



# Enginesis Platform Overview

Version 2.1.4 – October 16, 2017

Online games have grown in popularity with the ever expanding reach of the World Wide Web. Being a leader in this area has set forth many efforts to supply content and infrastructure to meet this growing demand. This document describes the features of a service-oriented architecture (SOA) platform providing comprehensive services for online gaming called Enginesis. Enginesis delivers a feature rich foundation for online game content for selected developers to supply games that include our community features and online game services.

Online games are about fun and challenging experiences, personalization, competition, and social interaction. The Enginesis platform helps make these goals easy to deliver. The benefit is the platform makes available a comprehensive set of services any game developer can tap into such that they focus their efforts on creating great games that take advantage of the platform services. This approach lowers the barrier of entry to making feature rich games while lowering the cost of game development and decreasing time to market. The design and architecture of the Enginesis platform uses proven open source technologies so it is cost effective to operate and maintain.

## What Is Enginesis?

Enginesis is a comprehensive web-services based application server optimized for online games. Enginesis supplies the common community services most online games require, such as player registration and profiles or co-registration, score submit, leader boards, awards, player reputation, multiplayer, ad delivery, tracking services and much more. Enginesis is a platform that provides its services to any number of web sites at the same time where each site can pick and choose the services it requires. Each web site maintains its own view of data as there is no sharing of data between sites or registration repositories.

Enginesis is a service – there is no user interface or GUI or anything to display at all. Web sites, games and other user targeted applications use the Enginesis API to build user interfaces utilizing the services. It provides all of the low-level services one would use to build upon it a robust community-oriented online gaming web site, game or application. A particular web site can selectively choose to expose any subset or all of the Enginesis features to its users. Centralizing the functionality provides immediate return on investment such as:

- Best of breed: Comprehensive set of features and services to rival any current competitive platform.
- Simple: the simple to use gaming platform opens the opportunity for any level of game developer to easily participate in providing services and content embedded within the online community.
- Open: The services are available using popular and proven technologies and the open implementation enables anyone from anywhere to use it.





