Use the Bullhorn SOAP API to Work with Candidates

Introduction

This tutorial is for developers who create custom applications that use the Bullhorn web services APIs. The tutorial describes how to work with the **Candidate** entity in the Bullhorn system. Code samples are provided in the tutorial for C#, Java, and PHP.

You learn how to:

- Create a candidate in the Bullhorn system by using the API.
- Add candidate work history.
- Add candidate education.
- Associate categories with a candidate.
- Associate skills with a candidate.
- Update the candidate record.
- Delete the candidate record.

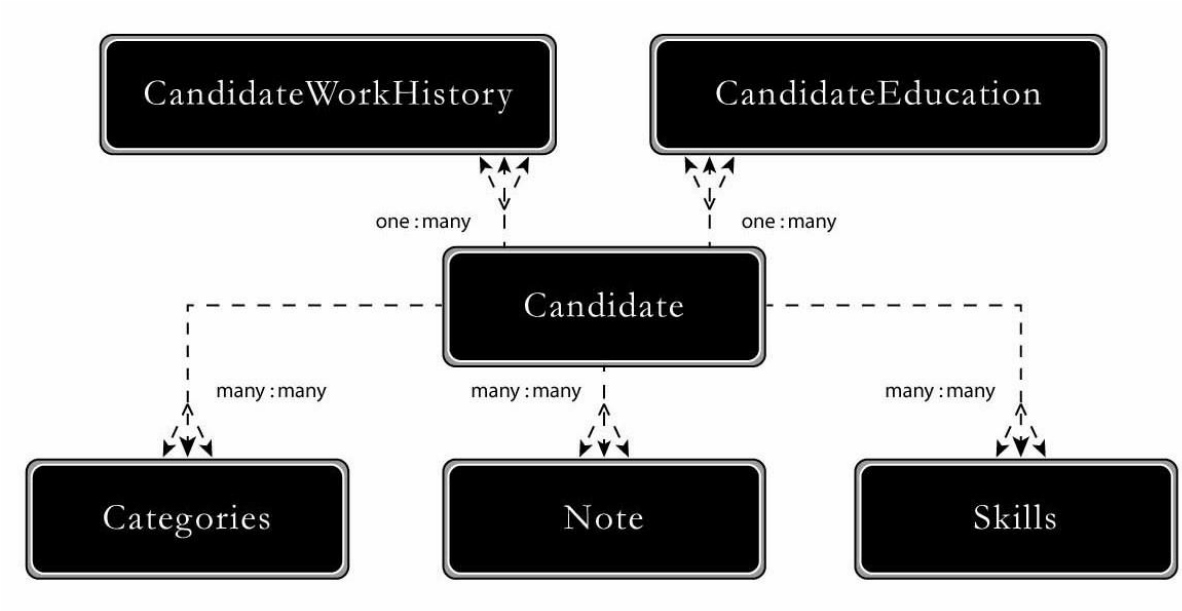
Prerequisites

- Knowledge of C#/ASP.NET, Java programming, or PHP
- Knowledge of HTML
- It is recommended that you familiarize yourself with one of the Getting Started tutorials

Understanding the Candidate entity

The **Candidate** entity is one of the primary entities managed by the Bullhorn system. It represents the contact details for a potential employee or contractor, and details about their profile, work history, education history, and employment information.

The following figure depicts the relationship between the **Candidate** entity and its most closely related entities.



Creating a candidate

This section walks through sample code for creating a new candidate. You learn how to create an instance of the Candidate DTO, set properties on the DTO, and save the candidate in the database.
You learn to perform the following tasks in C#, Java, and PHP.

- Create a session to the Bullhorn web service using your credentials
- Create an instance of the Candidate DTO
- Set the properties in the Candidate DTO
- Save the Candidate DTO.

Creating a session

You must create a session for all calls to the Bullhorn SOAP API. To create a session to the Bullhorn system in the web application, you require a username, password, and API key. For a detailed explanation of creating sessions, see the GettingStarted articles.

