

#BullhornEngage

ENGAGE

Automating Workflows Through Bullhorn and Partners

Haris Iqbal (hiqbal@bullhorn.com)

Kevin Jones (kevin@herefish.com)

Outline

- Intro to automation
- Deep Dive into Bullhorn Automation Framework
- Examples using Herefish

The problems that
automation can solve

- Current processes are **manual and prone to error**
- Communication with candidates, clients & contractors **gets forgotten**
- Automation **saves time**, as repetitive tasks are no longer necessary
- Automation can offer a **consistent experience** across the organization
- Use Automation to **enforce** custom rules and **standardize** validations

Bullhorn Automation Framework

ENGAGE

Bullhorn Automation Framework

The Bullhorn Automation Framework provides Professional Services, Systems Integrators, and other 3rd party developers with the ability to enable custom asynchronous functionality for clients without the need to deploy custom code.



Components of an Automation

The 2 components for these automations are:

Conditions

A condition is a rule that defines what qualifies a certain event in Bullhorn to trigger a response.

Outcomes

An outcome is a response to a condition. These let you define what should happen if your predefined conditions are met.



Conditions

- These define what rules need to be met in order to trigger the execution of an outcome
- An automation can consist of zero or more conditions
- Conditions can be compounded using AND or OR operators
- We are planning on supporting the following events as triggers for automations:
 - Adding a record
 - Deleting a record
 - Updating a record
- Entities we will support are: Candidate, Contact, ClientCorporation, JobOrder, JobSubmission, Lead, Opportunity, Placement, Sendout

Match Operators

Match Operators help you execute your conditions against fields. The feature will support the following match operators:

- Contains
- Does Not Contain
- Equals
- Does Not Equal
- Begins With
- Ends With
- Is Empty
- Is Not Empty
- Is Greater Than
- Is Less Than
- Is Updated

Outcomes

3 types of outcomes we are working on:

1. Update Entity

Lets you update data on an associated entity in Bullhorn

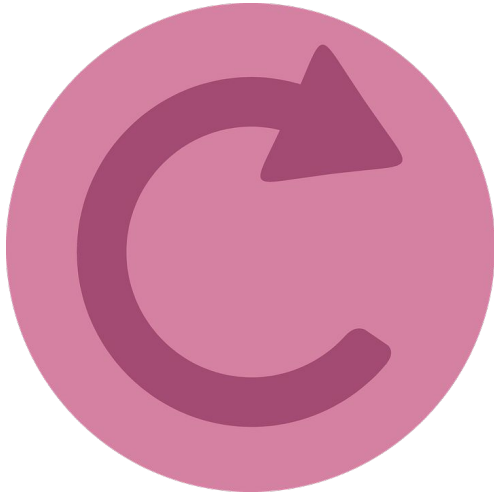
2. Webhook

Lets you make a REST call to an API owned by you or a 3rd party

3. Serverless Function New!

Lets you define custom serverless functions using Bullhorn-provided SDKs

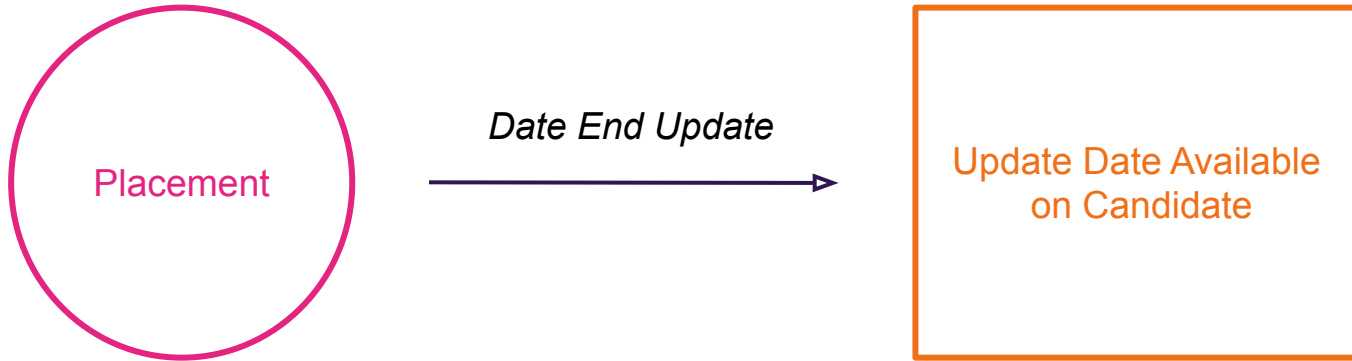
Update Entity Outcomes



- Let you easily update data on the entity itself, or other associated entities, e.g.
 - Update status on the Job Order itself
 - Update status on the Contact associated with the Job Order
- Data can be updated to a fixed value, e.g.
 - Update Job Status to 'Open'
- Data can be updated to a value obtained from an associated entity, e.g.
 - Update Contact customDate1 to what customDate1 is on the Job

Update Entity: A Use Case

Update Entity Outcomes will let you easily update data on Bullhorn Entities without the need to write any custom code. One case we'll look at today is to create an automation that does the following:



Update Entity: Setup

PUT

/rest-services/<corp-token>/services/automation/create

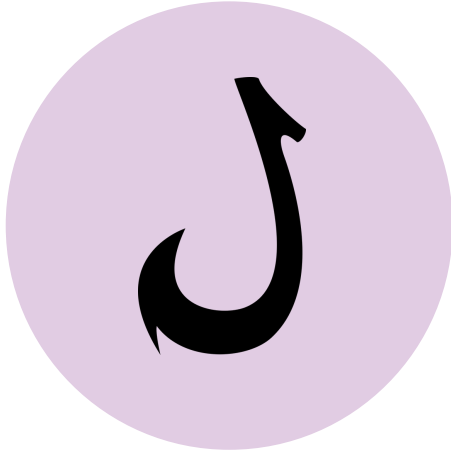
BODY

```
{
  "name": "Placement Date End Auto" ,
  "type": "POSTSAVE" ,
  "isEnabled": true,
  "onUpdate": true,
  "entity": "Placement",
  "conditionData": {
    "conditions": [
      {
        "entity": "Placement",
        "field": "dateEnd",
        "matchOperator": "ISUPDATED"
      }
    ],
    "logicOperator": "AND"
  },
  "outcomes": [
    {
      "outcomeType": "UPDATEENTITY" ,
      "entity": "Candidate",
      "data": {
        "dateAvailable": "$dateEnd"
      }
    }
  ]
}
```

Demo Time!

ENGAGE

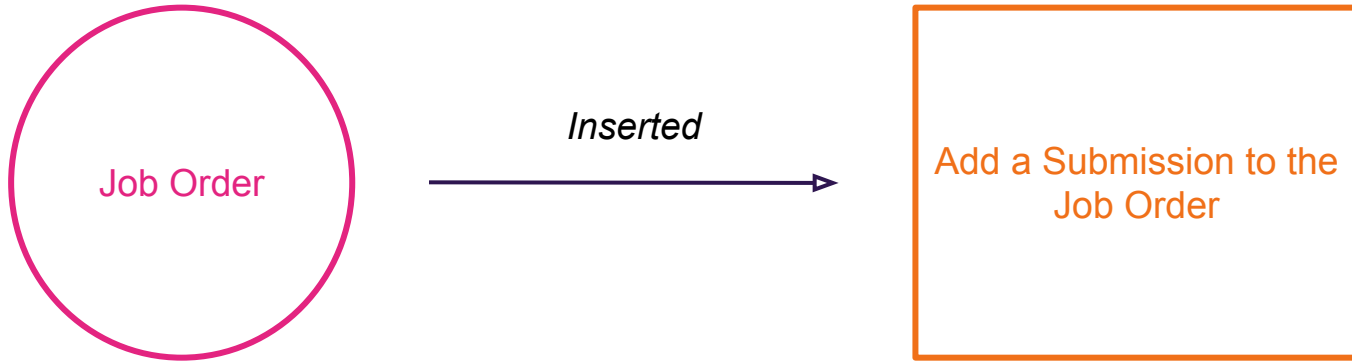
Webhook Outcomes



- Let you make a REST call to an endpoint owned by you or a 3rd Party
- Lets you pass a BH REST Token and a REST end point as parameters to the REST call
- Lets you pass data from your trigger event to the API endpoint, e.g.
 - On a placement update, you can pass through the candidate's id to this API endpoint
- Lets you pass other static parameters to the API end point as well - think API Keys

