Auto Update Spec

Review Date:

Problem Statement

Problem Statement:

- Currently, we are seeing only ~2% app update rate for DT users (to latest google play build). Whereas, normal app update rate is ~31%.
- Preload users who don’t update app are not exposed to newer features that we send out and hence there is no additive impact on their LTV.

Note: Preload users who update the build and play the game have 10 percentage points higher D1 Rolling retention compared to users playing the preloaded build.

This data was considered from Trip

Objective

- To improve app update rates for preload/ xpromo STI users and hence increase their LTV.

Experimentation

- Experiment name: ignite_auto_update
- Experiment allocations:
    - Control: 80% (No change)
    - Variant 1: 20% (The new flow with Ignite SDK Implementation and auto update)
- This should be a dashboard controlled experiment.
- The DT Auto update feature will be only for DT/ Xpromo STI users and not for Google play build installed users.
    - In Google Play build, we will only initialize ignite SDK to check SDK stability, but not the auto update flow and their permission flag will always be false.

Note:

- The permission flag will be at build level and not specific to experiment, only opt-in popup surfacing will be at experiment level.
- So if the users moves to control from variant whose auto-update flag is ‘true’ then the users will be eligible for auto-update and also can access the auto update item from settings screen.

Feature Brief

- We will be using Ignite SDK (A product from Digital Turbine) to implement this functionality.
- For this functionality to work, it is mandatory to have Ignite App present in the client device.(All DT preloads/STI will have ignite already in them).
- Auto Update functionality encompasses the following major activities:
    - Check if device user has opted in for Auto Updates, and, if not already opted in, surface a UI to opt in to Auto Updates.
    - Send Opt-in for Auto Updates and optional MMP tracking URL.
    - Listen to broadcast for auto update job scheduling and app installation results.

Feature Details: SDK Implementation

- To use auto update functionality in the app, we need to implement Ignite SDK in the client app [Crossword Explorer].
- PFB the documents to refer:
  - SDK Reference
  - SDK Initialization
  - Auto Update Apps
  - Code Snippet for Auto Updates
  - Response and error handling
  - Error Codes
  - How to setup test environment

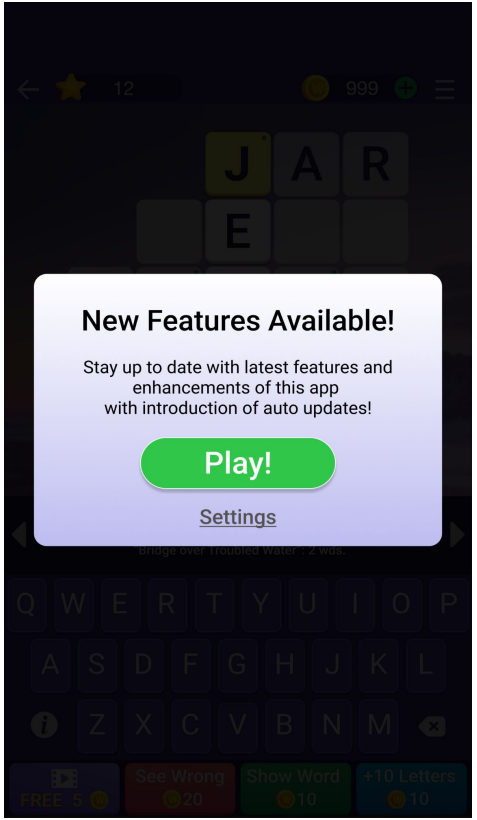
Feature Details: Check Auto Update Permission Status

- We have to check if the user has given permission for auto updates or not.
- The ***getIsAutoUpdateEnabled*** method allows you to determine if a user has already opted in for Auto Updates. It is the responsibility of the Host App to surface a UI based on the results of the ***getIsAutoUpdateEnabled*** method. For the ***getIsAutoUpdateEnabled*** method, DT returns the following values:
  - True: User has opted in for Auto Updates.
  - False: User has declined Auto Updates.
  - Null: Opt in has never been set.
- For false and null scenarios, we need to surface an ‘opt-in popup’ [Popup to introduce auto updates to the user] based on the scenario to encourage the user to sign up for auto updates. [More details in subsequent slides].