Java code

```
private static final QName SERVICE_NAME = new
       QName("http://apiservice.bullhorn.com/", "ApiService");

String username = "yourusername";
String password = "yourpassword";
String apiKey = "yourapikey";
String wsdlUrl = "https://api.bullhornstaffing.com/webservices-1.1/?wsdl";
URL serviceUrl = new  URL(ApiService_Service.class.getResource("."), wsdlUrl);
ApiService  service = new ApiService_Service(serviceUrl, SERVICE_NAME).getApiServicePort();

ApiSession currentSession =  service.startSession(username, password, apiKey);
```

C# code

```
String username = "yourusername";
String password = "yourpassword";
String apiKey = "yourapikey";
ApiService service = new ApiService();
apiSession currentSession = new apiSession();
currentSession = service.startSession(username,  password, apiKey);
```

PHP code

```
$params = array(
  'trace'  => 1,
  'soap_version'  => SOAP_1_1);
$BHClient = new SoapClient("https://api.bullhornstaffing.com/webservices-
1.1/?wsdl",$params);
$username = "yourusername";
$password = "yourpassword";
$apiKey = "yourapikey";

session_request = new stdClass();
$session_request->username = $username;
$session_request->password = $password;
$session_request->apiKey = $apiKey;
$API_session =  $BHClient->startSession($session_request);
$API_currentSession = $API_session->return;
```

Tip: Refreshing sessions
Sessions have a short expiration and must be refreshed with each call to the Bullhorn SOAP API. Each call to the

Bullhorn SOAP API requires a valid session object, and each response returns a new session object. Store that new session object in the current session variable so you always have a recently refreshed value.

Use the session object that is returned as part of the response in the next call that is made to the API, because previous session objects expire in 5 minutes.

Creating an instance of a Candidate DTO and setting properties

To create a candidate, you must create an instance of the Candidate DTO class and set properties for the candidate. There are many fields associated with a candidate. This tutorial focuses on the most commonly used ones. For the complete list, see the reference documentation.

The following table lists some of the key properties for a candidate.

Field Name	Field Description
firstName	First name of the candidate
lastName	Last name of the candidate
name	Full name of the candidate
phone	Phone number of the candidate
email	Email address of the candidate
address	Address of the candidate. This is a child object that sets the street, city, state, and ZIP code.
username	The username value must be globally unique across all Bullhorn customers. It is recommended you combine the first initial/last name with a random number or use a GUID.
status	Status of the candidate
description	The candidate's resume.
ownerID	This is the ID of the CorporateUser that is designated as the primary owner of the candidate record.
userTypeID	Most Bullhorn clients use the standard Bullhorn Candidate userType (which has an ID of 35), but some use a customized version. To identify the user types for your system, call the getUserTypes operation for the Candidate entity.
categoryID	This field must be provided to create a candidate, but also must be associated using the associate call.

Note:
Categories are descriptive classifications that are set up by each Bullhorn customer. To get the list of categories for a corporation, you create a query with an entity name of Category, and set the where clause with enabled=true.
In the code samples that follow, you use the category ID of 45 which is retrieved using the query operation.

Java code

```
Address candidateAddress = new  Address();
  candidateAddress.setAddress1("1234  Fictional Street");
  candidateAddress.setAddress2("Apt.  4");
  candidateAddress.setCity("Boulder");
  candidateAddress.setState("Colorado");
  candidateAddress.setZip("12345");
  candidateAddress.setCountryID(1);

CandidateDto candidateDto = new  CandidateDto();
candidateDto.setFirstName("Delvan");
candidateDto.setLastName("Minke");
candidateDto.setName("Delvan  Minke");
candidateDto.setPhone("4046537890");
candidateDto.setEmail("dminke@test.com");
candidateDto.setAddress(candidateAddress);
/* To make the username unique,  append the first/last name with a random number. You can
use any mechanism for  generating the random number. For ease, the Random class in Java
is being used  in this example. */
Random generator = new Random();
```

```
int r = generator.nextInt();

candidateDto.setUsername("minke"+r);
candidateDto.setStatus("Available");

/* The description field contains  the candidate's resume in HTML format. */
candidateDto.setDescription("{HTML  text of the candidate's resume}");

/* In this example, we make the  API user the owner of the candidate, but in other
situations
 it may be the  owner of the job to which the candidate applied or a shared user account.
*/
candidateDto.setOwnerID(currentSession.getUserId());
candidateDto.setUserTypeID(35);

/* When you create a Candidate,  it must have a category. Here we use 45. For a full list
of the categories used  in your system, use the query operation to retrieve instances of
the
Category  entity. */
candidateDto.setCategoryID(45);
candidateDto.setPassword("password");
```

