Microsoft

The Azure PlayFab Guide to LiveOps

Contents

Part 1: Introduction to LiveOps

It's an exciting time to work in games. The barriers that prevent people from playing together are coming down.

With cross-platform engines, cross-network services, and game streaming, the industry is on a path to bring games to any screen at any time. Cloud gaming connects players across the globe, automatic translation tech transcends language, and accessible hardware and software strive for inclusion. Now more than ever, **everyone can play together.**

It's time to think about making games differently. Games are shifting from one-off experiences to services that evolve. Developers of successful live games focus on understanding their players, meeting their individual needs, and cultivating long-term relationships. **This is what we call LiveOps.**

LiveOps is about bringing people together and giving them reasons to stay together by offering satisfying experiences. This guide is focused on a simple premise:

LiveOps help games build healthy communities that last for years.

Studios using a LiveOps strategy extend the life of their game and get a better return on their investment in development. As of Q4 2019, live services represented nearly 58% of Electronic Arts' net revenue, up from 44% a year prior.

To see the impact LiveOps can have on longevity, look at the top 10 games by revenue for 2018 and 2019. 80% of the games in the top 10 in 2018 were also in the charts in 2019, and they all operated with LiveOps.

	2018	2019
1	Fortnite	Fortnite
2	Dungeon Fighter Online	Dungeon Fighter Online
3	League of Legends	Honor of Kings
4	Pokémon Go	League of Legends
5	CrossFire	Candy Crush Saga
6	Honor of Kings	Pokémon Go
7	Fate/Grand Order	Crossfire
8	Candy Crush Saga	Fate/Grand Order
9	Monster Strike	Game for Peace
10	PlayerUnkown's Battlegrounds	Last Shelter: Survival

80% Crossover

Top games by revenue worldwide

The biggest earners are games that prioritize the player experience as their communities evolve. LiveOps also helps smaller developers create and release games with a lower upfront investment, making it possible for more creators to share their voice.

LiveOps only works with the right structure in place: server-side configurations, content data untethered from client versions, and in-depth analytics that create feedback loops to tell your team what's working and what's not. The sooner those are in place, the more chance you have of sustained success.

"New releases now only represent a part of our business, which is now focused on long-term engagement with our player communities. Our players not only play for more hours at a time but do so over a period of months or even years. We are thus able to offer them new experiences and content, thereby extending the lifetime of our games."
—Yves Guillemot, CEO, Ubisoft

The LiveOps Journey

Mastering LiveOps is about constantly learning and experimenting. While the work starts well beforehand, the "live" part of LiveOps goes through three post-launch stages:

1. Iterating your Game

After building an MVP (a minimum viable product or playable but unpolished version of your game), you can start rolling out fixes and updates. These iterations usually begin with identifying trends in engagement and retention data. From there, your team can make educated decisions on changes. At this stage the team needs a build, test-and-deploy pipeline, basic analytics, and content configurations.

2. Personalizing the Player Experience

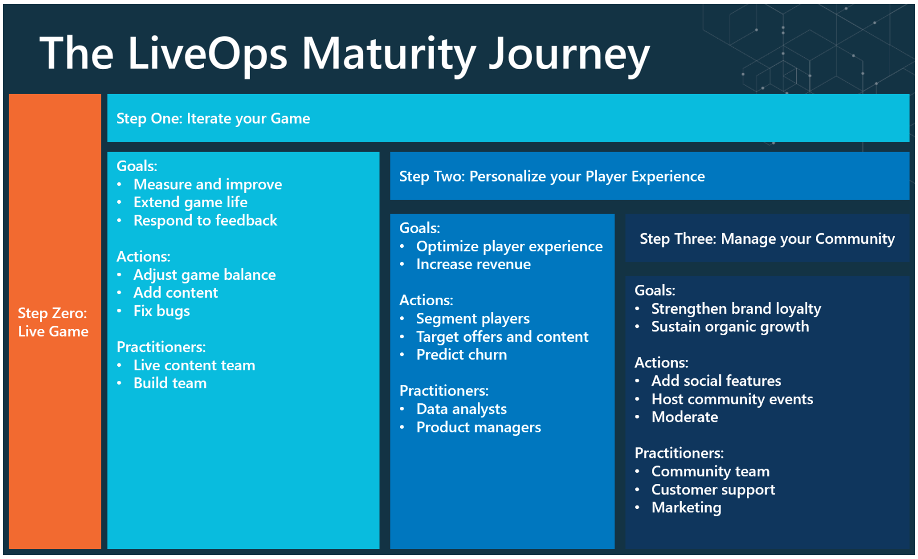
As confidence grows, your studio will be ready to segment players and more accurately target offers and content. This is the time to increase revenue and reduce risk with churn prediction and other more detailed analytics techniques. At this point, you may fold in data analysts and product managers to use more sophisticated tools such as recommendation systems and live events managers.

3. Managing the Community

At this point, the goal typically shifts to organic growth through sustained loyalty and engagement. The team may expand to include customer support staff, marketing, and community managers. The team may use more advanced tools such as guild systems, user-generated content, and multiplayer services for matchmaking, cross-network play, and communications.

"Looking ahead, we are doubling down on live services combined with our core franchises. We're investing in games that people play for longer and engage with much more deeply. This focus will continue to drive growth and profitability for the company through the remainder of this year and beyond."—**Blake Jorgensen, CFO/COO, Electronic Arts**

The next part of this guide focuses on specific LiveOps techniques ranging from launch models to live events and community management.



The LiveOps Maturity Journey

Step Zero: Live Game

Step One: Iterate your Game

Goals:
- Measure and improve
- Extend game life
- Respond to feedback

Actions:
- Adjust game balance
- Add content
- Fix bugs

Practitioners:
- Live content team
- Build team

Step Two: Personalize your Player Experience

Goals:
- Optimize player experience
- Increase revenue

Actions:
- Segment players
- Target offers and content
- Predict churn

Practitioners:
- Data analysts
- Product managers

Step Three: Manage your Community

Goals:
- Strengthen brand loyalty
- Sustain organic growth

Actions:
- Add social features
- Host community events
- Moderate

Practitioners:
- Community team
- Customer support
- Marketing

Part 2: LiveOps Techniques

Here we'll dive into the concepts and techniques that make up the LiveOps toolkit, with examples and tips on implementation.

Connecting to Your Player Base

LiveOps developers know that players and communities evolve. When creating a game, we're not movie directors with a singular vision, but more like TV network program managers, curating content to suit the audience.

LiveOps games are player-centric and react to player desires and needs in everything from initial attraction to customer support.

Acquisition

In the past, player acquisition focused on building hype to maximize day-one sales. With LiveOps, acquisition is an ongoing process. There are many ways to attract players:

- **Store discovery** marketing helps players find your game in a marketplace such as Steam or the iOS App Store. You can improve acquisition by testing different store artwork and descriptions.
- **Public relations** wins, such as awards and great reviews, put you on players' radars. Reach out to outlets matching your core demographic for the best results.
- **Paid digital advertising** is commonly used, but the effectiveness varies. Your return on investment (ROI) can change drastically depending on the ad platform, whether you pay per impression (CPM), per click (CPC), or per acquisition (CPA), and how well you target your audience.
- **Cross-marketing** is a form of advertising limited to your game portfolio. Cross-promoting with other studios is another way to acquire players.
- **Influencer marketing** can be a great way to get players excited and engaged with your release. Some games are designed with live streaming specifically in mind, for example leaving space for streamer video overlays or offering streamer-branded skins. Influencer marketing efforts vary in effectiveness and cost depending on the influencer and platform.

- **Social installs** are driven by invitations from friends or social media posts. Players are more likely to try a game recommended by a friend than an advertisement. Success depends on making it easy to invite friends, and incentives to invite (and accept invites).
- **User-generated content** can drive growth, especially if players can create and share skins or levels. It takes work to foster a welcoming community for creators, and moderation can be an issue.
- **Reactivating lapsed customers** and getting them to reinstall can be a challenge. For this to work, you need a mechanism to message those players (usually push notifications or email) and an incentive to return.