Feature Details: Popup for Auto Updates Permission

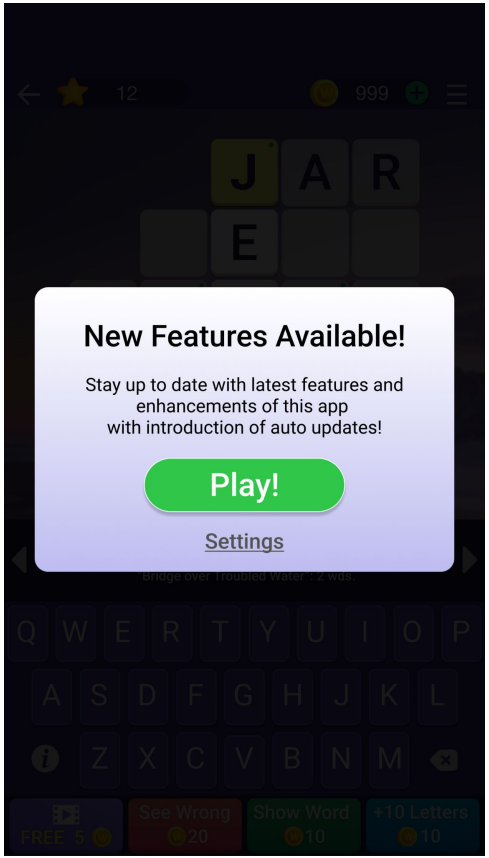
- The objective of the popup is to communicate the following to the user:
  - Notifying the user that auto-updates will take place.
  - Giving the user a deep link to (native) settings to opt-out.

- This popup can be surfaced on the puzzle screen after new puzzle view.
- The same was added in priority list for popup surfacing. PriorityOrder



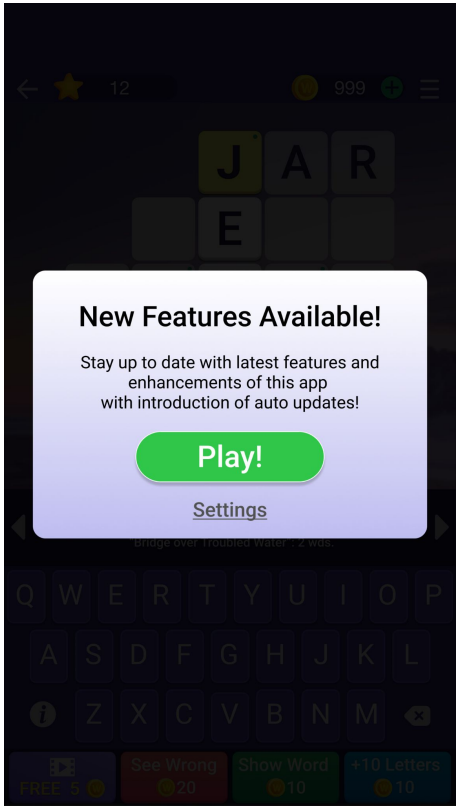
Feature Details: Popup for Auto Updates Permission

- The popup surfacing will be dependent on min_diff of 3 (runtime).
- Once user clicks on ‘Play’, auto update permission status should be set to true for these users from our side.
- Popup should surface post SDK initialization and permission check [based on the popup priority].
- Clicking on ‘Play’ takes them back to previous screen and resumes game play.
- Clicking on ‘settings’ will surface the opt-out popup and based on that we will get the permission.



Feature Details: Popup for Auto Updates Permission

- If a user kills and relaunch in this step, the permission status is still ‘null’, in this case we need to surface the popup again to the user (it won’t be counted for lifetime cap).
- If the app kill happens on popup view, then the next popup surfacing will happen based on calendar_day_cooldown of 3 (runtime).
- The popup resurfacing will have a lifetime cap of 1 (hardcoded). It should surface on Puzzle screen based on popup priority.