C# code

```
address candidateAddress = new address();
candidateAddress.address1 ="1234 Fictional  Street";
candidateAddress.address2 ="Apt. 4";
candidateAddress.city = "Boulder";
candidateAddress.state = "CO";
candidateAddress.zip = "12345";
candidateAddress.countryID = 1;
candidateAddress.countryIDSpecified = true;
candidateDto newCandidate = new candidateDto();
newCandidate.firstName = "Delvan";
newCandidate.lastName = "Minke";
newCandidate.name = "Delvan Minke";
newCandidate.phone = "4046537890";
newCandidate.email = "dminke@test.com";
newCandidate.address = candidateAddress;

/* To make the username unique,  append the first/last name with a random number. You can
use any mechanism for  generating the random number. For ease, the Random class in C#
is being used in  this example. */
Random generator = new Random();
int num = generator.Next();
newCandidate.username = "minke"+num;

newCandidate.status = "Available";
newCandidate.description = "HTML text of the candidate resume";
newCandidate.ownerID = currentSession.userId;
newCandidate.ownerIDSpecified = true;
newCandidate.userTypeID = 35;
newCandidate.userTypeIDSpecified = true;
newCandidate.categoryID = 45;
newCandidate.categoryIDSpecified = true;
newCandidate.password = "password";
```

PHP code

```
$address_array = array(
  'address1'  => '1234 Fictional Street',
  'city'  => 'Boulder',
  'state'  => 'CO',
  'zip'  => '12345',
  'countryID'  => 1
  );
/* To make the username unique,  append the first/last name with a random number. You can
use
any mechanism for  generating the random number. For ease, the rand class in PHP is being
used
in  this example. */
$random_num = rand(1000,9999);
$username = 'bjennings'.$random_num;
$candidate_array = array(
  'firstName'  => 'Delvan',
  'lastName'  => 'Minke',
  'name'  => 'Delvan Minke',
  'phone'  => '505-555-5555',
  'email'  => 'dminke@test.com',
  'address'  => $address_array,
  'username'  => $username,
  'status'  => 'Available',
  'description'  => 'HTML text of the candidate resume',
  'ownerID'  => $API_currentSession->userId,
  'userTypeID'  => 35,
  'categoryID  => 45,
  'password'  => 'password'
);
```

Saving the Candidate DTO

After you create a candidate, you must save it to persist the record in the Bullhorn database. The save operation is required to create the Candidate entity instance and transmit the Candidate DTO to the Bullhorn server. If the DTO includes a value for the entity's primary key or ID, the entity instance represented by that ID is updated with the values you provide. If the DTO that you send does not include a value for the primary key, a new entity instance is created, and the response message includes the new entity's ID.

Java code

```
ApiSaveResult results =  service.save(currentSession, candidateDto);
// Create a local variable for  the saved candidate DTO to be used in later sections
CandidateDto savedCandidate = (CandidateDto)results.getDto();
```

The API returns a new session object. Refresh the session after calling the save operation.

```
currentSession = results.getSession();
```

C# code

```
apiSaveResult results =  service.save(currentSession, newCandidate);
currentSession = results.session;
// Create a local variable for  the saved candidate DTO to be used in later sections
candidateDto savedCandidate =  (candidateDto)results.dto;
```

PHP code

```
$SOAP_dto = new SoapVar($candidate_array,  SOAP_ENC_OBJECT,
   "candidateDto",  "http://candidate.entity.bullhorn.com/");
$request_array = array(
```

```
  'session'  => $API_currentSession,
  'dto'  => $SOAP_dto
  );

$SOAP_request = new SoapVar($request_array,  SOAP_ENC_OBJECT, "save",
"http://save.apiservice.bullhorn.com/");
try{
  $saveResponse  = $BHClient->save($SOAP_request);
  } catch (SoapFault $fault) {
  var_dump($BHClient->_getLastRequest());
  die($fault->faultstring);
  }

$API_currentSession =  $saveResponse->return->session;
$savedCandidate =  $saveResponse->return->dto;
```

Adding a CandidateWorkHistory entity

The CandidateWorkHistory entity represents a single entry in the work history section of a candidate's resume. For each current or former position the candidate has held, there is a unique CandidateWorkHistory instance associated with that candidate. Each CandidateWorkHistory instance corresponds to a line item on the **Work History** tab of a candidate record in the Bullhorn application.

You must create the CandidateWorkHistory instance after a Candidate entity instance is created and saved to the Bullhorn database. The user ID of the candidate associates the CandidateWorkHistory DTO with a candidate instance.

The CandidateWorkHistory DTO has several properties that you can set. Please refer to the **reference documentation** to see the complete list.

The code snippet below creates an entry for one particular job held by our candidate by supplying the title, startDate, endDate, and comment properties in the CandidateWorkHistory DTO.