***Pro tip:* Track player engagement and retention based on source of acquisition and look for trends.**

Retention

Retention measures how many players return to your game. It's one of the only data-supported ways to know if players enjoy playing. Here are some techniques that keep players coming back:

- **Adding more content** such as new levels, enemies, or play modes is often the default for increasing retention. It can be tricky to balance the cost of creating quality content and releasing it in a timely manner. Often a steady trickle of content your team can deliver consistently works best.
- **Game systems** that encourage players to invest time and resources into progression tend to be sticky due to the psychology around fear of loss. Character, world and economic progression become even more effective when coupled with the ability to personalize content. Doing this well requires excellent game balance, and benefits from data informed iteration as the community matures.
- **Social activity** built into a game fosters relationships and incentives to return. Human relationships drive some of the highest retention in games, so make space for players to work toward shared goals, build trust, and communicate.
- **The core loop** of a game, if deep enough, can keep players coming back. The game Go is an excellent example, with simple rules that allow for endless new strategies. Game balance, room for player experimentation, and accessible controls all help improve a core loop.

However you try to get players to keep coming back, be careful not to desensitize them to your messages. Try to communicate only what's interesting and valuable, and mix up rewards so they don't become background noise.

***Pro tip:* Test different rewards for reactivation by segmenting your players and comparing performance.**

Engagement

Engagement measures both how fun your gameplay is and the strength of your community. Active communities engage with a game by playing, providing feedback, and promoting (discussing online or in person, creating fan content, and so on).

> *"Even as a small studio, we prioritized community early on because players will appreciate it if you talk to them and respond fast. Even if you can't fix things immediately, they will honor if you make the effort and answer them."*
> —Cem Aslan, CEO, ColdFire Games

A solid engagement strategy is essential for LiveOps. In fact, engagement is the only KPI that some studios measure.

Player habits vary from game to game, and a deep dive into your analytics can tell you which metrics most accurately reflect engagement. For example, d*aily active users* may work to check the pulse of some games, while *total monthly sessions per player* may be more accurate for others.

***Pro tip:* Look at how studios with games like yours engage their community as a baseline for your own engagement efforts.**

Support

One of the most critical responsibilities of your team is to ensure your game is working as intended and make it right when things break. Support is important for resolving player frustrations and as a valuable source of feedback.

Your team needs the tools to isolate and identify problems, fix or escalate them to a QA team, and communicate with players throughout the process. Some of the most valuable support tools include:

- **A ticketing system** so players can submit problems, and support teams can respond.
- **The ability to look up individual player profiles** and make manual changes.
- **A way to suspend or ban players** who violate your community code of conduct, and a way to reverse bans made in error!
- **A way for players to upload crash logs** for QA to investigate.
- **Ways to send messages to players**, both individually and to everyone.

***Pro tip:* Be sure your analytics track customer support data as well. Changes in support contact and resolution rates (e.g. number of support tickets opened and closed) can indicate larger issues.**

Data for LiveOps

Analytics

Timely, in-depth performance data helps you understand players and deliver content they'll love. Here are some common LiveOps metrics:

- **ARPU (Average Revenue Per User)**. The total number of unique players divided by total revenue in a given period. Can be used to gauge the general business health of your game.
- **ARPPU (Average Revenue Per Paying User)**. Like ARPU but limited to purchasing players. Useful for gauging monetization strategies, such as store design or DLC.
- **Unique Logins**. The number of unique players connecting to the game in a given period. Unique Logins is a great measure for acquisition, as it indicates newly acquired players.
- **Conversion Rate.** Number of players who made purchases divided by Unique Logins. Gauges your success at converting free players into paid players.
- **Retention.** The number of players that logged into your game during two specific reporting periods, gauging how well your game keeps players interested. Common reporting period intervals include 1-day, 7-day, and 30-day.
- **Average Session Length.** A straightforward metric used to gauge how long your gameplay loop stays fun or keeps players' attention.
- **Session Frequency.** The number of players logged in, divided by total number of login events. Indicates how often players engage with the game.
- **LTV (Lifetime Value).** Reflecting the ROI per player, LTV is the total number of unique players divided by total revenue generated.
- **Errors.** The number of errors generated by code. Gauges how stable your game is.
- **Content Logs.** Counts occurrences of specific content logs unique to your game being triggered. Gauges popularity, stability, and engagement of specific game content.

***Pro tip:** Unique Logins is often a good indicator of acquisition, Retention naturally) for retention, Session Frequency for engagement, and Errors & Content Logs for support.*

These metrics can be used to learn more about player behavior, track the impact of changes, and gauge performance.

Most analytics solutions provide at-a-glance reporting for basic metrics. For in-depth analysis, you'll need to run custom queries. That way you can isolate datasets within specific timeframes and segment your player base to test solutions.

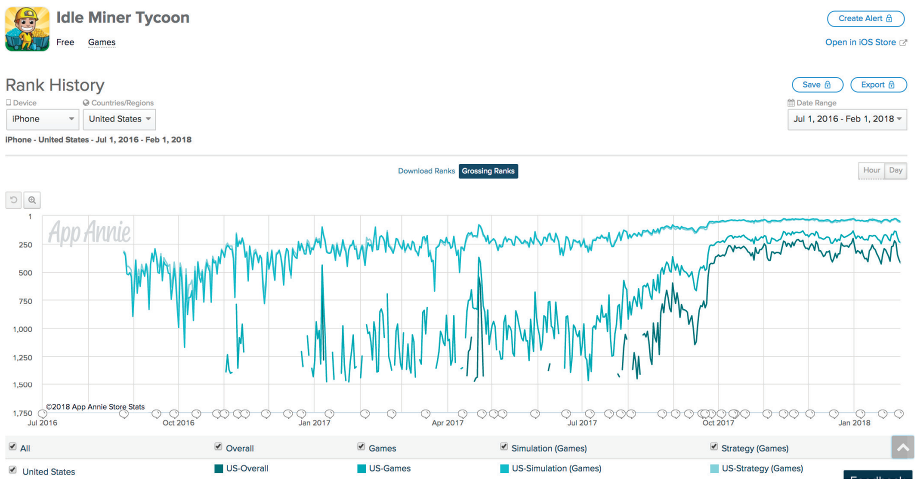
Another important consideration is the optimal frequency of data collection. Some metrics are best reviewed over long periods of time (e.g. Avg. Revenue), while others benefit from constant real-time updates (e.g. Errors). You might not figure out the right frequency until you've spent a good amount of time collecting data.

> *"I believe it's important to stay nimble in a constantly changing environment where you are willing to pivot to a new strategy at a moment's notice based off of rigorous data analysis."*—**Renee Ya, Founder, Tiger Byte Studios**

LiveOps Data at Launch

For Kolibri Games' *Idle Miner Tycoon*, the studio first launched a very lightweight version of the game and then iterated based on player data.

Instead of boosting acquisition through marketing or app store advertising, they built traction by focusing on early retention metrics such as daily active users, session length, and crashes. Even before there was enough data to spot trends, they used direct player feedback to tune the game. The game's climb up the iOS App Store charts in its first year shows how their strategy paid off.



Idle Miner Tycoon: focusing on customer traction pays off in the long run

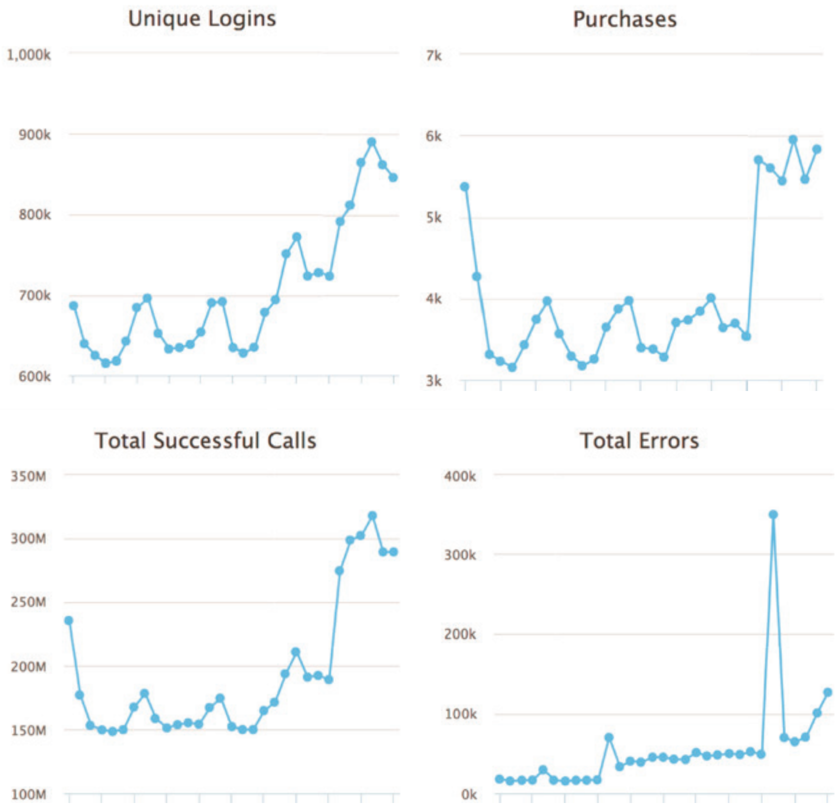
Subnautica had a similar story. After release, the team kept improving the game and built a huge community by implementing direct player feedback through a public Trello community board, letting users log bugs directly, and holding community votes on what to work on next.