Feature Details: Popup for Auto Updates Permission (Settings Click)

- Inside Settings, we need to create a new item named ‘Auto Update’, [this should be the last option in list] which asks the user if they want to opt out of auto update. (with game specific featuring).
- This “Auto Update” option in the settings page should be only visible to DT/STI users with auto update code.
- Clicking on “Auto Update” option in the settings page should take them to the Auto update opt-out popup.
  - If the user clicks on Yes, then we should change permission status to ‘false’ and we do not initiate auto update for these users henceforth.
  - If the user clicks on No, then the users will continue with the gameplay and permission status will be ‘true’, and we can initiate the auto update for them.
- ‘Auto Update’ section in settings screen should be visible only to users sufficing all conditions below:
  - Preload/ Xpromo User from STI
  - User in Variant
  - ~~User whose permission is ‘false’~~
  - ~~User has clicked on settings button on DT Auto Update popup before landing on~~ ~~settings screen.~~
  - The user who viewed the DT Auto update opt-in popup at least once nonetheless of there permission status (true/ false/ null).

Feature Details: Broadcasts for App Updates [Update Jobs Scheduling]

- The SDK stores all App Update opt-in data.
- After SDK initialization, SDK schedules update jobs for those opted-in for auto updates.
- We need to refer to jobStatusBroadcastReceiver to track scheduled jobs.
- This broadcast receiver may contain the following actions:
  - **ACTION_APP_UPDATE_JOB_SCHEDULED**: Sent when App Update is successfully scheduled.
  - **ACTION_APP_UPDATE_JOB_CANCELLED**: Sent when App Update is successfully cancelled.
- DT does not broadcast when the App Update job fails to schedule. Instead, the next time the host app initializes the SDK, DT will attempt to schedule the update again.

Feature Details: Listen to Broadcast - App Update Job Completion

- We need to refer to BroadcastReceiver for installation status of App Update jobs.
- This broadcast receiver may contain the following actions:
  - **ACTION_APP_UPDATE_SUCCESS**: Sent when app has been successfully updated
  - **ACTION_APP_UPDATE_FAILED**: Sent when app failed to update
- DT does not broadcast when the App Update job fails to schedule. Instead, the next time the host app initializes the SDK, DT will attempt to schedule the update again.

Feature Details: Auto Update Flow

- If update job is scheduled by the SDK, the app downloads in the background.
- Once download is completed, On new app launch or getting app from background to foreground the app will kill, install and then relaunch again on the higher version.

Notification post update

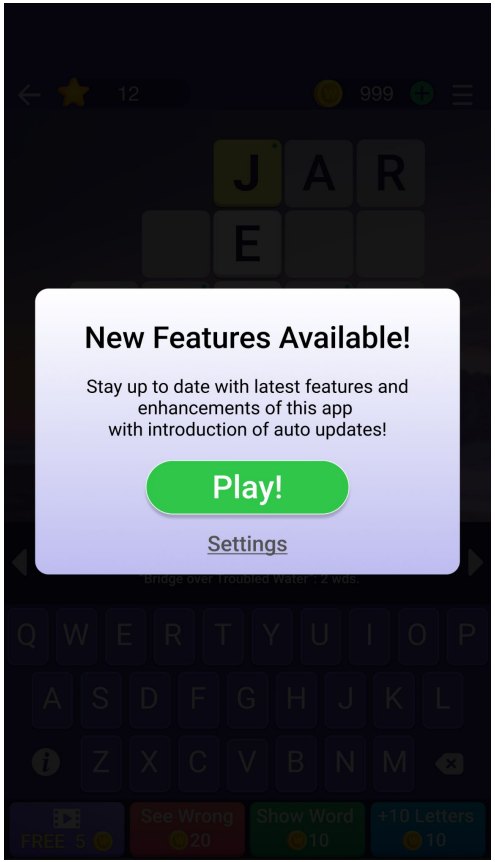
- Auto Update Notification post auto update:
  - Post successful auto update of the application we need to fire a local notif with the following copy: “Crossword Explorer” has been updated to the latest version. Tap to launch!”
  - This notif should fire everytime successful auto update happens.
  - This notif will not contribute to the overall notif cap.
  - It is a contextual notif, it is not tagged to any notif slot.
  - All notif related tracking should fire with notif name: ‘dt_auto_update’