Webhook Outcome: A Use Case

Webhook Outcomes will let you easily invoke REST calls to APIs owned by your or 3rd parties. One case we'll look at today is to create an automation that does the following:



Webhook: Setup

PUT

`/rest-services/<corp-token>/services/automation/create`

BODY

```
{
  "name": "Job Webhook Auto",
  "type": "POSTSAVE",
  "isEnabled": true,
  "onUpdate": false,
  "onCreate": true,
  "entity": "JobOrder",
  "conditionData": {
    "conditions": [],
    "logicOperator": "AND"
  },
  "outcomes": [
    {
      "outcomeType": "WEBHOOK",
      "url": "http://kevin-backend:3000/create-submission",
      "httpMethod": "POST",
      "queryParams": {
        "jobId": "$id"
      }
    }
  ]
}
```

Demo Time!

ENGAGE

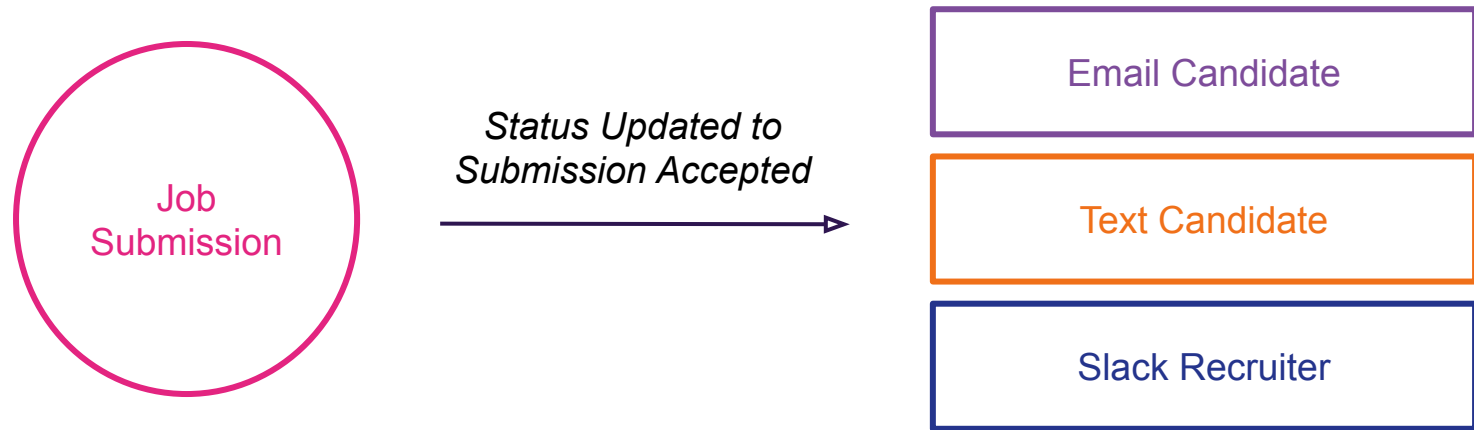
Serverless Functions



- Don't need to manage any servers
 - Just write your own code
 - No need to manage any hosting or worry about deployments
- Fast/easy to set up
 - Use Bullhorn's REST API to create these functions
 - Use provided SDKs with easy to use methods for common functionality
- Automatically scale
 - The function scales up or down based on usage