Java code

```
CandidateWorkHistoryDto savedCandidateWorkHistory =  new CandidateWorkHistoryDto();
savedCandidateWorkHistory.setCandidateID(savedCandidate.getUserID());
savedCandidateWorkHistory.setTitle("Architect");
/* The startDate and endDate variables are XMLGregorianCalendar  values which
need to be defined before being used. */
savedCandidateWorkHistory.setStartDate(startDate);
savedCandidateWorkHistory.setEndDate(endDate);
savedCandidateWorkHistory.setComments("Worked  in Avaya till 2009.");
service.save(currentSession,  savedCandidateWorkHistory);
```

C# code

```
candidateWorkHistoryDto savedCandidateWorkHistory =  new candidateWorkHistoryDto();
savedCandidateWorkHistory.candidateID =  savedCandidate.userID;
savedCandidateWorkHistory.candidateIDSpecified =  true;
  savedCandidateWorkHistory.title =  "Architect";
/* The startDate and endDate variables are  DateTime values which need to be
defined before being used. */
savedCandidateWorkHistory.startDate = startDate;
savedCandidateWorkHistory.startDateSpecified =  true;
savedCandidateWorkHistory.endDate = endDate;
savedCandidateWorkHistory.endDateSpecified = true;
savedCandidateWorkHistory.comments = "Worked  in Avaya till 2009.";
```

```
service.save(currentSession, savedCandidateWorkHistory);
```

PHP code

```
$workHistory_array = array(
  'candidateID'  => $savedCandidate->userID,
  'title'  => 'Architect',
  'comments' => 'Worked at Avaya until 2009.',
  'startDate' => new SoapVar('2008-01-01', XSD_DATE, "date",
    "http://www.w3.org/2001/XMLSchema"),
  'endDate' => new SoapVar('2009-12-31', XSD_DATE, "date",
    "http://www.w3.org/2001/XMLSchema"),
  'companyName' => 'Avaya'
  );
$SOAP_workHistory = new  SoapVar($workHistory_array, SOAP_ENC_OBJECT,
   "candidateWorkHistoryDto",
   "http://candidate.entity.bullhorn.com/");

$request_array = array(
  'session'  => $API_currentSession,
  'dto'  => $SOAP_workHistory
  );

$SOAP_request = new SoapVar($request_array,  SOAP_ENC_OBJECT, "save",
   "http://save.apiservice.bullhorn.com/");
try{
  $saveResponse  = $BHClient->save($SOAP_request);
   } catch (SoapFault $fault) {
       var_dump($BHClient->_getLastRequest());
       die($fault->faultstring);
  }

$API_currentSession = $saveResponse->return->session;
$savedCandidateWorkHistory = $saveResponse->return->dto;
```

Adding a CandidateEducation entity

The CandidateEducation entity represents an educational degree or course of study that a candidate lists on their resume. It can also be used for certifications, such as nursing licenses. Each CandidateEducation instance corresponds to a line item on the **Education** tab of a candidate record in the Bullhorn application.
You must create the CandidateEducation after a Candidate instance is created and saved to the Bullhorn database. The user ID of the candidate associates the CandidateEducation DTO with a candidate instance.

The CandidateEducation DTO has several properties that you can set. For the complete list, see the reference documentation.

The following code snippet sets the state and the comment properties in the CandidateEducation DTO.

Java code

```
CandidateEducationDto savedCandidateEducation = new  CandidateEducationDto();
savedCandidateEducation.setCandidateID(savedCandidate.getUserID());
savedCandidateEducation.setState("CO");
savedCandidateEducation.setComments("MS in  Computer Science");
/* The startDate and endDate variables are XMLGregorianCalendar values which need
to be defined before being used. */
savedCandidateEducation.setStartDate(startDate);
savedCandidateEducation.setEndDate(endDate);
service.save(currentSession,  savedCandidateEducation);
```

C# code

```
candidateEducationDto savedCandidateEducation = new candidateEducationDto();
  savedCandidateEducation.candidateID =  savedCandidate.userID;
  savedCandidateEducation.candidateIDSpecified = true;
  savedCandidateEducation.state = "CO";
  savedCandidateEducation.comments = "MS in  Computer Science";
/* The startDate and endDate variables are  DateTime values which need
to be defined before being used. */
savedCandidateEducation.startDate = startDate;
savedCandidateEducation.startDateSpecified = true;
savedCandidateEducation.endDate = endDate;
savedCandidateEducation.endDateSpecified = true;

service.save(currentSession, savedCandidateEducation);
```