Perhaps the best example is *PlayerUnknown's Battlegrounds* (PUBG), whose MVP and subsequent updates kickstarted the Battle Royale genre.

***Pro tip:* Use data visualization to find trends that help you plan content and performance updates.**

Data Visualization

When looking at visual reports of your analytics, you may find correlations between engagement or performance with weekly cycles or live events. Get acquainted with these trends and look for outliers that could indicate problems.



The right data visualization shows obvious trends and outliers.

You can also drill down into event history with custom queries to track and optimize performance, and benchmark against past results.

***Pro tip:* Use analytics to dig into specific problems, rather than focusing only on the headline metrics.**

Analytics are most useful when the data is actionable. Knowing your retention rate is important, but offers no insight on how to fix it. For that, you need to do a deep drill-down or segment your audience and experiment. Asking “How do we improve retention?” should lead to more detailed questions like “How are players who reached level 50 different than those who stopped playing at level 10?”

These insights can help your team craft additional content, improve the core gameplay loop, and please a growing community.

Segmentation

While every player is unique, similarities and trends often appear among groups of players. Segmenting groups is a necessary step to deliver the best content to the most players. Here are some key benefits:

- **Designers segment players based on in-game behavior** to understand their needs and develop player-centric content.
- **Monetization teams use segments to understand spending patterns**, identify fraudulent behavior, and predict revenue.
- **Marketers create custom segments and optimize messaging** for each to acquire or engage players. Tracking players by acquisition method helps evaluate which channels are worthwhile.

“The most important thing about the testing aspect is the cohort and the segmentation. So, understand who your players are, segment them up front, and then put them through the experiments to understand the impact of each one of those experiments.”—**Scott Koenigsberg, Head of Product, Zynga**

As your game gains traction, you may see segments form naturally, and your analytics toolset should let you define custom segments in a way that makes sense for you. Those segments might include “High XP Players,” “Players Who Made Purchases,” or “Lapsed Players,” depending on your goals.

A good starting point to segment players is a basic “funnel.” For example, you can define some simple segments like this:

- “new players”—anyone up to level 10
- “non-spenders”—over level 10, LTV = 0
- “spenders”—where LTV > 0 but < $100
- “high spenders”—where LTV > $100

Once you do this, it's easy to track your progress getting players to move through the funnel from one segment to the next. Other common player segments include time of adoption (e.g. beta players vs. launch players), platform (e.g. console vs. mobile), or by country.

Once you've defined the most important segments, you'll want to be able to treat players in each of those segments differently. For example, you might show different gameplay tips to each segment to encourage them or only show in-game ads to non-spending players.

Pro tip:* *Using machine learning can help you automatically segment players and simplify analysis.

Experimentation

"Nobody knows anything...not one person in the entire motion picture field knows for a certainty what's going to work. Every time out it's a guess and if you're lucky, an educated one."
—William Goldman, Oscar-Winning Screenwriter

What Goldman said about films used to be true about making games—you guessed what audiences would like, spent months or years making it, then released it and hoped for the best.

But now, LiveOps gives developers the tools to tailor content directly to player desires, even as they change. One of the most reliable ways to do that is through experimentation, so familiarize yourself with some of the basics.

Know What you're Testing

Experimenting can give you hugely valuable data, but not if you don't know what you're looking for. Good experiments have a hypothesis or some sort of goal KPI to change. Decide on one beforehand, or your team may waste time deciphering the results instead of getting clear direction.

Test Early and Often

Often teams want to test their assumptions before investing in development, but don't have enough data to make an informed decision. So, they're stuck. But teams can get useful information from "scrappy" testing, and your creativity as a game developer can be an asset.

There are some quick and cost-effective ways to test basic concepts before putting them into your game, for example measuring click-through rates on digital ads with identical text but different art styles. Tests such as these can keep your team's momentum and minimize investment risk.

A/B Testing

A/B testing (or split testing) is seen as the gold standard when it comes to running experiments because you can usually get statistically significant results to measure changes of any size.

The size of your audience can affect how complex your testing can be. A game with millions of players can easily test subtle changes, but one with a smaller audience will only get significant data from tests with stark variations. The same goes for how many tests you can run simultaneously—a smaller player base means fewer simultaneous tests are statistically reliable.

***Pro tip:* When A/B testing for a set period, be sure the test encompasses at least a whole week to measure fluctuations between weekday and weekend players.**

When running tests, be aware of cross-contamination. Gamers are famous for dissecting every aspect of the games they love. If you have an established player base, you can't assume your experiments will go unnoticed. In fact, players may get upset if they feel one group is getting a leg up on another.

Make sure if one player sees something different from another, they can clearly understand why.

Monitoring Player Behavior and Feedback

It's possible to treat every change to your game as an experiment if you're intentional about it. If you monitor user feedback on channels such as Discord or Facebook, you can gauge player reactions to changes. This approach may give the most vocal players too much influence, but it can be effective when player data is in short supply.

Another way to gather feedback is through in-game polls and micro surveys. Surveys should come during down moments in gameplay, such as between levels, and should only ask one question at a time. A simple poll such as "Would you recommend this game?" is commonly used because it's easily repeatable as changes are made, and it can lead into a social media recommendation if you incentivize it. Players enjoy sharing opinions, so a well-timed survey can improve a game experience.

"Failed experiments are an important part of the process to learn, especially for the team running the experiment. The companies that learn fastest, usually win."—**Sebastian Knopp, Growth and Innovation Consultant**

LiveOps Analytics – Best Practices

- **Learn which metrics best capture performance** for your game's KPIs, and set appropriate periods to monitor and review them, whether hourly, daily, or monthly.
- **Test gameplay mechanics early.** It's harder to test changes to core gameplay post-launch, after players have developed expectations.
- **When players have problems, analyze event history** to determine what happened, and look at ticket open and close rates to find wider issues.
- **Use limited-time events to test changes** to gameplay—players are often more tolerant of gameplay changes when called out as events.
- **Chart out the "funnel" progression** for players in your game and experiment with ways to motivate players to move through your funnel: in-game messaging, external marketing, email, etc.
- **Ensure your analytics tools let you view KPIs by segment** so you can compare performance. You may want to view your A/B test results by segment as well.
- **Establish a clear success metric** to gauge the impact of tests. This makes comparing effectiveness easier.
- **Test qualitative factors** by polling players with in-game surveys.

Launching with LiveOps

There's a lot that can go wrong during launch. But LiveOps make it easier for studios to dodge common pitfalls. It helps to put together a designated LiveOps team.

That team should consist of members responsible for technology, in-game content, user acquisition, monetization, community, and customer service. It shouldn't be a smattering of reps from different departments, but rather a close-knit team working toward common goals with clear roles and expectations.

- **Game pillars and content strategy** should be well-defined, and the team should be intimately familiar with game features, balance, game loops, and how those will expand in the first few months.
- **The feedback and data pipeline** should be established, as well as the KPIs and response methods for player feedback.

- **You need live reporting and response plans** for handling unexpected issues, as well as error reporting and plans for staffing team members to resolve player support issues.
- **Don’t forget game direction and ethical guidelines.** Just as important as gathering feedback is understanding how that feedback is to be acted on. Lines should be drawn before player feedback and data is flowing in, to avoid losing direction.