Testing

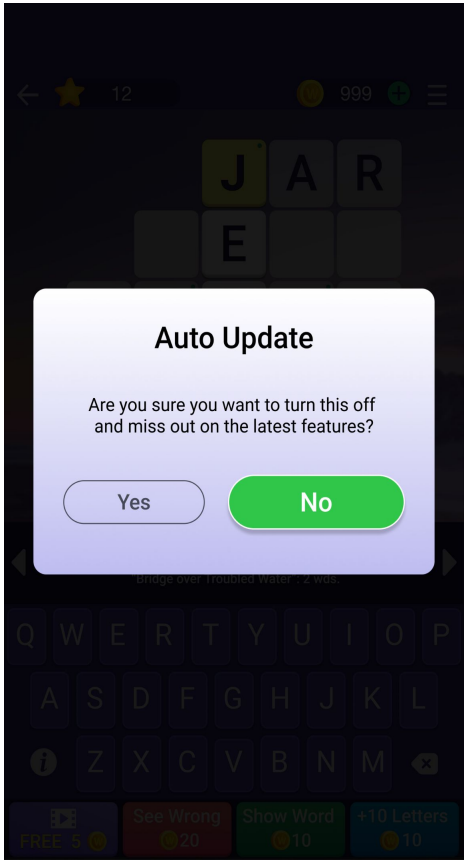
- Testing needs to be done on PROD build.
- The test needs to be done on DT device provided by DT with ignite integrated.
- The build that users will update to should be shared with DT (with auto update) & will be known as whitelisted build.
- Create a build with number lower than Whitelisted build (with auto update)& leave the device in ideal condition to check for update.
    - **Ideal conditions:**
        - A good wifi connection
        - Battery %> 50% & connected to a power source.
        - Device in idle state (No active apps running in foreground)

Note: The above mentioned things are taken from DT documentation, can contact other game QA where this feature is implemented for better clarity on the same