PHP code

```
$education_array = array(
  'candidateID'  => $savedCandidate->userID,
  'state'  => 'CO',
  'school'  => 'Colorado University',
  'degree'  => 'MS',
  'major'  => 'Computer Science',
  'graduationDate' => new SoapVar('2004-06-12', XSD_DATE, "date",
      "http://www.w3.org/2001/XMLSchema")
  );
$SOAP_education = new SoapVar($education_array, SOAP_ENC_OBJECT, "candidateEducationDto",
       "http://candidate.entity.bullhorn.com/");

$request_array = array(
  'session'  => $API_currentSession,
  'dto'  => $SOAP_education
  );

$SOAP_request = new SoapVar($request_array,  SOAP_ENC_OBJECT, "save",
   "http://save.apiservice.bullhorn.com/");
try{
    $saveResponse  = $BHClient->save($SOAP_request);
   } catch (SoapFault $fault) {
     var_dump($BHClient->_getLastRequest());
     die($fault->faultstring);
  }

$API_currentSession =  $saveResponse->return->session;
$savedCandidateEducation = $saveResponse->return->dto;
```

Associating categories

The main Candidate DTO has a categoryID property that you use to assign a primary category to a candidate. Categories have a many-to-many relationship with candidates. Use the associate operation to establish that relationship.

The associate operation creates an association between a primary entity instance and a secondary entity instance. A candidate is a primary entity instance and a category is a secondary entity instance.

The following code snippets show you how to retrieve a list of categories for a corporation and associate them with a Candidate DTO.

Java code

```
DtoQuery categoryQuery = new DtoQuery();
categoryQuery.setEntityName("Category");
categoryQuery.setWhere("enabled = 1");
ApiQueryResult categoryResult =  service.query(currentSession,  categoryQuery);
currentSession = categoryResult.getSession();
if (!categoryResult.getIds().isEmpty())
  {
   for (int  i =0; i< categoryResult.getIds().size() ; i++)
   {
     ApiFindResult category =  service.find(currentSession,
       "Category", categoryResult.getIds().get(i));
     currentSession  = category.getSession();
     CategoryDto  thisCategory = (CategoryDto)category.getDto();
     System.out.println("Found  category using find() method: "
      +thisCategory.getCategoryID() + ", "
      +thisCategory.getOccupation());
    }
  }
/*  Associate the new Candidate DTO with two categories, got category ID
from the  query above */
service.associate(currentSession, "Candidate", thisCandidate.getUserID(),
 "categories" , {value from list above});
service.associate(currentSession, "Candidate", thisCandidate.getUserID(),
 "categories" , {value from list above});
```

C# code

```
dtoQuery categoryQuery = new dtoQuery();
categoryQuery.entityName = "Category";
categoryQuery.maxResults = 10;
categoryQuery.where = "enabled = 1";
apiQueryResult categoryResult =  service.query(currentSession, categoryQuery);
currentSession = categoryResult.session;

foreach (int i in categoryResult.ids)
  {
    apiFindResult category = service.find(currentSession,  "Category", i);
    currentSession = category.session;
   // Run the  debugger to find the category ID for this category
   categoryDto thisCategory = (categoryDto)category.dto;
   }

service.associate(currentSession, "Candidate", savedCandidate.userID,
  "categories", {value  from list above});
service.associate(currentSession, "Candidate", savedCandidate.userID,
  "categories", {value  from list above});
```

PHP code

```
$query_array = array(
  'entityName'  => 'Category',
  'maxResults'  => 10,
  'where'  => 'enabled = 1',
  'parameters'  => array()
  );
$SOAP_query = new SoapVar($query_array,  SOAP_ENC_OBJECT, "dtoQuery",
 "http://query.apiservice.bullhorn.com/");
$request_array = array (
  'session'  => $API_currentSession,
```

```
  'query'  => $SOAP_query
  );

$SOAP_request = new SoapVar($request_array,  SOAP_ENC_OBJECT, "query",
  "http://query.apiservice.bullhorn.com/");
try{
  $queryResult  = $BHClient->query($SOAP_request);
  } catch  (SoapFault $fault) {
    echo  $BHclient->__getLastResult();
    die($fault->faultstring);
  }

$API_currentSession =  $queryResult->return->session;

foreach ($queryResult->return->ids as  $value){
  $findId  = new SoapVar($value, XSD_INTEGER, "int","http://www.w3.org/2001/XMLSchema");
  $find_request  = array (
  'session'  => $API_currentSession,
  'entityName'  => 'Category',
  'id'  => $findId
  );

  try{
     $findResult  = $BHClient->find($find_request);
     } catch (SoapFault $fault) {
        echo  $BHclient->__getLastResult();
        die($fault->faultstring);
     }

   $API_currentSession  = $findResult->return->session;
   $foundCategory = $findResult->return->dto;

   echo('Found category using find(): '.$foundCategory->categoryID.' - '.
     $foundCategory->occupation.'<br>');
}

$category_array = array(
  'session'  => $API_currentSession,
  'entityName'  => 'Candidate',
  'id'  => new SoapVar($savedCandidate->userID, XSD_INT, "int",
     "http://www.w3.org/2001/XMLSchema"),
  'associationName'  => 'categories',
  'associateId'  => new SoapVar(254969, XSD_INT, "int",
     "http://www.w3.org/2001/XMLSchema")
  );

$SOAP_request_associate = new  SoapVar($category_array, SOAP_ENC_OBJECT, "associate",
  "http://associate.apiservice.bullhorn.com/");
try{
   $associateResponse  = $BHClient->associate($SOAP_request_associate);
   } catch  (SoapFault $fault) {
     echo  $BHclient->__getLastResult();
     die($fault->faultstring);
   }
$API_currentSession =  $associateResponse->return->session;
```

Associating skills

The Skill entity represents a candidate's skill. Use the Skill entity in a candidate record to indicate that the candidate has that skill, or in a JobOrder record to indicate that skill is a requirement for applicants for that job.

Skills are similar to categories in that they have a many:many relationship to the candidate. The only difference is rather than just one association, skills have two associations: primarySkills and secondarySkills.

The following code snippet shows you how to retrieve a list of skills defined for a corporation and to associate a primary skill with a Candidate DTO.

Java code

```
// Get a list of skills for this corp
DtoQuery skillQuery = new DtoQuery();
skillQuery.setEntityName("Skill");
skillQuery.setMaxResults(10);
skillQuery.setWhere("enabled = 1");
ApiQueryResult skillResult =  service.query(currentSession, skillQuery);
currentSession = skillResult.getSession();
  if (!skillResult.getIds().isEmpty())
  {
   for (int  i1 =0; i1< skillResult.getIds().size() ; i1++)
    {
     ApiFindResult primarySkills =  service.find(currentSession, "Skill",
       skillResult.getIds().get(i1));
     currentSession = primarySkills.getSession();
     SkillDto thisSkillSet =  (SkillDto)primarySkills.getDto();
     System.out.println("Found  primary skills using find() method:" +
       thisSkillSet.getSkillID() + "," + thisSkillSet.getName());
    }
  }
// Add  skills to the candidate
service.associate(currentSession,  "Candidate",
thisCandidate.getUserID(),  "primarySkills" , {value from list above});
```

C# code

```
dtoQuery skillQuery = new dtoQuery();
skillQuery.entityName = "Skill";
skillQuery.maxResults = 10;
skillQuery.where = "enabled = 1";
apiQueryResult skillResult =  service.query(currentSession, skillQuery);
currentSession = skillResult.session;

foreach (int i in skillResult.ids)
  {
    apiFindResult primarySkills = service.find(currentSession,  "Skill", i);
    currentSession = primarySkills.session;
    // Run the  debugger to find the category id for this category
    skillDto thisSkillSet =  (skillDto)primarySkills.dto;
  }

service.associate(currentSession,  "Candidate", savedCandidate.userID,
    "primarySkills" , {value  from list above});
```

PHP code

```
$skills_array = array(
  'session'  => $API_currentSession,
  'entityName'  => 'Candidate',
  'id'  => new SoapVar($savedCandidate->userID, XSD_INT, "int",
     "http://www.w3.org/2001/XMLSchema"),
  'associationName'  => 'primarySkills',
```

```
  'associateId'  => new SoapVar(121052, XSD_INT, "int",
"http://www.w3.org/2001/XMLSchema")
  );
$SOAP_request_associate = new  SoapVar($skills_array, SOAP_ENC_OBJECT, "associate",
    "http://associate.apiservice.bullhorn.com/");
try{
    $associateResponse  = $BHClient->associate($SOAP_request_associate);
   } catch  (SoapFault $fault) {
     echo  $BHclient->__getLastResult();
     die($fault->faultstring);
  }
$API_currentSession =  $associateResponse->return->session;
```

Updating candidates

To update a candidate in the Bullhorn system, you must retrieve an instance of the specific Candidate DTO class and change the properties for that candidate.

