarantees the data cannot be read in transfer by third parties.

Usage of industry standards like SOAP, XML and WSDL leveraging excellent tool support, hence easy integration.

Purpose and Benefit of the EasyCruit Web Service Interface

A customer has project and candidate data stored within the Visma EasyCruit system. This data represents an important part of the HR business. A customer may want to be able to retrieve that data for further processing or storage. They might want to store it within an archiving system or they might want to export it to an enterprise resource planning system or a human resources management system.

The WS functionality enables a customer to do just this.

Data in XML format

Now what form are customers offered their data in? All data comes in XML form, which is an industry standard. The XML is retrieved using the SOAP message format over HTTPS, which is also an industry standard. The service is laid out in a WSDL file, which is another industry standard. This facilitates automatic generation of program code to retrieve the data. To profit from this, a customer needs a suitable software development kit like, for instance, Microsoft Visual Studio.

Once a customer has retrieved the XML, it can be put to all sorts of uses. It can simply be stored for archiving purposes. XML being a malleable format, the data can be submitted to transformations and made suitable for further processing in other applications a customer may run. The input to other applications could be XML or another format. With XML and transformations, there are no limits to what uses the data can be put to.

An Interrelated Set of Methods

The Visma EasyCruit web service interface consists of a set of named methods. These methods are interrelated. This is done in such a way that the information retrieved by invoking one method may be used as input to another, related method.

There are methods to retrieve lists of projects and lists of candidates. This provides an overview over the data available. A particular project or candidate can then be selected for a full export. This retrieves all of the data associated with the item. This combination of comprehensive list and full detail ensures a customer can retrieve all of their data.

This is best made clear giving an example. There is one method called CandidateIdList, invoking which a customer may retrieve a list of candidate record identifiers. All of the data stored for a given candidate may then be retrieved by supplying the corresponding identifier to another method, called CandidateExport.

Functions in detail

This section lists all methods available as part of the Visma EasyCruit WS interface. The semantics of what individual methods do and how they relate to others are explained here. The exact XML format for input and output, however, is not explained here. It may be explored by studying the XML Schema files referenced from the WSDL file. You should not normally need to deal with this aspect yourself, though, as your tools should do that part of the job.

WSDL file - https://www.easycruit.com/wsdl/ws/EasyCruit.wsdl

Common Authentication Parameter

Each and every method requires you to send authentication data (defined in the SOAP HEAD) along with your specific request. No request is handled without successful authentication and authorization of the incoming request's author. This authentication data consists of customer name, user name and password. This is exactly the same kind of data that is required when logging in to the Visma EasyCruit web application via the login form. Note, however, that in order to access the web service you need to supply the credentials of a web service user.

CandidateSearch

CandidateSearch provides you with a list of candidates within the search parameters supplied. The list contains a candidate's ID, given name, middle name, family name, date added, date modified, date hired and which category (candidate, cv_candidate, cv_prospect, co-worker) he/she are placed in.

XSD files:

- CandidateSearchRequest.xsd:
  https://www.easycruit.com/wsdl/ws/CandidateSearchRequest.xsd
- CandidateSearchResponse.xsd:
  https://www.easycruit.com/wsdl/ws/CandidateSearchResponse.xsd

Parameters:

- DateAddedFrom:
  ISO formatted date YYYY-MM-DD
- DateAddedTo:
  ISO formatted date YYYY-MM-DD
- DateModifiedFrom:
  ISO formatted date YYYY-MM-DD
- DateModifiedTo:
  ISO formatted date YYYY-MM-DD
- Category:
  String enumeration (candidate, cv_candidate, cv_prospect, co-worker)

CandidateExport

CandidateExport returns a string that is a character entity encoded XML document. You would need to decode this and parse the XML. Further description of the XML is in Candidate.xsd.

XSD files:

- CandidateExportRequest.xsd:
  https://www.easycruit.com/wsdl/ws/CandidateExportRequest.xsd
- CandidateExportResponse.xsd:
  https://www.easycruit.com/wsdl/ws/CandidateExportResponse.xsd

- Candidate.xsd:
  https://www.easycruit.com/wsdl/ws/Candidate.xsd

Parameters:

- CandidateID:
  Integer [required]
- Language:
  Two character iso [optional]

CandidateImport

CandidateImport reads a character entity encoded XML document sent as a string, and adds it to the customer CV-Pool. Further description of the XML is in Candidate.xsd.

XSD files:

- CandidateImportRequest.xsd:
  https://www.easycruit.com/wsdl/ws/CandidateImportRequest.xsd
- CandidateImportResponse.xsd:
  https://www.easycruit.com/wsdl/ws/CandidateImportResponse.xsd
- Candidate.xsd:
  https://www.easycruit.com/wsdl/ws/Candidate.xsd

Parameters:

- Candidate:
  String (see https://www.easycruit.com/wsdl/ws/Candidate.xsd for details) [required]

CandidateUpdate

Same as CandidateImport, but the candidate's ID are required in the XML document.

XSD files:

- CandidateUpdateRequest.xsd:
  https://www.easycruit.com/wsdl/ws/CandidateUpdateRequest.xsd
- CandidateUpdateResponse.xsd:
  https://www.easycruit.com/wsdl/ws/CandidateUpdateResponse.xsd
- Candidate.xsd:
  https://www.easycruit.com/wsdl/ws/Candidate.xsd

Parameters:

- Candidate:
  String (see https://www.easycruit.com/wsdl/ws/Candidate.xsd for details) [required]

AddCandidateToRecruitingProject

XSD files:

- AddCandidateToRecruitingProjectRequest:
  https://www.easycruit.com/wsdl/ws/AddCandidateToRecruitingProjectRequest.xsd
- AddCandidateToRecruitingProjectResponse:
  https://www.easycruit.com/wsdl/ws/AddCandidateToRecruitingProjectResponse.xsd

Parameters:

- CandidateId:
  Integer [required]
- RecruitingProjectId:
  Integer [required]
- DepartmentId:
  integer [required]
- RecruiterId:
  Integer [optional]

DocumentUpload

DocumentUpload could be used to attach a CV or picture to a candidate record. The file cannot be larger than 2 MB, and a file of type **CV** can only be in one of the following formats:

doc, pdf, ppt, rtf, sxw, sdw, txt, text, wpd, xls, lwp, WKS, WK1, WK3, WK4, docx, xlsx, pptx, jpg, jpeg, png, tif, tiff, gif, bmp

File of type **Picture** can only be in one of the following formats:

jpg, jpeg, png, tif, tiff, gif, bmp

XSD files:

- DocumentUploadRequest.xsd:
  https://www.easycruit.com/wsdl/ws/DocumentUploadRequest.xsd
- DocumentUplaodResponse.xsd:
  https://www.easycruit.com/wsdl/ws/DocumentUploadRequest.xsd

Parameters:

- CandidateId:
  Integer [required]
- DocumentType:
  String enumeration (cv, picture) [required]

- DocumentName:
  String [required]
- Document:
  String (MIME Base64 encoded version of the document) [required]

DocumentDownload

XSD files:

- DocumentDownloadRequest.xsd:
  https://www.easycruit.com/wsdl/ws/DocumentDownloadRequest.xsd
- DocumentDownloadResponse.xsd:
  https://www.easycruit.com/wsdl/ws/DocumentDownloadResponse.xsd

Parameters:

- CandidateId:
  Integer [required]
- DocumentName:
  String [required]

CandidateValidateUsername

CandidateValidateUsername are used to check if a username are available and properly formatted.