> *“With a LiveOps game, the real work starts with launch instead of ending there. And that’s a big challenge for game developers”*
> —Sebastian Knopp, Growth and Innovation Consultant

Soft Launch

There’s no such thing as a dry run in live games. If you’re investing in a long development cycle prior to launch, the best decision may be to plan a **soft launch**. A soft launch can give you nearly as much insight into your gameplay and operational performance as a full launch, with a fraction of the risk. Soft launches can show you whether your game connects with its audience, track whether players are behaving as expected, and tweak monetization—without stopping your team’s momentum.

One way to soft launch is to choose a smaller geographic area, ideally with the same language as your core audience (Australia, New Zealand, and Canada are popular options for games in English), and run your game for a few months.

You can also soft launch by limiting your initial audience with an Early Access or Beta period. Then, pay close attention to the core engagement metrics (Retention, Avg. Session Length, etc.) to fix any flaws before full launch. Did everything work as planned? How did customers react? Did your reps break anything? If you experienced any surprises, consider how you might prevent similar surprises in the future.

Pro tip:* *During soft launch, confirm that you can update the game without causing disruption to players – and make sure that you can roll back changes if problems arise during deployment.

Try to run retrospectives frequently and be prepared to iterate quickly to make improvements because while a soft launch is a smart investment, the quicker you can move into full launch, the sooner you grow revenue. That said, many developers are moving away from soft launches in favor of lean launches.

Lean Launch

A **lean launch** deploys an MVP version of the game, connects with a target audience, and then tunes the game based on player data and feedback. A lean launch builds traction by focusing on the features and functions that resonate the most with your audience. The trick? You need a reliable data pipeline, a smaller manageable audience without inflated expectations, and you need to be able to adapt your game quickly.

> *"As a small, indie studio, you don't have the money to do user acquisition for a soft launch. So, you just go ahead and publish the game. With Idle Space, we went live and released the content we had immediately, to get a first glance at how users are playing the game and if they liked it. And then we tracked engagement, and A/B tested a lot while the game was already live."*
> —Renee Ya, Founder, Tiger Byte Studios

Collecting and analyzing your **crash data** and **retention metrics** is a must, even when taking a lean approach. With real-time analytics, these launch models help solve the problem of investing too much time in up-front development vs. the benefit of releasing a game sooner. But they're dependent on an effective LiveOps pipeline that allows for developing several pieces of content at once, and agile deployment.

LiveOps Launches – Best Practices

- **Assemble a LiveOps team** (or designate individuals) responsible for decision-making and rolling out iterations. Make sure they have defined processes in place for making decisions as well as regular review sessions.
- **Develop a calendar** for the first few months after launch to keep everyone in sync across development, content rollout, promotions, and events.
- **Put validation checks in place** to avoid mistakes in running events.
- **Rehearse key LiveOps tasks** such as adding items to the catalog, adding new content, and sending in-game messages.
- Ensure your team has a way to **roll back changes** that might cause unforeseen issues.
- **Set roles and permissions** so team members can't make changes they're not trained for.

Game Updates

"With games-as-service you have to keep players engaged with new content. There cannot be an endgame."—Pascal Clarysse, CMO, Eden Games

Today, games of any size can build an audience quickly if they resonate with players. Keeping a game at the top of the charts is a different matter. Strategic game updates are crucial to maintaining player interest.

Game updates aren't limited to new levels or game mechanics. They can consist of new items for purchase, events, balance patches, bundles, or anything else that encourages a player to come back and play more.

***Pro tip:* Invest time trying to understand your players through research, in-game data, and player feedback outside of the game.**

Understanding your player base is a key element in designing and delivering relevant updates. How old are they? Where do they live? What motivates them to play? Creating player profiles and segmenting can help you get in touch and create more relevant content for your community. It can also help you determine how often to update the game.

Update Strategies

Frequency and consistency are as important as quality when making updates. Managing player expectations and habits is essential to long-term success.

Building a consistent update pipeline can be challenging, especially if your game releases on platforms that require certification. So LiveOps teams need to employ several techniques to keep their games fresh.

Choosing what to update is an art and a science. You want to maintain or increase engagement and monetization without burning players out. Segmenting your players and targeting them with different updates (if you have the resources) is a great approach, but even thoughtfully timed updates can work for teams with fewer resources.

***Pro tip:* Experiment with different intervals between updates (weekly balance changes, updates each month, major content updates every 3 months, etc.) to see if they impact engagement or retention.**

In the past, studios had to release a new client version to roll out updates. But client updates require testing and store platform approvals, and can force players to download large files. So, unless you can invest in a large, dedicated build team, save client updates for entirely new features or large assets.

Teams should focus on content configuration changes deployed via cloud or onto owned servers. This requires a different pipeline than client updates. For this, assets such as art and gameplay logic are included in the client, but how those assets are displayed to players is driven by server-side logic that can change without a client update.

To do this effectively, plan your content architecture in advance and move as much of your game logic as possible onto the server or cloud. Backend services let you upload to the cloud and manage the logic to make changes in real time. In PlayFab you can even toggle live events on and off, change which items are available for purchase, run sales, and message players, without worrying about traditional build processes.

> *"Every three months we add new features, so we're constantly improving it. We're continuously coming up with new operators [characters], environments, storylines, and offensive and defensive gadgets. We're committed to making [Rainbow Six] Siege fun to play for a long, long time."*
> **—Sébastien Puel, Executive Producer, Rainbow Six Siege**

LiveOps Game Updates – Best Practices

- **Make a list of everything in** your game that could be considered "content," and a plan for updating that content over time. Consider new maps or levels, in-game items, quests, events, achievements, cosmetics, and UI changes.
- For each type of content, **decide how you will update** that content. Does it require a client update, or can you update entirely on the server?
- **Plan how assets will get to the client**. Will you deploy a client update or download individual assets from a content delivery network (CDN) at runtime?
- **Think about offline mode**. If you depend on the backend for updates, make sure your game still runs even if it can't connect to the server, if only for a limited time. This requires some form of content caching on the user's device.
- **Vary your updates**. Consider both limited-time content that eventually goes away, as well as permanent content updates.
- Consider **targeting new content to specific player segments**, at least initially.
- **Invest in a few months of forward-looking content** by launch and maintain a focus on new content creation once the game is live.
- **Consider cloud streaming or downloading assets** in the background during gameplay to reduce friction.

Live Events

A **live event** is any temporary but meaningful change to a game's content.

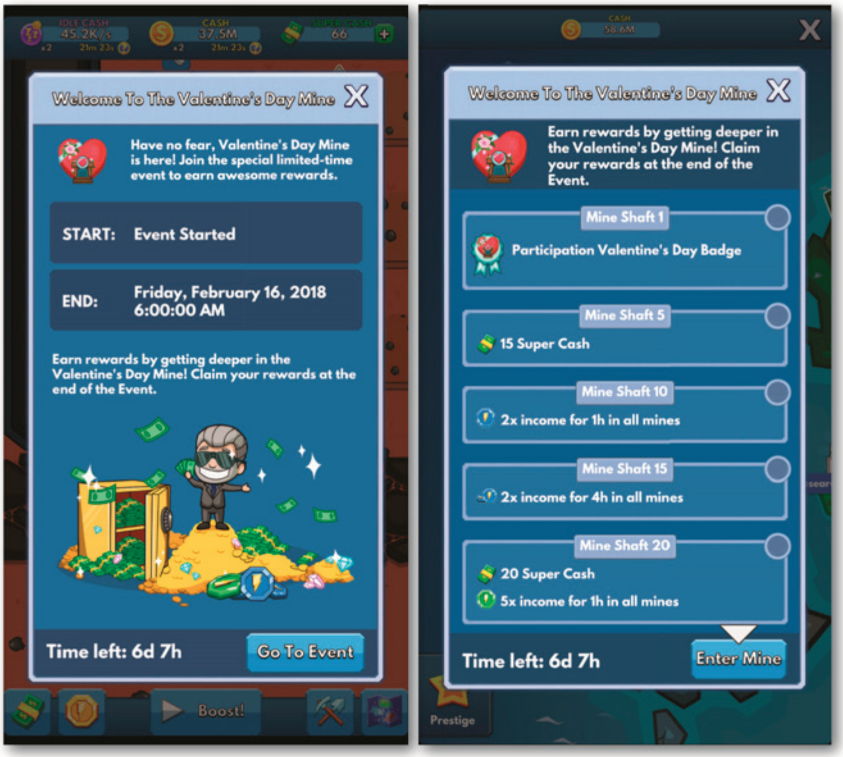
Running events is an essential LiveOps skill. Good events are great for engagement and monetization, and *great* events do both without causing player burnout. Successful events often include:

- **A measurable goal** the team can design toward. Make each event a part of your iterative, learning mindset and measure performance against goals.
- **A limited-time period** gives each event a sense of urgency and uniqueness.
- **Engaging themes and content** draw attention to the gameplay updates or options players might not know about.
- **Surprise and predictability** build expectation and excitement.
- **A sense of community effort** reinforce the feeling an event is a special shared moment, for example a high score challenge for the community to reach within a time limit.
- An effective means of **communicating with players** is necessary to quickly draw people who aren't currently playing into the event.

Live Event Frequency

The right frequency of events will keep players engaged without overwhelming them. Testing and experimenting with different options and comparing the engagement, retention, and monetization metrics will help you find that happy medium.

For example, *AdVenture Capitalist* developer Hyper Hippo Games saw a data trend where activity would spike enormously during an event, and then fall to below-average numbers for a few days. By experimenting with event timing, they were able to settle on an event schedule that raised their baseline engagement while also minimizing lapsed players. For *AdVenture Capitalist*, that optimal frequency happened to be every 4 to 6 weeks.



Welcome To The Valentine's Day Mine

Have no fear, Valentine's Day Mine is here! Join the special limited-time event to earn awesome rewards.

START: Event Started

END: Friday, February 16, 2018 6:00:00 AM

Earn rewards by getting deeper in the Valentine's Day Mine! Claim your rewards at the end of the Event.

Time left: 6d 7h

Go To Event

Welcome To The Valentine's Day Mine

Earn rewards by getting deeper in the Valentine's Day Mine! Claim your rewards at the end of the Event.

Mine Shaft 1
Participation Valentine's Day Badge

Mine Shaft 5
15 Super Cash

Mine Shaft 10
2x income for 1h in all mines

Mine Shaft 15
2x income for 4h in all mines

Mine Shaft 20
20 Super Cash
5x income for 1h in all mines

Time left: 6d 7h

Enter Mine

Idle Miner Tycoon by Kolibri Games runs regular monthly events and specialized holiday events, with unique rewards for participating in each.

Consider running repeatable events, especially if you can find a natural fit. Holidays work because players will be more understanding of temporary changes, and often have more time to play. *Clash of Clans* and *Overwatch*, for example, make a practice of running major events for Halloween and the winter holidays.

***Pro tip:* Contests or tournaments can run at a smaller scale and require less overhead than full-game events. Adding a special, limited-time leaderboard for a specific in-game goal is a common event.**

Events can also run in parallel, not just in series. *Clash Royale*, for example, runs daily and weekly quests, in addition to special events with bigger rewards.

Event Calendars

Running a live game can be like directing a huge, complex ballet, so it's recommended that you **build and maintain a calendar** view of your live events shared by appropriate departments.

Organizing this way helps manage one of the key risks with running events: player fatigue. It's normal for players to slow down their activity after an event, and you'll typically see drops in engagement and monetization metrics.

But trying to run as many events as possible to get those player spikes can lead to burnout and eventually opting out completely. It helps to have a holistic view of events to avoid this.

Building Events into your Game's World

Events often have a "fiction" that masks the actual changes associated with the event. The changes themselves can be simple, such as a higher drop-rate on a rare item or a new item bundle for sale. However, the *fiction* of the event might be something dramatic like "A Midsummer Night's Scream" with an equally thematic description. This helps make each update feel distinct and gives the community a shared history to enjoy. The community's ability to look back fondly on events or moments in the game's history is as important as the content of the events themselves.

"Anything can be an event – anything in the game. Timebox it, reward it, there you go. You have something more engaging that you'd be doing anyway, but here you are getting that extra bonus for taking part."
—Peter Eykemans, Director of Publishing, Kongregate

Live Events and Player Communication

Great player communication is critical to the success of live events. It's important to be able to communicate with players dynamically through channels that match how they engage with your game. Short-session mobile games will want to use **push notifications** to inform players, while a premium MMO game should establish regular communication via **email bursts**. Both notifications and emails can be effective tools for re-engagement, so make sure you give players enough lead time to get interested and return to the game.

A dedicated **social media** team can also engage with players, collect feedback, and answer questions about events and other content updates. Make sure to use all the channels available. Facebook, Discord, and Reddit are great places to start. Being as transparent as possible throughout the process helps build trust and foster a player-centric community.

LiveOps Live Events – Best Practices

- **Make a list of everything** you might want to change as part of an event. This might include new game levels and quests, unique in-game items, store discounts, and special leaderboards.
- Prepare to **run events from the server**, without a client update. This means exposing game configuration settings on the server so you can tweak the game during events.
- Find natural ways to **promote upcoming events in-game**, perhaps through triggered content logs or in-game notifications.
- **Capture event data in your data warehouse** so you can report on and identify winners of events after the fact.
- **Let your team be flexible** when creating events. You never know what they'll come up with to keep the game fresh.
- **Set goals for events** you can measure against. These can range from increasing KPIs to triggering a desired community effect.
- **Keep an internal event calendar** so your team knows what's coming up and when. Use the calendar to avoid situations that could result in player fatigue.
- Use events to **experiment with ideas** to see what stimulates the player behavior you want. Track and analyze performance metrics to gauge community response.
- **Establish an event framework** for your team that separates the repeatable parts of event management from work that's unique to each event.

Monetization

If you want to stay in business, your game must generate revenue, and prevalent strategies for doing so have come a long way.

How you monetize shouldn't be solely about maximizing revenue, but also satisfying your players' short and long-term desires, building a lively community, and keeping the work within your team's capabilities.

Choosing a Monetization Strategy

The earlier you settle on a monetization strategy, the better. And while every game is different, every discussion about monetization should consider:

- **The kind of game you're building**, the platforms you plan to target, and the demographics of your audience. You may have a great design for a puzzle game, but if data shows that puzzle gamers don't make purchases, you will be fighting an uphill battle.
- What you want your relationship with players to be like. Choose a strategy that aligns player needs with your revenue goals. If players love character customization, offer cosmetics in your in-game store. **Give them what they want.**
- **Ethical guidelines for monetization.** Teams can get caught up in the nuts and bolts of monetizing, so make sure the team agrees to guidelines before moving to a live environment.
- **How your competition is monetizing**, and how you can exceed expectations of players you have in common.

Microtransactions

One of the most common forms of monetization is microtransactions (or "in-app purchases"). Microtransactions are an *à la carte* strategy where players can purchase only what they want. This relies on players making multiple small purchases over a game's lifetime, so you need to show the value to your players.

That **player value** will vary for each game. For example, collectible card games offer exponential value because more cards give the player more choices in deck building.

The wrong offerings can also alienate players, for example if an item offers a competitive advantage to buyers. Be sure to consider game balance and community dynamics.

Types of Microtransaction

There are many types of microtransaction, and successful games often offer a variety of them. Here are some of the most common, with their benefits and drawbacks.

- **Cosmetics** are items that affect the physical appearance of the player character or game interface. These are best used when you're able to offer a wide variety of options, and ideally packaged in bundles for purchase.

  + Lets players express themselves, giving them a new goal, especially in multiplayer games.

  - May not be as effective in single-player games. Often a steady content pipeline is needed to maintain interest.

  - Relying solely on cosmetics for monetization can result in a lower percentage of paying players than a power-based monetization system. (Consider that Socializers are one of four player categories in Richard Bartle's taxonomy.)

- **Account Upgrades** are permanent enhancements to a player account, such as increased inventory space or faster level-ups.

  + Helps players avoid buyer's remorse (as upgrades are permanent).

  - Can inspire animosity in your player base between those with and without the upgrade, especially if it provides a gameplay advantage.

- **Consumables** are items that can be used once for a temporary effect, such as an XP or currency booster.

  + Can be priced cheaply for budget-minded players.

  - Making multiple small purchases can turn off players. One-time use items can lead to player anxiety as they wait for the perfect moment to use.

- **VIP Programs** are subscription-based programs giving players access to perks and enhancements. The best of these offer an obvious increase in value for the investment. *Fortnite* integrates progression into its Battle Pass which gets players doubly invested. Properly executed, such a system incentivizes engagement, retention and monetization. (Be sure to consider the perceived time vs. money equation carefully.)

  + Eliminates barriers to entry and inconvenience by bundling multiple benefits into one purchase.

  - Subscription price tags can be a turn-off to budget-minded players.

- **Content Access** to new story campaigns, maps, or game modes is commonly used by premium games in the form of DLC, and can also be a great offering for smaller games.
  - + Can result in great revenue generation if the content is high-quality.
  - - Certain types run the risk of splitting players into haves and have-nots, creating animosity in the community.
- **Random Boxes** (or loot boxes) are items players can purchase without knowing exactly what they'll receive. These must be implemented with the utmost care, as they can be divisive.
  - + Opening boxes can be exciting for players, if a certain amount of value is guaranteed.
  - - Are heavily regulated and even illegal in some countries and markets. May confuse or alienate players if the value prop isn't made clear, or if specific items are only available through them.

> *"10 years ago when "freemium games" came out, the mentality was just trying to get players to spend. And now it's much more than that. It's a full experience that you want to provide to your players so that they keep coming back, and that's going to encourage them to pay. But first and foremost, they need to come back to your game."*
> **—Tammy Levy, VP of Insights and Analytics, Kongregate**

Store Management

Active store management is a must to keep players engaged and making repeat purchases. Keep in mind some of these elements of in-game store management:

- **Store presentation and layout.** An easy-to-use store is a productive one. Make in-store items easy to identify, along with their benefits and price point (especially if the price has dropped recently).
- **Catalog management.** New items and offers should be entering your store regularly. (A good rule of thumb is once a week.) Having a rotating catalog can keep your store feeling fresh and not overly cluttered.
- **Pricing.** Price is often what determines whether players consider an item valuable. Some items are highly elastic, meaning price greatly affects how many players are willing to buy, while others aren't.

- **Offers and promotions.** Special offers and promotions such as limited-time discounts may boost conversions, but don't be too aggressive. Frequent promotions can affect how players perceive value, possibly hurting your revenue stream.
- **Fraud.** As soon as you start offering items with real-world currency value, there will be fraud. There's no way to stop fraud completely, so make use of your platform-level fraud protection services and monitor it closely.

Pro tip:* *Use server-side receipt validation (e.g. Apple or Google receipts) or validation of third-party processing through your backend platform to services such as Xsolla, PayPal, or Steam for added security.

Conversion

A good strategy to convert free players to paying players not only increases revenue, but also boosts retention as players become more invested in the game. But there are two main challenges: eliminating the barriers to entry and showing your players value.

Every extra step in the process (entering payment info, email confirmations, and so on) is a **barrier to entry** that can drop 50% of players from the sales funnel.

Demonstrating clear value to players can make or break your strategy. Look to in-game data to understand when players are most engaged because that's often the best time to make an offer. Experiment with timing and offers until you find something that works but be aware it won't be one-size-fits-all. Segmentation can help you make offers that feel more valuable to players, as they're tailored to a particular playstyle.

"In general, players have more of a propensity to pay once they have a trust relationship with the game and the developer, and they really understand the value...I see a lot of developers that go down that path making mistakes of providing, say, starter packs that are way too valuable and negate the need for future purchases..."—**Dave Rohrl, Owner, Mobile Game Doctor**

Understanding how players spend

When thinking about value, keep your players' **spending potential** in mind. Players will have different levels of spending they are comfortable with. It's easy to get caught up focusing on big spenders or trying to sell as much as possible as soon as the game launches. But those methods are often unreliable, unsustainable, and may reflect poorly on your studio.

Pro tip:* *Build a broader, more reliable, and engaged spending base rather than chasing "whales".

Paying rate is just as important as ARPPU for measuring monetization. Depending on the genre and audience, games can wildly vary in the paying rate from single-digit rates to over 50%.

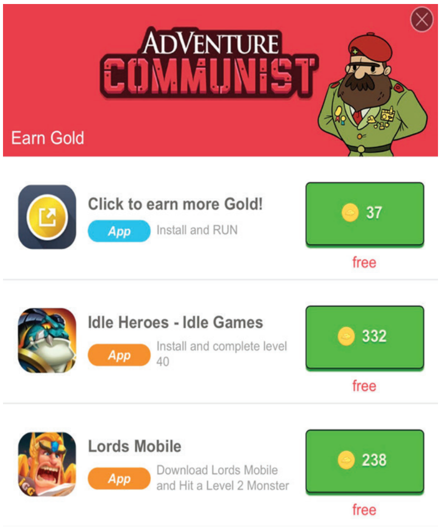
Home in on players that may never have high LTVs with low-cost options that fit their budget. A thousand players paying $10 is preferable to ten players paying $1000 because there is more opportunity for repeat purchases.

Advertising

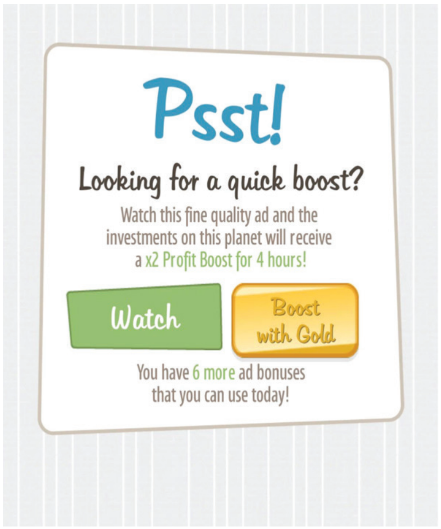
In-game advertising is an important revenue stream for many studios. One of the most popular forms is ***rewarded video***—short videos often promoting a different game or app, watched for an in-game reward or more playtime. Some studios are wary of rewarded video, especially if they rely on high-spending players for revenue, as players might be lured away by a competitor's game.

It doesn't have to be that way. Well-designed video ads that are smoothly integrated into gameplay often outperform other ads. If rewarded ads are part of your game, you'll want to monitor engagement with the ads, and tweak the associated rewards.

In addition to video, consider **banner ads** and **interstitial ads** (ads served during mid-game breaks). As with almost every other LiveOps effort, you need to continuously test different solutions. Many developers segment their audience and only show ads to certain segments, often limiting them to non-paying players.



Click to earn more Gold!	App — Install and RUN	37 free
Idle Heroes - Idle Games	App — Install and complete level 40	332 free
Lords Mobile	App — Download Lords Mobile and Hit a Level 2 Monster	238 free


Psst!

Looking for a quick boost?

Watch this fine quality ad and the investments on this planet will receive a x2 Profit Boost for 4 hours!

You have 6 more ad bonuses that you can use today!

Offering choices for in-game ads can help limit player dissatisfaction and churn.

Regardless of which path you take, you'll need to test your way into optimizing. Make sure you can play around with who sees your ads, and then experiment!

Targeting and Optimizing Ads

Once you've decided to run ads, the next step is figuring out how to optimize their impact. Joining an ad network is a simple way to boost revenue, but has limitations, particularly if you have more than one game to offer.

On the other hand, advertising a relevant in-game event or highlighting one of your other games may have a lower revenue floor than showing an ad from an external network, but the ceiling has the potential to be much higher.

For non-paying players, you're better off generating revenue through a paying ad. You can usually do an on-the-fly calculation to compare the value per impression of an in-house-ad versus one from an external network, so you can decide what to show for a given player segment.

Pro tip:* *Never stop testing your monetization efforts, because your players' perception of value (both real-world and in-game) will change over time.

In-Game Economies

In-game economies have proven to be engaging for players, and they scale well over long periods of time. They can also be hard to balance while maintaining player satisfaction.

Pro tip:* *Many games use two virtual currencies: a "soft" currency earned in-game, and a "hard" purchased currency. Using both gives more control over game balance because you can lock items to one or other currency.

Consider designing items or upgrades that can be purchased or achieved through progression. In-game trading can also help build a community that bridges paying and non-paying players. But be sure to exclude any items that could encourage paying players to prey on non-payers.

Player-to-player trading comes with the need to balance and prevent fraud. Use strict anonymity in trading to reduce fraud opportunities, or only allow commodity markets, where players can offer to buy or sell an item at a set price and the transaction completes only after a matching offer is made.

Balance can be difficult to test due to the time required by progression system gameplay. You can get around this by **modeling economy data**. Build a matrix of all the sources and sinks for in-game resources and build a model of the economic activity you can adjust in a tool such as Microsoft Excel, without rolling out updates.

Remember that sudden balance changes to in-game economies can leave players upset if, for example, someone stockpiled a highly valuable item that was later changed to be less valuable.

"You can artificially boost initial conversion by putting together a super good deal that tanks your economy. What we want is sustained investment and signs that a player has really perceived value in what they purchased"
—Emily Greer, Founder/CEO, Double Loop Games

LiveOps Monetization – Best Practices

- **Choose a monetization strategy** that fits your game and audience, and be willing to change it over time.
- **Set ethical and quality guidelines** for monetization so your team is aligned.
- **Take steps to prevent fraud** with server-side receipt validation, anonymity in trading, and rollback capabilities to undo in-game damage.
- **Focus on simplicity and variety** in your in-game store design. Players need to find items easily and expect different offers regularly.
- **Create bundles** to sell based on purchase behavior. If players often buy the same three items, bundle them in a time-limited discount for a revenue boost.
- **Run sales in tandem with live events** to hit players when their engagement is high.
- **Offer rewards for social sharing**, such as in-game coupons for sharing a post on Facebook.
- **Diversify the external ad networks** you use to maximize effectiveness across different regions.
- **Keep loss aversion in mind**, especially when it comes to balancing. If a player stockpiled a specific item only to have it nerfed, they'll naturally be upset. (Western players are especially loss averse.)
- **Always be testing** down to the last detail. A/B test price points, item appearances and descriptions, and even the color of the "Buy" button in your store.

Multiplayer

Few games launch nowadays without some form of multiplayer.

How to best configure and run multiplayer (from hosting multiplayer servers to running matchmaking) can vary depending on the gameplay modes you want to offer. Is the gameplay synchronous (real-time competition), or async (in turns)? Is it co-op or PvP (player vs player)? You can find detailed documentation on multiplayer architecture at **playfab.com/multiplayer**. This section focuses on multiplayer elements you can develop and roll out quickly for LiveOps.

Leaderboards

While it's not necessarily what people think of when we say "multiplayer," adding leaderboards can bring multiplayer elements even to single-player games.

"As soon as you add a leaderboard in a game, even if it's a single-player game, players start seeing progress against other people, and people all of a sudden start engaging more"
—Tammy Levy, VP of Insights and Analytics, Kongregate

No matter how players interact, leaderboards give them a way to gauge skills and see the community's skill level at large.

Filtering Leaderboards

Filtering leaderboards can make them fun for more players. Some of the most common ways to filter are by:

- **Geography:** Physical location, which can range from specific regions to entire continents (France, North America, EMEA, etc.)
- **Platform:** The device or software played on (Xbox, PlayStation, PC, iOS, etc.)
- **Mode:** Gameplay configurations that affect other game mechanics (PvP, PvE, campaign, etc.)
- **Option:** Specific options chosen by the player (control options, difficulty, language, etc.)
- **Level:** In-game level or map (Level 1-1, Lucky Lagoon, etc.)
- **Statistic:** Values generated based on player actions (wins, losses, K/D ratios, etc.)
- **Time:** For example, a leaderboard of today's best players.

Often the most fun leaderboards combine variables in interesting ways. For example, by combining the variables **Platform, Level,** and **Statistic** you could create a leaderboard for "Fastest time (Statistic) to complete *Ventura Highway* (Level) by PC players (Platform)." That sounds a lot more interesting than a leaderboard for "Best Time."

Other Leaderboard Techniques

Keep in mind, **leaderboards don't have to be purely competitive**. Tracking the biggest contributor to a team victory can make the social aspect of a multiplayer game more rewarding and add replay-ability.

Resetting a leaderboard can give your player base a great reason to dive back in immediately and aim for the top. If you can incentivize the climb with some reward, you may even reactivate some lapsed players.

There are also a few methods **to prevent cheating** on leaderboards. You can encrypt client-cloud communications to prevent packet sniffing or use server-side telemetry to investigate suspicious scores.

Pro tip:* *Use prize tables to set actions to trigger when a player or group reaches a new rank in the leaderboard during a reset cycle.

Player Groups

Groups such as guilds (or even temporary parties) can get players more invested in a game by making it more social and creating shared goals.

Like leaderboards, player groups can be integrated even in single-player games, especially if players can communicate and compare game progress or stats.

Pro tip:* *Determine how short-term groups are formed based on how much players need to trust teammates to succeed. Strategic, role-based team games should let players self-select into groups, while straightforward raid battles can have randomly generated groups.

Long-term groups (such as guilds) have been proven to increase player retention, especially when players can customize aspects of the guild. Letting players control cosmetic features, assign member ranks, or use multiple ways to communicate (such as internal chat or group-wide push notifications) can make them more invested in the group's success.

Guilds can even boost monetization, say if some wealthier members of a guild feel generous enough to gift items to teammates.

"One of the things that I look at is how do I provide you the best experience not only within your guild, but when your guild is gone at some point. Guilds fade away. So, when your guild which used to be ten people now has five people, what happens? How do I match you with the right people? It comes down to matchmaking, but it's making sure that the right type of people are filling the right type of guilds and they have the right aspiration together as a group."—**Scott Koenigsberg, Head of Product, Zynga**

Managing Communities

Gaming communities can be some of the friendliest and most inviting, and they can also be unfriendly.

Limiting communication options to pre-selected options, using filters to prevent certain words or phrases from appearing in chat, and creating a code of conduct that players must agree to are some ways to curb bad behavior.

Some developers even design gameplay mechanics to foster positive interaction. The team behind *Guild Wars 2* reportedly built the whole game around the idea that "players should always be happy to see one another." The community around the game is so inviting that it's often called the "noob-friendliest MMO," which is probably why it has one of the biggest active player bases in the genre.

> *"Behavior gets the most toxic when no expectations are set of any sort and people come in with a mentality of 'anything goes.' The more you can provide a framework for people to operate in, the more likely they are – there's peer pressure to do it, there's social standing reasons."*
> **—Raph Koster, CEO, Playable Worlds**

Pro tip:* *A dedicated community manager can help keep players satisfied and foster a positive community by being the liaison between players and the dev team and keeping that team aware of the latest trends and player sentiment.

Localization

While not strictly a multiplayer feature, localization can be a huge part of facilitating interaction.

You never know where your game will find its audience. Demand can appear out of nowhere in a specific region of the world, and meeting that demand quickly means you maximize revenue.

Localization can also mean the difference between a free and paying player. There are estimates that up to 50% or more of online users will only buy when presented offers in their native language.

Pro tip:* *Give your localization team direct access to your string table, so they can edit the text directly and see the results in the game right away.

Localizing a live game is especially challenging because the content changes constantly. It's one thing when you only need to localize text in the game UI and tutorial. You can do that once, using an outsourced vendor, and then never worry about it again.

But when you're running continuous A/B tests (as you should be in LiveOps) or launching new events every week, you'll have a continuous stream of content that needs localization. Store as much of the in-game text on the server as possible, so it can be easily edited and localized.

Localization also goes beyond language: time and date formats, fonts, currency formats, and images with text should all be tailored to the player's location, preferably at the code level.

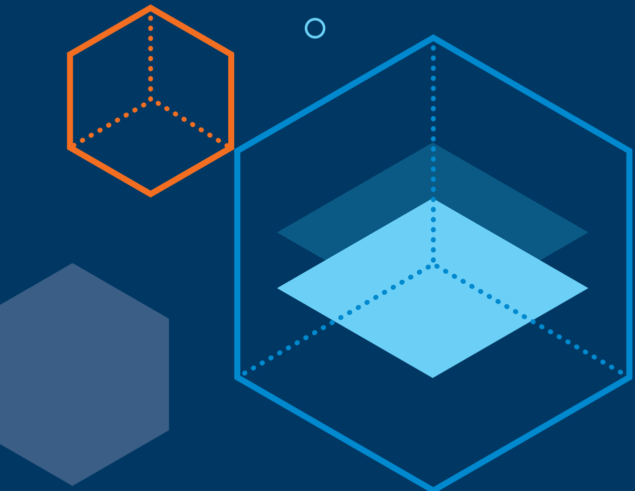
Depending on the region, you may need to change parts of the game to fit cultural norms or avoid controversy. The use of alcohol, certain foods, symbols, hand gestures or displays of affection are all things that may offend players. The earlier you plan for localization, the better your game will perform, and the happier your players will be.

LiveOps Multiplayer – Best Practices

- **Consider every possible way** you can add multiplayer elements to your game. For single-player games, leaderboards, player groups, and even some asynchronous competition can be added after launch.
- Early in development, **decide on multiplayer modes** that fit the core gameplay and expected player behavior.
- **Experiment with your matchmaking algorithm**. Often the simplest mechanism is the best.
- **Prioritize the most important variables** in matchmaking. Some games prioritize player region or language, while others don't need to.
- If you host your own servers, plan how you'll **scale the multiplayer servers** as the player population grows.
- **Offer multiple ways for players to communicate.** Give players accessible options to text chat, voice chat, start group messages, or integrate with an external messaging service such as Discord.
- **Offer ways for players to customize their groups.** Let them edit names, appearances, member ranks, group size, whether players must be invited to join, and so on.
- **Reset leaderboards** on a regular basis to give players a sense that there's always a new opportunity to win. Use prizes to incentivize players to come back after resets.
- **Build a "refresh" button into your game,** that forces the game to download and display the latest localized text. This lets your localization team tweak text in-game and see the effects immediately.
- **Make sure all player communications can be localized** (not just in-game text, but also push notifications, emails, and so on).

Part 3: Tools and services for LiveOps

"Games can last for decades as a service if you know how to operate them properly and invest in scalable solutions. Some great scalable solutions are automation and tools. Try to automate all the things that don't matter and build tools for all the things that do. They not only reduce costs and improve efficiency, but they reduce pressure and allow the team to concentrate on the most important parts of the service." **—Ken Go, Founder, DECA Games**



To run LiveOps effectively you need the right tools and services. Visit **playfab.com** for more information.

PlayFab Feature Overview

Multiplayer Services

Cross-Network Identity and Data

- **Player Authentication:** Start with frictionless authentication and let players link accounts to roam across Windows, Xbox, Steam, PSN, Nintendo, Facebook, iOS, Android and more.
- **Player Data Management:** Share player information and game state across devices.
- **Matchmaking:** Use the proven capabilities of Xbox Live's SmartMatch on any platform to help players find opponents.

Multiplayer Servers

- **Dedicated Servers:** Deliver low-latency real-time gameplay for any platform.
- **24/7 Monitoring and DDoS Protection:** Protect against DDoS attacks and other incidents.
- **Global Reach with Microsoft Azure:** Get closer to players on a cloud with more global regions than any other provider.
- **Control Costs:** Dynamically scale server cores in response to demand.

Chat

- **Party:** Connect players with low-latency peer-to-peer communication.
- **Text and Voice Chat:** Facilitate accessible voice chat, transcription and translation.
- **Accessibility:** Use automated speech-to-text transcription and voice synthesis to make communication simple for everyone.
- **Real-Time Translation:** Break down global barriers and grow player concurrency by translating voice and text chat between more than 30 different languages.
- **Encryption:** Ensure secure player communications.

Leaderboards and Statistics

- **Tournaments and Leaderboards:** Facilitate permanent or time-limited competitions amongst friends or strangers.
- **Scheduling:** Reset leaderboards on a schedule and archive standings so players can view past results.
- **Prizes:** Reward players based on their actions and leaderboard ranks.
- **Cheat Prevention:** Defend against unwanted behavior and remove fraudulent players and accounts.

LiveOps

Engagement and Retention

- **Game Manager Web Portal:** A shared space (with roles and access permissions) where studio members can build, configure and operate your game.
- **Daily Reports:** Evaluate your game's performance through the lens of the top metrics used across the industry, pre-calculated for you daily.
- **Player Profiles:** Track players across authentication services and platforms.
- **Customization:** Use server-hosted player data and logic to build custom game mechanics.
- **Achievement Systems:** Use the rule engine and custom player events to build a powerful achievements system.
- **Real-Time Segmentation:** Act immediately on targeted groups of players.
- **Player Communication:** Talk to your community with push notifications, emails, and message-of-the-day pop-ups.

Content Management

- **Title Data:** Manage your game configuration remotely.
- **Item Catalog:** Configure your catalog of items available for in-app granting or purchase, and update at any time.
- **Content Delivery Network:** Upload, host and deliver game assets via Game Manager.

A/B Testing

- **Player Buckets:** Run experiments with randomly assigned groups of players.

Monetization

- **Stores and Sales:** Target player segments with personalized store offers and support payments with Xbox, Steam, Google, PayPal and more.
- **Virtual Economy:** Mint promotional coupons and virtual currencies with support for setting initial balances and optional auto-recharge.
- **User-Generated Content:** Empower players to create, upload and search for moderated content.
- **Drop Tables:** Craft attractive bundles for first-time or regular users and stimulate demand with item scarcity.
- **Fraud Prevention:** Use server-side receipt validation to make sure purchases are genuine before completion.

Automation

- **CloudScript:** Build lightweight logic processing when you want server authority without a dedicated server.
- **Task Scheduling:** Set up pre-defined actions to manage anything from prices and events to messaging lapsed players.

Data and Analytics

Real-Time Analytics

- **PlayStream:** Monitor a unified real-time stream (and historical archive) of every event fired by your game.
- **Real-Time Rules Engine:** Set up powerful actions and triggers that respond to PlayStream events.
- **Real-Time Segmentation:** Use data properties to bucket players into segments and trigger actions as they enter or exit.
- **Event Filter and Search:** Zoom in on a time slice to analyze players, events and error conditions in detail.
- **Reports:** Review summaries of your game's daily and monthly performance with automatic reports available by daily email and on-demand.

Data Management:

- **Insights:** Gather all event and processed data into a single title database in the cloud.
- **Event Archiving:** Schedule event exports to a pre-existing Azure Blob Storage or Amazon S3 data warehouse.

Compliance:

- **GDPR:** PlayFab is committed to being General Data Protection Regulation (GDPR) compliant and as your service provider, ensuring that we provide you with the hooks you need to allow players to view or delete their data.
- **COPPA:** PlayFab is in use today by COPPA compliant games.

SDKs

ActionScript

Android Studio

C++

C#

Cocos2D

Java

JavaScript

Lua

NodeJS

Objective-C (iOS)

Phaser.io (JavaScript)

PHP

Python

Unity

Unreal Engine (Blueprints & C++)

Xamarin

Support

Documentation: Get started quickly with tutorials, samples, and comprehensive reference documentation.

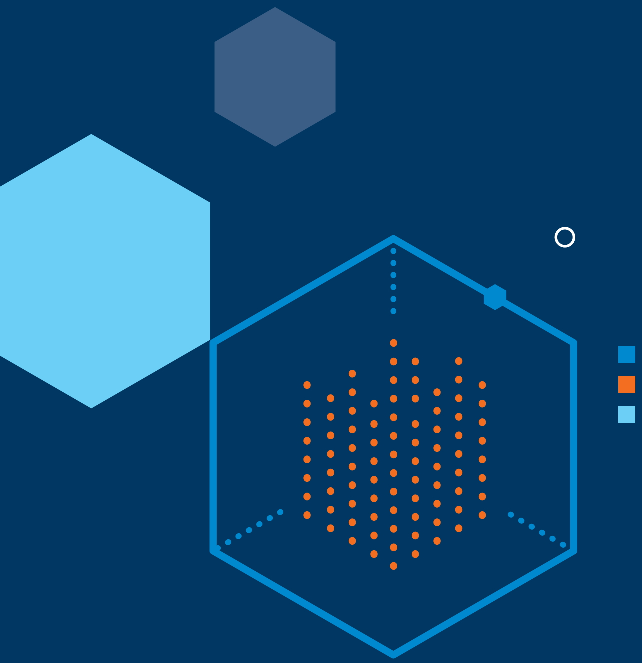
Forums: Learn from experts and share your knowledge in the community forums.

Slack: Join channels for ongoing conversations with other developers and direct communication with PlayFab Developer Success.

Real-Time Service Health: Visit status.playfab.com for current and historical service health information.

Tickets: Get enterprise-level ticketed support.

24/7 Emergency Escalations: Get around-the-clock assistance on immediate-priority issues.



Visit **playfab.com** for more information.