Important: When you save an instance of a DTO, you must include all values that should remain in the database, not just the values being updated. When a DTO is saved, the API service compares the values in the DTO to those currently in the database and automatically determines which fields need to be updated. If you provide only the values you want updated, the API service will assume that you want to make all other properties null, which will result either in an error (for non-nullable fields) or in unwanted changes to your data.

The following code snippet demonstrates how to retrieve a candidate using the find operation. You then change its phone number by modifying the phone property. You also need to use the save operation to save the Candidate DTO back in the Bullhorn system to make the update permanent.

Java code

```
DtoQuery candidateUpdateQuery = new DtoQuery();
candidateUpdateQuery.setEntityName("Candidate");
candidateUpdateQuery.setMaxResults(10);
candidateUpdateQuery.setWhere("FirstName =  'Brad'");
ApiQueryResult queryUpdateResult =  service.query(currentSession, candidateUpdateQuery);
currentSession = queryUpdateResult.getSession();
if (!queryUpdateResult.getIds().isEmpty())
  {
    for (int  i =0; i< queryUpdateResult.getIds().size() ; i++)
  {
    ApiFindResult candidate = service.find(currentSession,  "Candidate",
      queryUpdateResult.getIds().get(i));
   currentSession = candidate.getSession();
   CandidateDto thisCandidate = (CandidateDto)candidate.getDto();
    System.out.println("Found  candidate using find() method: " +thisCandidate.getName()
     + " of  phone number " +thisCandidate.getPhone());

    thisCandidate.setPhone("4081111111");

    ApiSaveResult resultSavePhone = service.save(currentSession,  thisCandidate);
    currentSession = resultSavePhone.getSession();
  }
  }
```

C# code

```
dtoQuery candidateUpdateQuery = new dtoQuery();
candidateUpdateQuery.entityName =  "Candidate";
candidateUpdateQuery.maxResults = 10;
candidateUpdateQuery.where = "FirstName =  'Brad'";
apiQueryResult queryUpdateResult =  service.query(currentSession, candidateUpdateQuery);
currentSession = queryUpdateResult.session;
```

```
foreach (int i in queryUpdateResult.ids)
  {
    apiFindResult candidate = service.find(currentSession,  "Candidate", i);
    currentSession = candidate.session;
    candidateDto thisCandidate = (candidateDto)candidate.dto;

    thisCandidate.phone = "4081111111";
    apiSaveResult resultSavePhone = service.save(currentSession,  thisCandidate);
    currentSession = resultSavePhone.session;
  }
```

PHP code

```
$query_array = array(
  'entityName'  => 'Candidate',
  'maxResults'  => 10,
  'where'  => "name = 'Brett Jennings'",
  'parameters'  => array()
  );
$SOAP_query = new SoapVar($query_array,  SOAP_ENC_OBJECT, "dtoQuery",
   "http://query.apiservice.bullhorn.com/");
$request_array = array (
  'session'  => $API_currentSession,
  'query'  => $SOAP_query
  );
$SOAP_request = new SoapVar($request_array,  SOAP_ENC_OBJECT, "query",
    "http://query.apiservice.bullhorn.com/");
try{
    $queryResult  = $BHClient->query($SOAP_request);
    $API_currentSession  = $queryResult->return->session;
   } catch  (SoapFault $fault) {
      echo  $BHclient->__getLastResult();
      die($fault->faultstring);
  }

foreach ($queryResult->return->ids as  $value){
  $findId  = new SoapVar($value, XSD_INTEGER, "int",
    "http://www.w3.org/2001/XMLSchema");
  $find_request  = array (
  'session'  => $API_currentSession,
  'entityName'  => 'Candidate',
  'id'  => $findId
  );

try{
    $findResult  = $BHClient->find($find_request);
   }catch  (SoapFault $fault) {
     echo  $BHclient->__getLastResult();
     die($fault->faultstring);
  }

$API_currentSession  = $findResult->return->session;

//change their phone number
$thisCandidate  = $findResult->return->dto;
$thisCandidate->phone  = '408-111-1111';

//save them back
```

```
$SOAP_thisCandidate  = new SoapVar($thisCandidate, SOAP_ENC_OBJECT, "candidateDto",
    "http://candidate.entity.bullhorn.com/");
$request_array  = array(
  'session' => $API_currentSession,
  'dto' => $SOAP_thisCandidate
  );

$SOAP_request  = new SoapVar($request_array, SOAP_ENC_OBJECT, "save",
   "http://save.apiservice.bullhorn.com/");

try{
     $saveResult  = $BHClient->save($SOAP_request);
   }catch  (SoapFault $fault) {
     echo  $BHclient->__getLastResult();
     die($fault->faultstring);
  }
$API_currentSession  = $saveResult->return->session;
}
```

