**Nitro Platform Questions**

1. Describe how an action is logged in the Nitro platform. Include any relevant methods, along with the sequence of execution of the method(s), and a description of the relevant parameters.

A Nitro *Action* is a trackable user activity. This could be **consuming a content, creating content, or sharing content.** In order to log an *Action,* you must:

* Create An Action in Nitro Admin Console
* Log the Action with user.logAction() API

For example, in order to track users who click on the ‘Like’ button.

* First, under the Nitro Admin Console, create the ‘Like’ Action.
* Second, on the client-side, include the Nitro Javascript Library (Nitro.js).   
  The primary function used is ***user.logAction(sessionId, tags,[value], …).*** Where tags is the name of your action.

Execution of ***user.logAction()*** is handled by Nitro.js, a Javascript library that asynchronously call the REST API. Once, a Nitro object is properly defined, the ***logAction*** method is called with some required and optional parameters.

* ***SessionId –*** String - Response from user.Login, required if authentication is enabled.
* ***Tags*** – Comma-separated value list [], useful metadata about an *Action* which can be later used by Nitro for generating reports, games, leaderboards, etc.
* ***StoreResponse –*** Boolean – if enabled, client can look up response to action from *user.getResponses*
* ***Value*** – Integer – Send a value associated with this *Action*. Can be used to tally leaderboards, assign rewards and reporting to client engagement department.

A sample client-side code would look like this:

…(define nitro object)

nitro.callAPI(“method=user.logAction&sessionId=f02aa5c5dfb844b4&tags=LIKE,STAR&value=13”, callBackFunction(data, token));

callBackFunction will be called when the method returns and the JSON *data* is returned. The *token* is an optional argument provided at the request.

In conclusion, Nitro allows *Actions* to be logged and is flexible enough to be site-specific and is a key component in Bunchball’s Gamification strategy.

2. A prospect asks the following question:

We have four systems that we want to "talk" to Nitro. Each system uses the same unique identifier for our employees, and stores different behaviors about them (# hours worked, page visits on our Intranet, comments, article submissions, etc.).

I want to know the following:

a) How does registration work in Nitro? How do we sync existing users?

b) How do these four systems talk to Nitro? I want to incent users every time they visit a page, and whenever they comment on an article.

c) We also have an iOS app. Is Nitro compatible? If so, are there are restrictions I should be aware of?

Users are created on-the-fly in Nitro, authentication is expected to be handled by client’s existing system. All 4 of these systems can use the same unique identifier to log user *Actions.*

1. To register and sync existing users in Nitro, choose the same unique identifier for your employees as the *userId* parameter to Nitro’s **user.login** *API.* First Nitro searches for this unique alphanumeric string and if it doesn’t already exist, Nitro will create it. From then on, all the gamification elements of Nitro can be used on this existing user.
2. These four systems can talk to Nitro with the Nitro API, which supports both XML and JSON, RESTful API, and it is platform independent, can be accessed with Java, C#, PHP, Ruby, etc.   
     
   Bunchball understands what promotes user engagement, a few of them are ***Competition, Status, Achievements, Self-Expression.*** Nitro utilizes this online psychology and translate them into trackable results. Competition can be implemented as a Leaderboard; Status can be tallied as Points; Achievements can translate to levels and trophies or badges; Self-Expression is

allowed when users comment or share a content.

* *Actions* can be tracked that when a page loads, it logs the user who visits the page.
* *Actions* can also be setup to track comments. It can trigger *rewards* or lead to a success of a *challenge or* earn *trophies / badges.* 
  + All these *Actions, Rewards, Notifications, Trophies* and *Challenges can be administered* within the ***Nitro Admin Console*.**

1. iOS apps are compatible with Nitro. One restriction that comes to mind is the Flash capability of iOS devices. Nitro utilizes *Flash Connector* in order to trigger Nitro events. You might want to consider making a iOS friendly-version of these Flash components.

3. A prospect asks the following question after reviewing the product documentation and having access to a Nitro sandbox:  
  
Referral programs normally work by not awarding the originator of the invite until the newly  
registered person has signed up and then achieved something on the site. So  
we thought about when the recruited person reaches 1000 points, or logs in x  
number of times and watches 3 videos etc. But there's no way to setup a  
challenge to award user X when user Y does A, B & C. Let me know if this is  
something you would consider building or if it's too much of an edge case  
for your product. We could definitely write some additional code to support  
this, but it would be fairly complex.  
  