Art Mocks

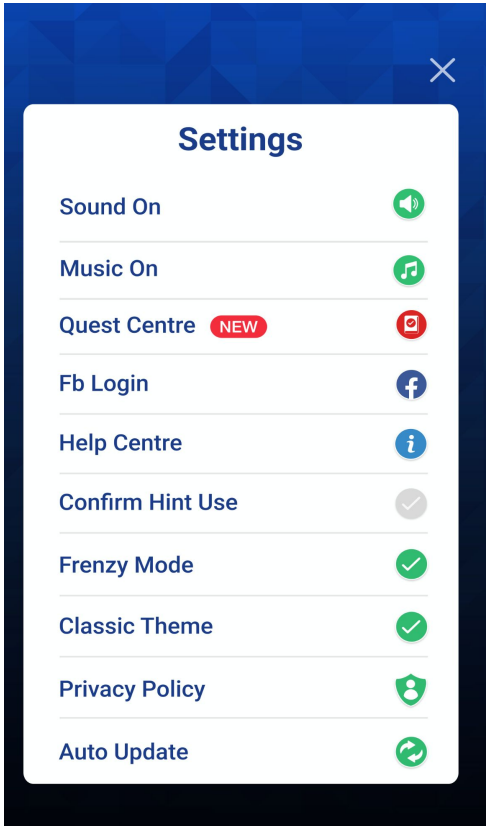


DT Auto Update Opt-in Popup

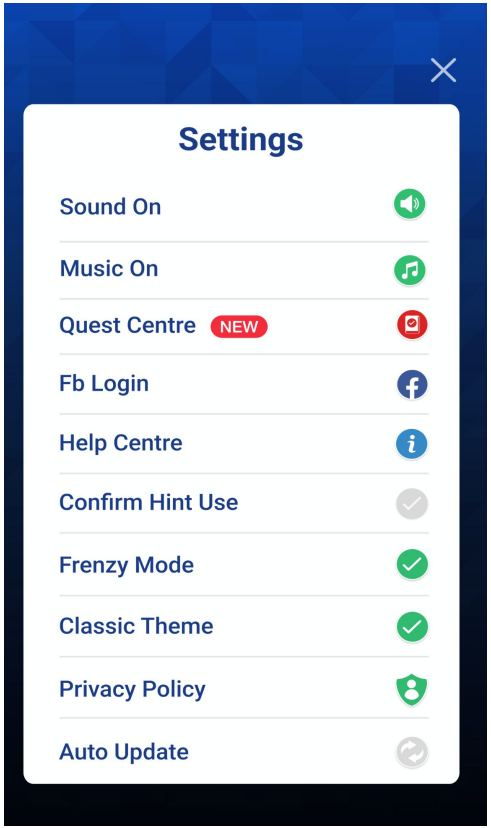


DT Auto Update Opt-out Popup

Art Mocks



Settings Auto Update surfacing (enabled)



Settings Auto Update surfacing (disabled)

Content Requirements

Auto update Popup:

- Header: “New Features Available”
- Body: “Stay up to date with latest features and enhancements of this app with introduction of auto updates!”
- CTA:
  - Play!
  - Settings

Auto update Opt-out Popup:

- Header: “Auto Update”
- Body: “Are you sure you want to turn this off and miss out on the latest features?”
- CTA:
  - Yes
  - No

Runtime

Min_diff: 3

Calender_day_cooldown: 3

Tracking

DT Auto Update							
When the DT Auto update opt-in popup is surfaced	dialog	puzzle_id	<node number>	ignite_auto_upgrade_opt-in	view/ play/ settings	<total number of popup's surfaced till now>	<puzzle id species>
When the DT Auto update opt-out popup is surfaced from settings clicks or icon click from settings screen	dialog	puzzle_id	<node number>	ignite_auto_upgrade_opt-out	view/yes/no	<total number of popup's surfaced till now>	<puzzle id species>
When the permission for DT Auto update is captured either from the popup or manul click from settings screen.	ignite_auto_update	puzzle_id	<node number>	job_scheduler	scheduled/ cancelled	< previous screen name>	
To be fired when the auto update happens for the users or it gets failed	ignite_auto_update	puzzle_id	<node number>	job_executor	success/ failed	<previous screen name>	

THE END

DT Auto Update Changes

Problem Statement:

- We need to mitigate starting auto update in build 520 due to compliance issues.
- If we start ignite_auto_update it will start for build 520 as well, which we don't want.

Solution:

- We will have a new experiment with different name [Experiment name: ignite_auto_update_new] to mitigate this issue. So that auto updates only takes place for DT build above 520.

Expected Outcome : Funnel

<Product Owner>

-

Rough Costing

	Art	Animation	Dev	QA
Target	PreFilled	PreFilled	PreFilled	PreFilled
Estimate				

Runtimes

- Permission Popup Daily Cap: 1
- Permission popup calendar days cooldown: 0
  - [If 0, it should surface for everyday user is coming]
- Lifetime Cap: 1000

Spec Status

Date of Meeting:

Approval Status: Approved/Not Approved

Comments/Notes:

Plans for v2

Plans for v2

<Feature Owner>

- This is where we move and track feature ideas that could not be accommodated in the current version of the feature.
- Please strikeout details in the v1 Full Spec and comment as v2 scope or not to be considered in v1, before moving details to this section.

Execution

Review Date:

Prod + PostProd Milestones

<Producer to enter, based on costing from Full Spec>

Tech Plan Creation:

Tech Plan Review:

Dev Start:

Playable:

Code Review:

Build Delivery:

Test Case Creation:

Test Case Review:

Alpha Start:

Release:

Tech Plan

Link:

Owner	Review Status	Comments
Design Owner		
Product Owner		
QA Owner		

Test Cases

Link:

Owner	Review Status	Comments
Design Owner		
Product Owner		
Dev Owner		

Build Review

Build Review Date:

Build Review

- Link to Build:
- Link to Feedback:

Owner	Review Status	Comments
Design Owner		
Product Owner		
UX Owner		
QA Owner		

Ramp Up Plan

<Product Owner>

Reads

Early Reads

<Product Owner>

Final Reads

<Product Owner>

Optimizations

RunTime Changes

Runtime Change Request:

Runtime Change Reason: