

# TalentHook ATS Export Integration

---

## Introduction

TalentHook finds and retrieves candidate information and resumes from the Internet. Recruiters are able to export a candidate's data and resume to their ATS (Applicant Tracking System) of choice, using TalentHook's "ATS Export" functionality. This document describes the data structures and protocols TalentHook employees to communicate with these systems.

To assist in an efficient integration process, we are providing some sample code files, which may prove useful in parsing the XML:

You can download the files (zipped) described below, here:

<http://talenthook.com/ats/ATSExportSamples.zip>

Sample server-side code in VBScript:

*candidate\_import\_vbscript.asp*

Sample server-side code in JavaScript:

*candidate\_import\_jscript.asp*

Response Envelope XML Structure required from server:

*hrxml\_response\_envelope.xml*

Candidate data packet sent by TalentHook:

*candidate\_import\_example\_request.xml*

## Overview

When the user requests to export a candidate from TalentHook to their ATS system, the actual transfer of data takes place via a single HTTP transaction. TalentHook will submit the candidate data as an XML structure via an HTTP POST action to the ATS system's specified URL. The ATS system will process the submission and then return an XML response structure in the content body of the HTTP response.

## Export XML Structure

### JobPositionSeeker

TalentHook passes candidate data to a receiving server-side process using the JobPositionSeeker data structure as defined by the HR-XML consortium. The specification version at the time of this document creation is 1.1. Complete HR-XML specifications and DTDs may be found at <http://hr-xml.org>. The following table describes how the TalentHook data fields are mapped to the JobPositionSeeker structure.

<b>JobPositonSeeker Element</b>	<b>TalentHook Data Field</b>	<b>Notes</b>
JobPositionSeeker. JobPositionApplication. JobPositionPostingId	Requisition number	The TalentHook job req. number can be used to correlate the candidate record between the two systems.
JobPositionSeeker.PersonalData. PersonName.GivenName	Candidate's first name	This first name is a user-editable field during export.
JobPositionSeeker.PersonalData. PersonName.MiddleName	Candidate's middle name	This middle name is a user-editable field during export.
JobPositionSeeker.PersonalData. PersonName.FamilyName	Candidates last name	This last name is a user-editable field during export.
JobPositionSeeker.PersonalData. PersonName.For mattedName	Full name	TalentHook stores only the candidate's full name
JobPositionSeeker.PersonalData. PostalAddress.DeliveryAddress. AddressLine	Street address	
JobPositionSeeker.PersonalData. PostalAddress.Municipality	City	
JobPositionSeeker.PersonalData. PostalAddress.Region	State	In the abbreviated form (e.g., CA, AK, WY, etc)
JobPositionSeeker.PersonalData. PostalAddress.CountryCode	Country	
JobPositionSeeker.PersonalData. PostalAddress.PostalCode	Postal / Zip code	
JobPositionSeeker.PersonalData. VoiceNumber[@label= home]. TelNumber	Home phone number	
JobPositionSeeker.PersonalData. VoiceNumber[@label=work]. TelNumber	Work phone number	
JobPositionSeeker.PersonalData. VoiceNumber[@label=mobile]. TelNumber	Mobile phone	
JobPositionSeeker.PersonalData. FaxNumber. TelNumber	Fax phone number	
JobPositionSeeker.PersonalData. PositionTitle	Posting description	This description data comes from candidate's resume posting info.
JobPositionSeeker.Profile. AvailabilityDate.Date	Posted / available date	Date may be availability or the last time the resume was posted or modified.
JobPositionSeeker.PersonalData. E-mail	Email address	

JobPositionSeeker.PersonalData.WebSiteTitle	Website address	
JobPositionSeeker.Resume.StructuredResume.SummaryText	Recruiter's notes	Notes that the recruiter has entered about the imported candidate
JobPositionSeeker.Resume.TextOrNonXMLResume.TextResume	Resume data	Resume can be sent as HTML or plain text (as predefined for each ATS)
<b>Additional non-candidate related, supplier identification fields</b>		
JobPositionSeeker.Supplier.SupplierName	"TalentHook"	Identifies the submitting system

## Request Envelope

The JobPositionSeeker structure is wrapped in an HR-XML Envelope for delivery to the server. Similar in nature to a SOAP envelope, this structure provides additional information required by the receiving process, including authentication data. Here is an example of the Request Envelope structure.

```
<Envelope version = "01.00">
  <Sender>
    <Id>recruiter_login /Id>
    <Id2>recruiter_login</Id2>
    <Credential>recruiter_password</Credential>
  </Sender>

  <Recipient>
    <Id>target_url@ats.com</Id>
  </Recipient>

  <TransactInfo transactType = "request">
    <TransactId><!--future use --></TransactId>
    <TimeStamp>2000-10-09T14:14:11Z</TimeStamp>
  </TransactInfo>

  <Packet>
    <PacketInfo packetType = "request">
      <PacketId><!--future use--></PacketId>
      <Action><!--optional--></Action>
      <Manifest>jobpositionseeker.dtd</Manifest>
    </PacketInfo>

    <Payload><![CDATA[
      <JobPositionSeeker>
        .....
      </JobPositionSeeker>
    ]]></Payload>

  </Packet>
```

```
</Envelope>
```

This Request Envelope includes information about the sender, receiver and action to be taken. The following table describes how the primary Envelope elements are populated.

Envelope Element	Data	Notes
Envelope.Sender.Id	TalentHook user's unique ATS login.	
Envelope.Sender.Credential	TalentHook user's unique ATS password.	
Envelope.Recipient.Id	Target URL of receiving process	
Envelope.TransactInfo.TimeStamp	Time of transaction	
Envelope.Packet.Manifest	XML data type definition	This would be for the JobPositionSeeker if included
Envelope.Packet.Action	Optional ATS definable action code	
Envelope.Packet.Payload	Data content	Consists of nested JobPositionSubmission and JobPositionSeeker

## Server Response

Once the Request Envelope has been submitted, the transaction is complete when the receiving process returns the populated HR-XML Response Envelope packet. An example of the Response Envelope is shown below.

```
<Envelope version = "01.00">
  <Sender>
    <Id> recruiter_login</Id>
    <Id2>recruiter_login</Id2>
    <Credential>recruiter_password</Credential>
  </Sender>
  <Recipient>
    <Id>target_url@ats.com</Id>
  </Recipient>
  <TransactInfo transactType = "response">
    <TransactId><!--optional--></TransactId>
```

```

    <TimeStamp><!--optional--></TimeStamp>
  </TransactInfo>

  <Packet>
    <PacketInfo packetType = "response">
      <PacketId><!--future use--></PacketId>
      <Manifest/>
      <Status>
        <Code>200</Code>
        <ShortDescription>Success</ShortDescription>
      </Status>
    </PacketInfo>
    <Payload/>
  </Packet>
</Envelope>

```

The only required field in the response packet is the `Status/Code`. The following table lists the valid return codes for these two elements.

Code	Description
200	<b>Success:</b> The request was successfully received and processed.
210	<b>Duplicate Record.</b> Request was successfully received, however the submitted candidate was already in the database.
400	<b>Bad Request:</b> Request could not be processed because it was not understood. Either the Action was not valid or the XML was malformed.
401/403/407	<b>Authentication Error:</b> Client sent invalid authentication credentials.
500	<b>Internal Error:</b> Request could not be processed due to an internal error.
405	<b>HRXMLRequest Error:</b> Request could not be parsed.