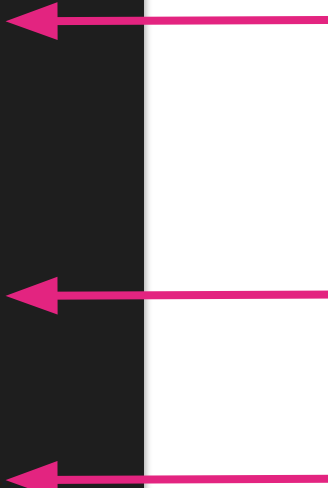
Serverless Functions: A Use Case

Serverless Function outcomes can be especially useful for performing common 3rd party actions. One case we'll look at today is to create an automation that does the following:



Serverless Function: A Use Case

```
async function performCustomLogic(request, responseBody) {  
  let mail = await emailService.sendEmail(  
    emailUser,  
    emailPassword,  
    request.email,  
    request.message,  
    '<p>Hey ' + request.candidateName + '! Your submission for Job: ' + request.jobTitle +  
    ' got updated to status: ' + request.status + '</p>'  
  );  
  
  let message = await messageService.sendMessage(  
    twilioAccountSid,  
    twilioAuthToken,  
    request.phone,  
    'Hey ' + request.candidateName + '! Your submission for Job: ' + request.jobTitle +  
    ' got updated to status: ' + request.status + '.'  
  );  
  
  let slack = await slackService.postSlackMessage(  
    slackToken,  
    request.channel,  
    'Hey! The Submission for candidate #' + request.candidateId + ' - ' + request.candidateName  
    + ' and job #' + request.jobId + ' - ' + request.jobTitle + ' got updated to: ' + request.status  
  );  
  
  responseBody = { ...responseBody, mail, message, slack };  
}
```



Serverless Function: Setup

PUT */rest-services/<corp-token>/services/function/create*

BODY

```
{
  "functionName": "Submission_Test_2" ,
  "customFunction": "URL-Encoded Function Body here"
}
```

RESPONSE

```
{
  "entityCorporationId" : 4063,
  "functionId" : 10,
  "functionName" : "Submission_Test_2" ,
  "functionArn" :
    "arn:aws:lambda:us-east-1:120804024563:function:Submission_Test_2"
  ,
  "resourceId" : "jd0j7n" ,
  "functionUrl" :
    "https://ldjxaug822.execute-api.us-east-1.amazonaws.com/Engage_Test/Submission_Test_2" ,
  "entityId" : 10
}
```

Serverless Function: Setup

PUT

/rest-services/<corp-token>/services/automation/create

BODY

```
{
  "name": "Engage Submission Status Auto",
  "type": "POSTSAVE",
  "isEnabled": true,
  "onUpdate": true,
  "entity": "JobSubmission",
  "conditionData": {
    "conditions": [
      {
        "entity": "JobSubmission",
        "field": "status",
        "matchOperator": "EQUALS",
        "value": "Submission Accepted"
      }
    ],
    "logicOperator": "AND"
  },
  "outcomes": [
    {
      "outcomeType": "FUNCTION",
      "functionId": 10,
      "queryParams": {
        "email": "$candidate.email",
        "phone": "$candidate.phone",
        "candidateName": "$candidate.name",
        "candidateId": "$candidate.id",
        "channel": "$sendingUser.pager",
        "jobTitle": "$jobOrder.title",
        "jobId": "$jobOrder.id",
        "status": "$status"
      }
    }
  ]
}
```

Demo Time!

ENGAGE

More Use Cases

- Automate tasks in your Submission/Placement workflow to:
 - Send out notifications
 - Add task reminders
 - Create Notes & more
- Use serverless functions to push data to or integrate with other systems/APIs e.g. ADP
- Job gets assigned to recruiter and they get a slack message/Email notification



Possible Future Enhancements

- Make REST API session tokens available to serverless functions
- Make it possible for marketplace partners to provide their SDKs to be used in serverless functions
- Scheduled Outcome executions
 - Send Notifications to Candidates that have placements ending in the next 24 hours