Deleting candidates

For most entities, records are not deleted from the Bullhorn system. This ensures that the system always retains a record of all information that was handled by the organization and that referential integrity is maintained. However, specific candidate records can be hidden from normal view by setting their isDeleted property to true.
To delete a candidate in the Bullhorn system, you must set the isDeleted property to true on the specific instance of the Candidate DTO class.

Java code

```
/* Use the find  operation to look for a candidate record you want to delete.
Then set the  IsDeleted flag on that record to mark it for deletion. */
DtoQuery candidateDeleteQuery = new DtoQuery();
candidateDeleteQuery.setEntityName("Candidate");
candidateDeleteQuery.setMaxResults(10);
candidateDeleteQuery.setWhere("FirstName =  'Brad'");
ApiQueryResult queryDeleteResult =  service.query(currentSession, candidateDeleteQuery);
currentSession = queryDeleteResult.getSession();
if (!queryDeleteResult.getIds().isEmpty())
  {
   for (int i  =0; i< queryDeleteResult.getIds().size() ; i++)
  {
   ApiFindResult candidate =  service.find(currentSession,
   "Candidate",
   queryDeleteResult.getIds().get(i));
   currentSession = candidate.getSession();
   CandidateDto thisCandidate =
   (CandidateDto)candidate.getDto();
   thisCandidate.setIsDeleted(true);
/* To actually  delete the candidate, you would pass thisCandidate to the save() operation.
This step has been omitted in this case. */
```

C# code

```
/* Use the find operation to look  for a candidate record you want to delete.
  Then set the IsDeleted flag on that  record to mark it for deletion. */
dtoQuery candidateQuery = new dtoQuery();
candidateQuery.entityName = "Candidate";
candidateQuery.where = "FirstName =  'Brett'";
apiQueryResult queryResult =  service.query(currentSession, candidateQuery);
currentSession = queryResult.session;
```

```
foreach (int i in queryResult.ids)
  {
    apiFindResult candidate = service.find(currentSession,  "Candidate", i);
    currentSession = candidate.session;
    candidateDto thisCandidate = (candidateDto)candidate.dto;
    thisCandidate.isDeleted = true;
/* To actually delete the  candidate, you would pass thisCandidate to the save() operation.
This step has  been omitted in this case. */
  }
```

PHP code

```
/* Use the find operation to look for a candidate record you want
to delete. Then set the IsDeleted flag on that  record to mark it for deletion. */
candidateDeleteQuery.setMaxResults(10);
$query_array  = array(
  'entityName' => 'Candidate',
  'maxResults' => 10,
  'where' => "name = 'Brett  Jennings'",
  'parameters' => array()
  );

$SOAP_query  = new SoapVar($query_array, SOAP_ENC_OBJECT, "dtoQuery",
   "http://query.apiservice.bullhorn.com/");
$request_array  = array (
  'session' => $API_currentSession,
  'query' => $SOAP_query
  );

$SOAP_request  = new SoapVar($request_array, SOAP_ENC_OBJECT, "query",
   "http://query.apiservice.bullhorn.com/");
try{
    $queryResult =  $BHClient->query($SOAP_request);
    $API_currentSession =  $queryResult->return->session;
  } catch (SoapFault $fault) {
    echo $BHclient->__getLastResult();
    die($fault->faultstring);
  }
$API_currentSession  = $queryResult->return->session;
foreach  ($queryResult->return->ids as $value){
  $findId = new SoapVar($value, XSD_INTEGER,  "int",
     "http://www.w3.org/2001/XMLSchema");
  $find_request = array (
    'session' => $API_currentSession,
    'entityName' => 'Candidate',
    'id' => $findId
    );
  try{
    $findResult =  $BHClient->find($find_request);
     } catch (SoapFault $fault) {
       echo  $BHclient->__getLastResult();
       die($fault->faultstring);
     }
   $API_currentSession =  $findResult->return->session;

    //Set  the isDeleted property to true
    $thisCandidate =  $findResult->return->dto;
    $thisCandidate->isDeleted  = 'true';
```

```
    /* To actually delete the candidate, you would pass thisCandidate to the save()
     operation. This step has been omitted in this case. */
  }
```