XSD files:

- CandidateValidateUsernameRequest.xsd:
  https://www.easycruit.com/wsdl/ws/CandidateValidateUsernameRequest.xsd
- CandidateValidateUsernameResponse.xsd:
  https://www.easycruit.com/wsdl/ws/CandidateValidateUsernameResponse.xsd

Parameters:

- CandidateUsername:
  String [required]

CVSetup

CVSetup returns the available data fields for the specified customer; this could be used to populate customer specific questions.

XSD files:

- CVSetupRequest.xsd:
  https://www.easycruit.com/wsdl/ws/CVSetupRequest.xsd
- CVSetupResponse.xsd:
  https://www.easycruit.com/wsdl/ws/CVSetupResponse.xsd

Parameters:

- Language

Events

Events returns available events based on the specified parameters.

XSD files:

- EventsRequest.xsd:
  https://www.easycruit.com/wsdl/ws/EventsRequest.xsd
- EventsResponse.xsd:
  https://www.easycruit.com/wsdl/ws/EventsResponse.xsd

Parameters:

- EventModule:
  String [required] e.g. cv
- EventClass:
  String [required] e.g. hired
- HandledByIdentifier:
  String [optional] e.g. my_web_service_client
- HandledByTracked:
  Boolean [optional] keeps track of events you have sent earlier
- HandledByStatus:
  Enumeration (all, processed, new) [optional] what to display, default are new

MONO .NET C# example

```
using System;
class ec_soap
{
     public static void Main(string[] args)
     {
          easycruit Service = new easycruit();
          Service.AuthHeaderValue = new AuthHeader();
          Service.AuthHeaderValue.Customer = "customer";
          Service.AuthHeaderValue.Username = "username";
          Service.AuthHeaderValue.Password = "password";
          /** Candidate Export **/
          MessageCandidateExport CandidateID = new MessageCandidateExport();
          CandidateID.CandidateId = "123456789";
```

```
            CandidateID.Language = "gb";
            MessageCandidateExportResponse CandidateResponse =
            Service.CandidateExport(CandidateID);
            Console.WriteLine(CandidateResponse.Candidate);
      }
}
```

To compile the above example you need to download the WSDL file from Visma EasyCruit and compile it.

WSDL URL

https://www.easycruit.com/wsdl/ws/EasyCruit.wsdl

Command line example of compilation

```
wsdl easycruit.wsdl
mcs /target:library easycruit.cs r:System.Web.Services
mcs /r:easycruit.dll ec_soap.cs
```

PERL SOAP::LITE example

```
my $service =
SOAP::Lite>service('https://www.easycruit.com/wsdl/ws/EasyCruit.wsdl');

my $candidate =
$service>CandidateExport(SOAP::Data>name('ec:CandidateId',123456789),SOAP::Da
ta>name('ec:Language','gb'),
      SOAP::Header>name(
            AuthHeader => [
                  SOAP::Data>name('Customer', 'customer'),
                  SOAP::Data>name('Username', 'username'),
                  SOAP::Data>name('Password', 'password')
                  ]
      )
);

print ($candidate gt '' ? $candidate : 'Unable to retrieve data..');
```

RAW SOAP request example

```
<?xml version="1.0" encoding="UTF8"?>
<soap:Envelope xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchemainstance"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns="urn:easycruit">
      <soap:Header>
            <AuthHeader>
                  <Customer>customer</Customer>
                  <Username>username</Username>
                  <Password>password</Password>
            </AuthHeader>
```

```
     </soap:Header>
     <soap:Body>
          <CandidateExport>
               <Language>no</Language><CandidateId>123456789</CandidateId>
          </CandidateExport>
     </soap:Body>
</soap:Envelope>
```

RAW SOAP response example

```
<?xml version="1.0" encoding="UTF8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
     <soap:Body>
          <CandidateExportResponse xmlns="urn:easycruit">
               <Candidate>DATA</Candidate>
          </CandidateExportResponse>
     </soap:Body>
</soap:Envelope>
```

The DATA for this example needs to be decoded, and the resulting XML is based on this schema: https://www.easycruit.com/wsdl/ws/Candidate.xsd