Can you think of a way to do this using Nitro? You can use the Achievement  
Callback URL and ask the customer to write some code.

Although there is no direct way of doing this, but you can use the Trophy URL Callback to trigger a custom-designed code. When user Y does A,B,&C, create a *Challenge* that will complete on behalf of Y’s *Action.* Under the **Nitro Admin Console -> Sites -> Callbacks (Achievement Callbacks URL),** the user can upload a URL Callback. On that page, the customer can write code to handle this *Challenge completion* and decide to award user X with Y’s Actions.

* Achievements will be called during a *Challenge* completion or *Changes in Levels*.
* A HTTP 200 Response back to Nitro server will indicate a success. Requests > 2 sec will be attempted again in 5 minutes.

A query string will be sent to the provided URL, example method call can look like this:

<http://api.clientapp.com/?type=userChallengeAchieved&apiKey=FooSite&ts=1217531601&userId=johndoe&sig=12adfccc47eca94fdf0fa43098968839&trxId=12345>

4. A prospect tells you that they can only change front-end code on their website (HTML, CSS, JavaScript). They are eager to use gamification, but are concerned about the limitations that come with not being able to change the server-side code.

In particular, they want to know -

a) How is Nitro integrated without access to server-side code?

b) How do I prevent malicious users from abusing the system? Does Nitro have any built-in mechanisms to prevent fraud / "gaming the system"?

Front-end code access is enough to start using gamification with Nitro. Features like *Points, Levels, Badges* and *Trophies* and completing *Challenges* are all available to you*.* Since Nitro stores all that data on its own back-end server and provides you with the front-end tools you need to track these engagements. You can integrate with Nitro using tools like:

* Nitro.js – Nitro Javascript Library
* Nitro Flash Connector
* NML – Nitro Markup Language
* Widgets – HTML5 or Flash

Security is adjustable for both development and production environments. The Nitro API provides the necessary safety checks to prevent malicious users from “gaming” the system. For example, *Points* associated with a User that is credited / debited thru methods which would require a **secure signature match**. Methods like user.creditPoints or user.debitPoints and many other security sensitive methods require user authentication or rule-based confirmations such as whitelisting. Let’s take a look into the different security compoents of Nitro’s API:

* Keys – to make API calls into certain Nitro methods, you’d need a combination of Nitro key and secret key. API Key is public but your secret key is, well, secret and should not be placed on public files or shared.
* Signature – Some transactions are signature-based, its calculated on both the client side and server side with this formula MD5(apiKey + secretKey + ts + userId + length).
* Security Levels – *Low, Medium, High.* Used for different purposes, from prototyping to payment transactional-level security
* IP Whitelisting – with security level of Medium or High enabled, the **Nitro Admin Console** can allow certain methods to be executed from these safe range of IPs. Useful for 3rd Party Vendors or Partners testing the product.

Also, the most crucial user engagement data, like *Points, Status, Leaders, etc,*  are stored on the Nitro server and not on the client-side DOM. The client has to handshake with Nitro’s security layer to *get, set, update, and delete* user’s engagement information.

5. A prospect is getting a mixed security mod error when connecting to Nitro. Describe the process and tools you would follow to help the prospect resolve their issue. In your connection params, set the optional flag “debug : true”.

First, identify the problems by error code:

<https://bunchballnet-main.pbworks.com/w/page/53132145/Nitro_Error_Codes>

If the error occurred in the connection process to Nitro, the most common error could be:

* 101 – Invalid API Key
* 102 – Invalid Signature
* 114 – Session Expired

Make sure user.Login is called and has successfully returned. To identify the problem, open up the open *Developer Tools -> Network*, and look for name and status of the JSON method firing, and record down the error code. Also you can look into the Javascript Console,

If the request is taking over 2 seconds, due to latency, try turning down the *Security Level* to Medium or Low and see if your problem persists.

A list of the most common Nitro errors can be found here:

<https://bunchballnet-main.pbworks.com/w/page/53132252/Understanding%20Error%20Codes#id.z0uctcj4mtq>

Please record down your version of Client-Browser or iOS devices and contact me if you have further problems.