Automation with Herefish

Powered by
Bullhorn Integration

ENGAGE

Say hi to **Jane**



Talent Pool

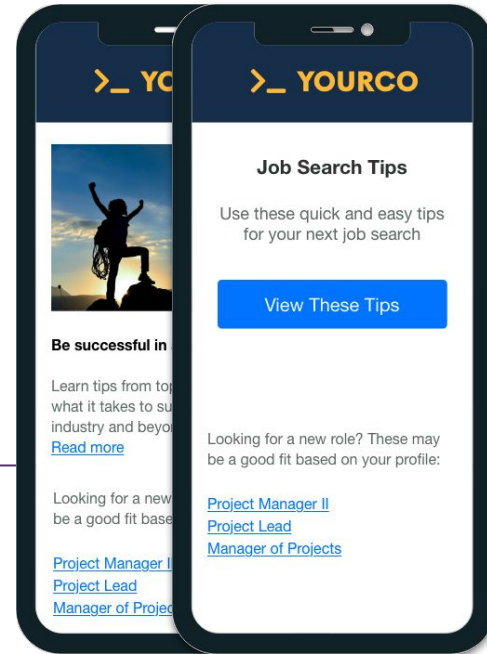
Consistently communicate with contacts

Personalize jobs based on their profile

Drive from passive to **active**



Talent Pool



Apply

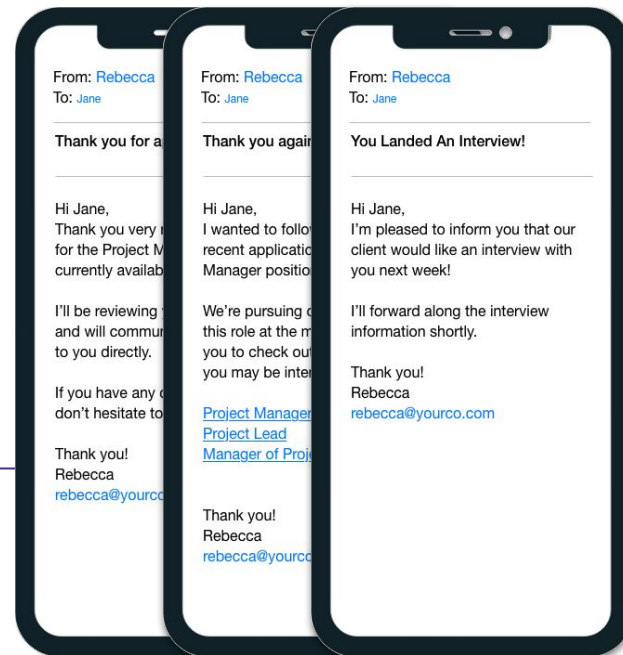
Direct response from recruiter

Notify Candidate of key changes

Update status in the ATS



Apply

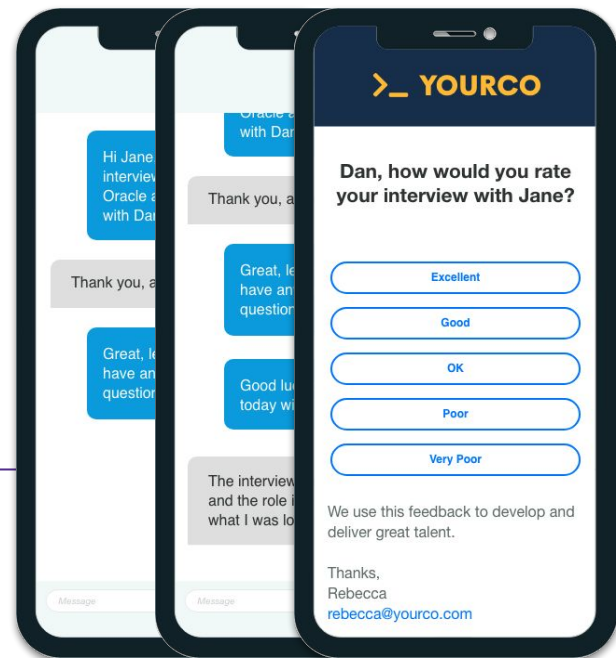
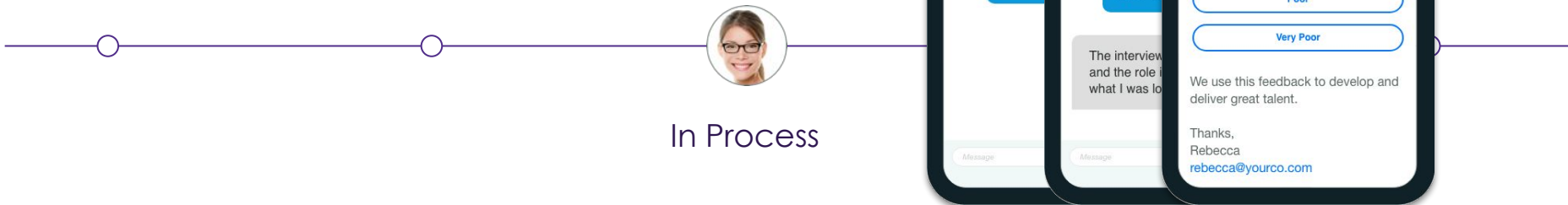


In Process

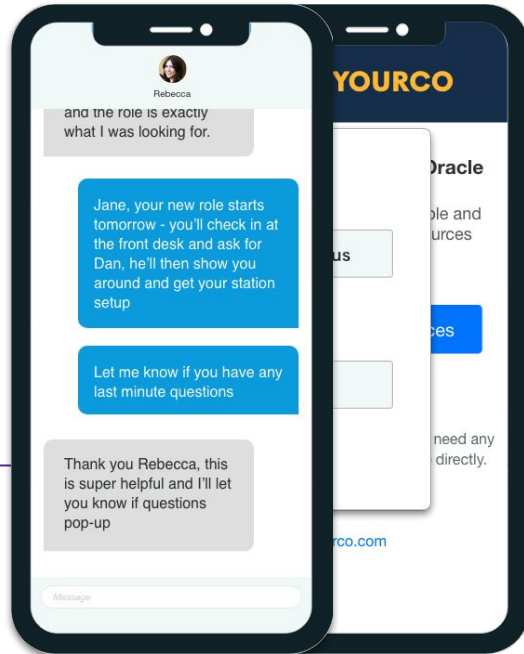
Communicate around key dates

Keep candidates **in-the-loop**

Capture hiring manager feedback



Hire



Share key information

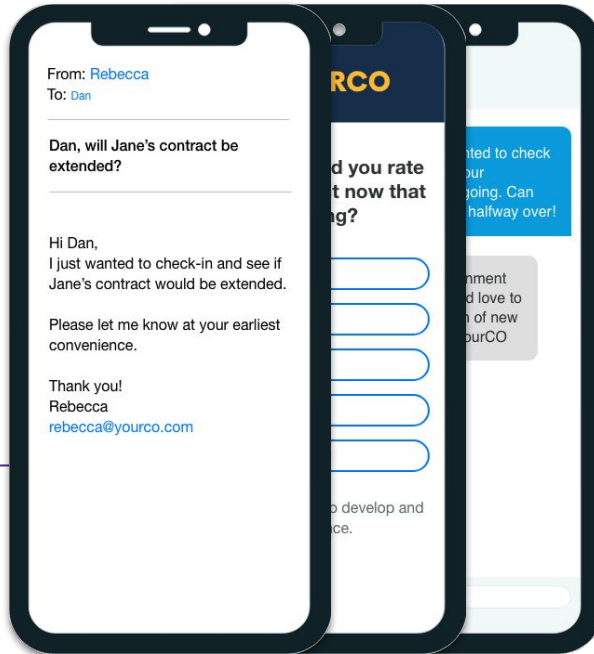
Update **ATS status**

Enable **1:1 communication**



Hire

Contractor Care



Engage throughout the assignment

Gather feedback during assignment

Communicate with company contacts



Contractor Care



**RECRUITING
BUSY WORK**

Demo Time!

ENGAGE

Questions?