## Enginesis Platform Overview

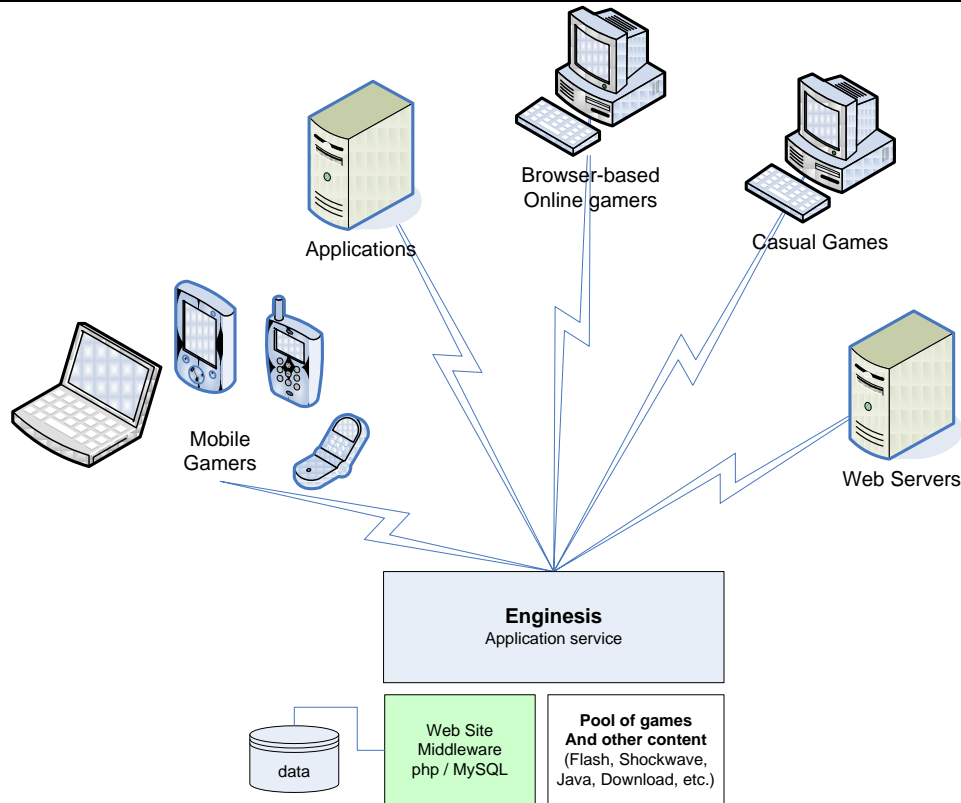
---

- Service oriented approach: cleanly separate the functionality (back-end) from the presentation (front-end). This leaves the application (games, web sites) to expose the features they want in a presentation consistent with their own application design.
- Economy of production: one small development and administration team focused on building and maintaining the services, instead of utilizing each web site's limited resources.
- Focused internal experts: dedicated team members are platform experts, skilled in this problem domain.
- Flexible hosting: the platform can be hosted centrally or at a designated hosting facility. The platform uses mostly open-source components. Central hosting offers economy of scale providing the services, deployment and scalability. However, an application can choose to supply their own hosting to possibly reduce costs or maintain control over their product.
- Comprehensive gaming services: game syndication, user profiles, multiplayer services, chat, leader boards, ranking, digital rewards, gifting, affinity points, contests are just a few of the growing features. Each web site does not have to develop, license, deploy and maintain these services.
- Community: aggregating the user community such that we enjoy the benefit of a network effect and participation in gaming events (multiplayer, tournaments, challenges, etc.) is open to a broad and diverse audience.
- Sponsorship: allow games to offer sponsorship, ad delivery, contextual advertising and impression metrics.
- Localization: today's market demands global reach. Almost all of the Enginesis API's are capable of delivering results including translations into local languages.
- Growth: as new services are identified or demanded by the market, they are rolled out centrally. New games to legacy games immediately participate in the new features.

Consumers of the Enginesis platform are connected, online applications. An application using Enginesis would be another web site (employing HTTP services with Java, Ajax and others technologies), a browser based game client (using Flash, Shockwave, Unity, Java, ActiveX), a PC or Mac game, mobile platforms such as iPhone, Android, Windows Mobile or Blackberry, or a console game with an Internet connection. Basically, anything capable of initiating a HTTP transaction and receiving XML or JSON in response may use Enginesis.



## Enginesis Platform Overview



**Figure 1: Enginesis Platform Access Diagram**

The “business rules” of community services and online gaming is coded within the Enginesis Application Service layer such that games and web site page coding does not concern itself with the details, such as player rankings, ordering leader boards, badge assignments, tournament rules and so forth.

## Why Enginesis?

Why invest in centralized software development of what, on the surface, appears to be a relatively simple problem? Many individual games and even some web sites would not require all the features provided by Enginesis. Centralizing online game community features and delivering them as a service affords businesses both large and small many advantages, such as:

- Individual games and/or web sites can pick and choose the features they require from a large and expanding platform;
- Content syndication: content can be added to the platform and shared by subscribing sites (where it makes sense);
- Control: the application (game and/or web site) determines the games, content and services they require;
- Audience aggregation: social media sites flourish on large and diverse populations of users. Small sites gain the advantage of increased users from larger sites; large sites gain the advantage of a more diverse audience.
- Pooling of development and support resources: limited redundancy of one or more sites developing similar features or learning from the same mistakes;



## Enginesis Platform Overview

---

- Platform scalability;
- Feature development, all servicing sites gain the knowledge and experience of the more advanced sites;
- Open API allows both internal and external developers to develop new content independent of the core platform team;
- Consistency: a similar set of features deployed across all gaming and social media sites affords the audience familiarity and comfort while interacting with any Enginesis enabled site.
- Client agnostic: any front-end client can utilize the services: a web browser using Ajax, a web site using HTTP, Flash, Shockwave, C++, Java and so on. Any technology that can communicate using the HTTP protocol and process XML or JSON can use Enginesis.

## Enginesis Architecture

The Enginesis architecture is a three-tiered application stack consisting of:

1. Client-side SDK, where the client could be Flash (AS2 or AS3), Unity 3D, Shockwave, PHP (HTML based games, web sites, Social Networking sites), Java or a mobile platform such as iPhone or Android;
2. The Enginesis platform layer, which provides session management, security and data wrapping, implemented in PHP;
3. The data layer, providing all persistence and most business logic, implemented in MySQL.

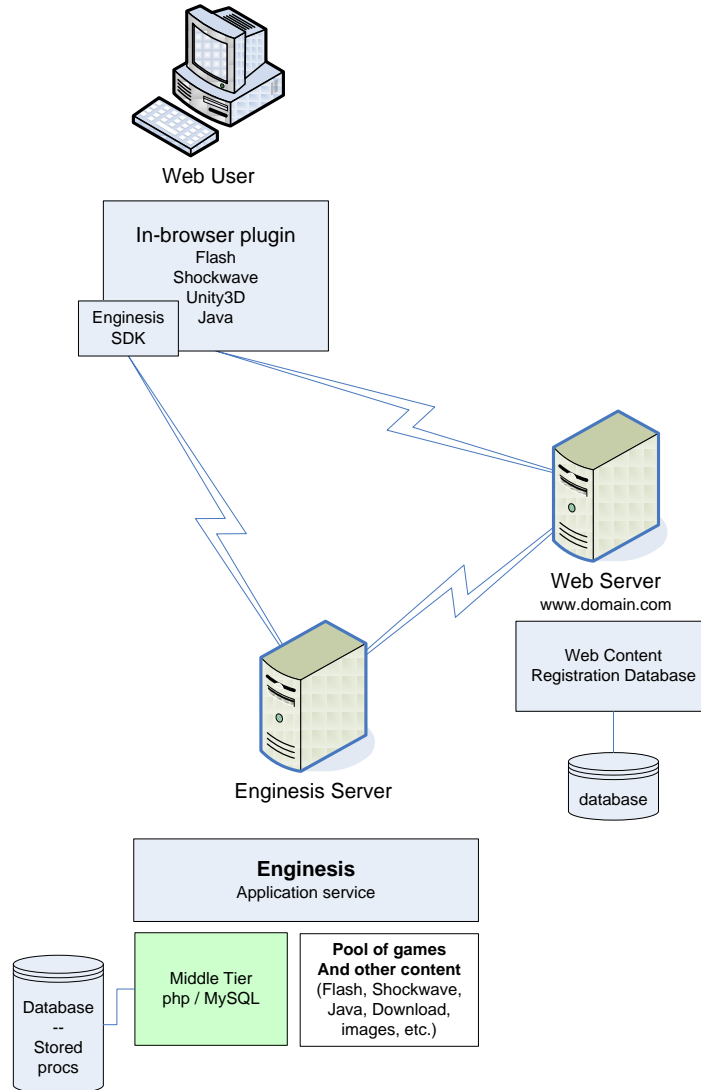
Refer to Figure 1: Enginesis Platform Access Diagram and Figure 2: Enginesis Services Relationship with a Hosting Web Site and see the section on Scalability.

Enginesis has a complete user registration and profile service but typically Enginesis is going to act as a symbiotic extension to an existing web site. Most web sites already have a complete registration system but they lack the services specific to games. Enginesis can extend the profile service of the hosting web site and provide its capabilities: leader boards, badges, experience points, and so on. Enginesis provides this extension through a co-registration service (refer to the section Co-registration on page 13.)

To perform co-registration, Enginesis must understand how the hosting site identifies authenticated users and requires access to the current logged-in user's display name and identifying key such as user id. For example, on Facebook we use the Facebook Connect API to perform user recognition, then use the Facebook user identifier to match the logged in user to a user in the Enginesis database.



## Enginesis Platform Overview



**Figure 2: Enginesis Services Relationship with a Hosting Web Site**

## Open API

The Enginesis platform is open to all application developers and authorized 3<sup>rd</sup> party web sites. We provide API support through a Flash SDK for Flash clients, PHP interface files for web sites using PHP, documentation and sandbox environment where new concepts can be tried and tested. A Java (for J2EE site development), Unity 3D and Shockwave SDK are on the roadmap.

Because Enginesis is built as a web service, just about any application capable of issuing HTTP POST operations and parsing XML or JSON response packets is a potential Enginesis client. Enginesis itself implements no user interface or direct end user services. This methodology keeps things simple so that any moderate programmer can easily integrate their required services and format the result to their design.

### *Security*

Everyone is familiar with the woes of keeping things secure on the Internet. Security is a particular issue with Enginesis due to the competitive nature of games and the natural tendency for hackers to attack the games and the web sites they live on. An even broader security issue is the lack of trust in game clients themselves, since most games are developed by third party external developers who may not be concerned enough or qualified to deal with proper security.

Enginesis operates under a strict three-tiered security model.

- **Public services** – functions available to anyone on the public Internet. An example public service is viewing a particular leader board.
- **Authenticated services** – functions that are called by users, who are logged in and properly authorized to communicate with Enginesis servers. An example of an authenticated service is score submit.
- **Private services** – functions that are only called by authorized internal administrators or services running on their behalf on a private network (i.e. Intranet or VPN). An example of a private service is adding a new game to the database.

We employ a variety of measures to address security, including CAPTCHA, SHA1 hash, encrypted cookies and encrypted transmission payloads (e.g. Blowfish encryption of POST data) where appropriate. Through the use of our client SDK most security measures are modular yet transparent to the game programmer.

### *Scalability*

The current implementation of Enginesis uses very simple but standard web technology: Linux, Apache, MySQL and PHP (LAMP). However, very little of what we have done is strictly dependent on this application stack. We believe nothing we have done is dependent on the operating system or the web server.

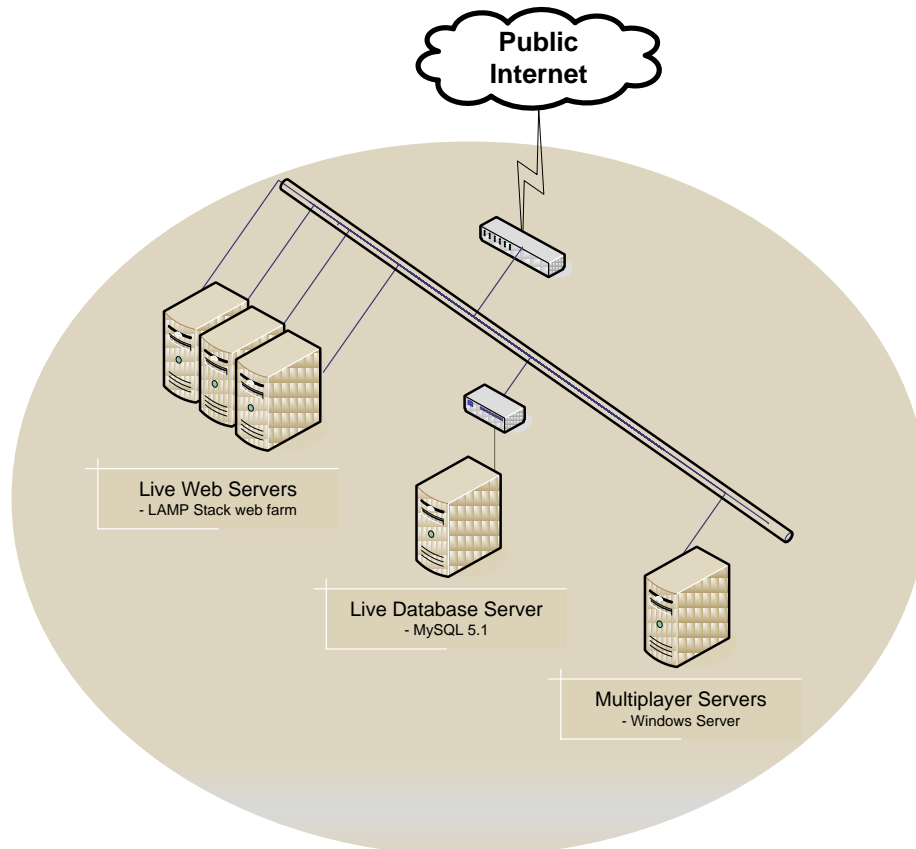
Most of the core functionality and the business rules of the platform are coded in SQL Stored Procedures and run in the database engine. There is little that is inherently dependent on MySQL. We could migrate the code to any robust SQL compatible database (such as Microsoft SQL Server or Oracle).

All our interface code, authentication and supporting services are coded in PHP. This layer of functionality is dependent on a PHP module running on a web server, yet it is not OS or web server dependent (PHP runs on almost any platform). However, since we are dependent on PHP to take advantage of another technology (ASP or Java) would require major code rewriting and possible re-architecting parts of Enginesis.

This architecture allows Enginesis to run on as little as a single low-end computer servicing a very modest audience (a few hundred users) and scale up from there to as large a web

## Enginesis Platform Overview

farm as required to service the target audience. The following diagram shows the current Enginesis deployment capable of serving hundreds of thousands of registered users and a fairly large concurrent audience of tens of thousands of simultaneous users.



**Figure 3: Enginesis Production Infrastructure**

We also are able to deploy Akamai Edge Suite delivery for many of our assets and certain queries that are not immediately time sensitive. By integrating Akamai technology into Enginesis we have some dynamic control over how hard our web servers and database are hit during peak load times.

## Database Partitioning

An application design such as this usually leads to a bottleneck at the data layer where all processes block trying to read or write to the same place. The Enginesis design addresses this issue with several partitioning schemes.

The primary key to most tables is the web site id, allowing data partitioning based on web site registration repository. Multiple sites can be stored in the same database using this key. When required, the entire data set is partitioned to separate database servers based on this key as there is no cross-pollination of data between registration repositories.

# Enginesis Services

This section describes all the features of Enginesis. Categorically these services are:

- Player Profile Services – services related to an individual registered user;
- Gaming Services – services related to a game or group of games;
- Multiplayer Services – services supporting multiple registered users playing the same game concurrently, along with chat;
- Notifications services, such as email, newsfeed and alerts;
- Tracking & metrics;
- Ecommerce;
- Localization;
- Search;
- Social Services;
- In-game ad delivery;
- Quiz Widget.

The following sections will outline the features and services of each area.

## Player Profile Services



A player is an entity that encapsulates the data attributes and services associated with a registered user of Enginesis. Most sites call this a user, but since this is a gaming service and typically we call participants in games *players*, we adapt that term here. The terms player and user are interchangeable throughout this discussion.

The profile is a container of all the attributes for any individual registered player. Such attributes include a screen name, password, birth date, location, etc. A player in our Enginesis database identifies a unique individual and forms the relationship of the player with any participating data objects such as leader boards, tournaments, reviews, comments, and teams, to name a few.



## Enginesis Platform Overview



Figure 4: Profile With Avatar, Tag Line, Level Rating and Other Attributes

### Profile Attributes

Attributes are the data items associated with a specific profile. A summary of profile attributes:

- **User Name:** A pseudo-name or screen name the player selects to represent him or herself to other players. The User Name must be unique.

A player's user name is their unique moniker that appears in any GUI element to represent the player (leader boards, lobbies, chat, profile, etc.) This helps to protect the player's true identity and provides a fun way to represent an alter-ego throughout the gaming web sites. A screen name must be unique across any single site (in case the database represents multiple web sites).

When co-registration is used the user name comes from the hosting web site.


- **Real name:** The user's real name. This is captured in the profile when site authorized communication occurs between the site administration and the user. This is never revealed to other users.
- **Location:** Location information includes city, state, zip code and country code. Enginesis does not actually validate any of this information: it is expected the hosting front-end would enforce its business rules on location information.
- **Date of Birth:** we capture this to enforce COPPA compliance, demographic tracking and to represent social context in profiles and gaming services.
- **Gender:** used for avatar representation, demographic tracking and social context.
- **Tag Line:** Each player may enter a sentence or two to make a statement that is viewable by other players.
- **User Info:** A free-form text field user's may enter narrative data about themselves or things that are important to them to be used in a social context (e.g. trivia about

## Enginesis Platform Overview

- me; my most important things, etc.) or a site may choose to structure this data for specific responses (as World of Pop did with 20 specific pop trivia responses associated with each profile).
- Security features such as password and user security question (e.g. What was your high school mascot?).
  - Opt-ins: we capture up to 31 individual opt-ins per user. For example, the user may opt-in to a site newsletter and separately opt-in to receive information from a particular site sponsor. Each opt-in is interpreted according to the business rules of the hosting site. Enginesis only manages the data capture and provides services to process opt-ins.
  - Avatar: a registered user customizes a site-wide avatar to represent their image on site, in leader boards, and within games that support it.
  - Site-wide experience points: as players play games and submit scores or win/lose in multiplayer games the business rules of the hosting site determine how many experience points that play earned the player. Experience points are accumulated on a site-wide basis to determine most active players and distinguish really good players from mediocre players. (NOTE we also support tracking experience points on an individual game and group of games basis. These accumulators are discussed in the Gaming Services section).
  - Game Play History: Enginesis tracks a certain amount of prior game plays on behalf of each registered user for each game they play.
  - Best Score: Enginesis tracks the player's best score for each game played on a given site. Refer to the section on Gaming Services for more information about scores.
  - Profile usage statistics: various statistics are tracked such as date profile was created (so a site can show "member since 1999"), date/time of last login, number of logins, date profile was last modified.
  - Buddy List: Users maintain their own list of other players on a given site they regularly associate with. Mainly used for chat, but also for multiplayer gaming, challenges and teams.



Welcome DarkMatters!



[Edit My Avatar](#)

13 Profile Views  
Last Login: 11/17/07 10:05 pm

---

Game Play History

Too Many Questions

Best Score: 13,221

Rank: 9,160

Date	Score
09/28/07 1:28 pm	503
09/28/07 1:28 pm	502
09/28/07 1:26 pm	501
09/28/07 1:25 pm	500
07/13/07 10:37 am	10,809
07/09/07 5:11 pm	3,537
07/07/07 1:41 pm	13,221
07/03/07 7:00 pm	5,807
06/12/07 12:39 pm	6,474

## Enginesis Platform Overview



**Figure 5: Profile With Buddy Editor**

- **Favorite Games:** Each player can maintain a list of their favorite games. Usually used as a short cut.
- **Comments:** Other players can comment on players. This forms a community arbitration system.
- **Gifts:** A collection of gifts the player earned or was gifted.

## Enginesis Platform Overview



**Figure 6: Profile With Comments**

- **Badge & Achievements:** A player can earn badges in certain games for superior game play or achieving certain predefined milestones. Once earned the badge is never taken away from that player.
- **Game Rating:** a player earns a rating when playing individual multiplayer games, since a rating is dependent on the player's performance in that specific game (it's not a site-wide value). This is described in more detail in the multiplayer section.
- **Co-registration site key:** for sites requiring co-registration we track the user's key on the other site such that the profile can be linked.
- **Personalized game content:** Certain games support personalized content (level builders, custom tracks, etc.). For examples of this feature see Deer Stacker on Comedy Central or White Rapper Boom Box on VH1.
- **Currency (affinity points):** Registered users accumulate currency as they interact with certain features that reward specific interaction. Currency differs from experience points in that they can be redeemed or deducted from the players account. Currency is a form of e-commerce where players can redeem their currency to purchase items (in-game, digital rewards, etc.). Currency is tracked per user on a site-wide basis and by individual game when set up. Currency is discussed further in the e-commerce section.
- **Notifications:** Information alerts sent to users indicating certain important events that have occurred involving their profile. Such events include: buddy add/remove, another user commented, forum reply, leader board change, challenge event, team update, etc.
- **Message boards and Forums:** users may post comments in forums.



## Enginesis Platform Overview

---

- **Team Memberships:** registered users may have team affiliations and a role within the team (captain, player, substitute.) Games, challenges and tournaments will then be extended to recognize team competition.
- **Groups:** a network of users who maintain similar interests. For example, the Trivia Lovers group.
- **Mobile device:** player's mobile device address for alerts, presence indication, challenges.

### Profile Services

Services are the functions provided by the system that act upon profile attributes and compound other services, such as security. Enginesis services specific to profiles are:

- Register new account;
- Update existing registration;
- CAPTCHA service;
- Login;
- Forgot password;
- Get User Status – returns various attributes such as last login time, number of logins, date account was created, etc.
- Get Users Online;
- Logout;
- Co-registration with other external web sites or profile management systems;
- Avatar rendering service (for example, see [http://www.enginesis.com/avatar/index.php?site\\_id=100&user\\_id=1&size=1](http://www.enginesis.com/avatar/index.php?site_id=100&user_id=1&size=1))
- Various access, set and listing methods for all of the previously discussed attributes.

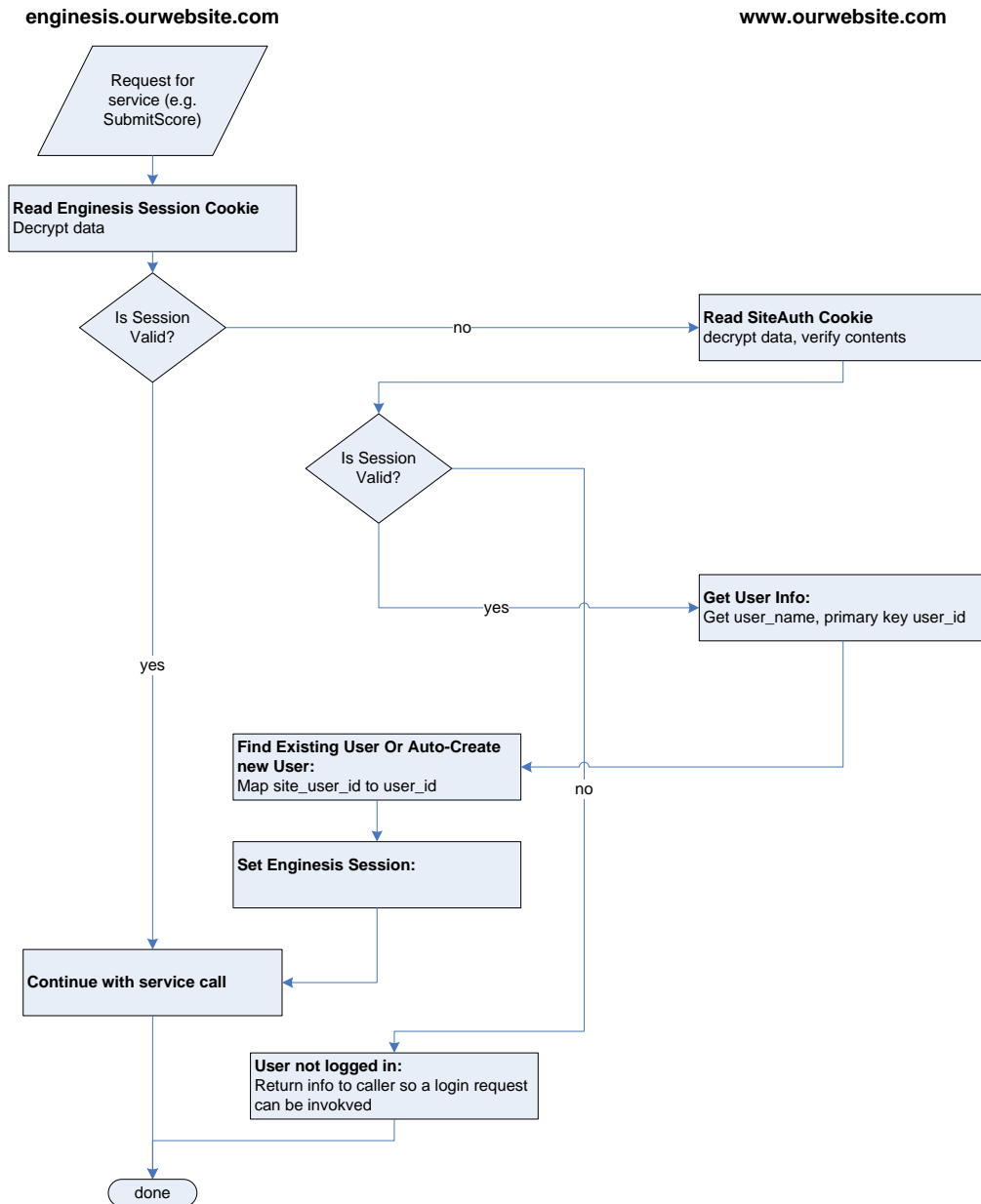
### Co-registration

Depending on initiatives, resources and timing we partner with external 3<sup>rd</sup> party sites for content and services. We always want the user experience to appear seamless such that the user believes it is one cohesive site. It may be necessary for user profile data to exist in more than one place, or parts of the profile exist in an auxiliary database. We developed a co-registration service that allows Enginesis to know where to look if profile data exists outside its realm. Co-registration presents two data problems:

1. User registration data originates on a 3<sup>rd</sup> party site and should be linked to the Enginesis database. This methodology has been implemented on MTV.com, VH1.com, ComedyCentral.com (using both UREG and Flux registration), Facebook, Yeti Games and multiplayer GameTrust games.
2. User registration originates in the Enginesis database and linked to a 3<sup>rd</sup> party web site. We use this method with Cheetah Mail, certain Electrotank and Facebook games.

A user can register on a hosting web site and Enginesis can access the authentication information and automatically match that user to a profile in the Enginesis database. Enginesis holds and associates the hosting site's user key such that profile data and user game data are linked.





**Figure 7: Enginesis Co-registration Model**

## Other Profile Concepts

Enginesis is a robust platform containing many features. This section describes features and concepts that may require additional information to understand the platform.

### *Teams*

A team in the online gaming sense is a collection of players whose collective game play results are considered together (as if a single player) for scoring, competition and ranking purposes. The team can have attributes such as a name, a mascot and a

## Enginesis Platform Overview

bio. Teams are usually fixed in size but that number depends on the application the team participates in. The team will act as a single entity under certain gaming models such as in a tournament teams can compete against each other and look like a single player. The team terminology may also include guilds, clans and possibly others depending on the profile of the web site audience.

### Groups

A group is an organization of players with a common interest. Organizing players into groups is a common social media construct. Group profile attributes include the group name, description, logo and image. Groups also have a membership (a collection of existing users) and owner/moderators (users who can edit the group attributes and membership.)

### Avatars

The avatar system we developed is very flexible and not dependent on any particular web site. Avatars can be included inside games and on web pages and in stand-alone applications. The avatars themselves are divided into parts, or layers, and individual sites have a small degree of flexibility determining which layers they use and complete flexibility determining the assets that make up the individual parts. For example, there are different head shapes, hairstyles, eyes, nose and mouth, as well as stylish accessories such as hats and body art.



Figure 8: Comedy Central Avatar Builder

Enginesis provides open services to render avatars as .png formatted files in a variety of different sizes. This way any application has access to the avatar image.

## Enginesis Platform Overview

---

### Game Services



Game services encapsulate the functionality related to a particular game or group of games residing on the Enginesis servers.

A game is a container of the attributes associated with a single game. Game attributes are name, description, thumbnail, implementation technology (such that the code knows how to generate the <object> tag and communicate configuration information to the game), links, and so on. Games may also have other rendering flags such as active/inactive (do not show inactive games in any game list), public/private (private games available only to logged in users), time sensitive begin date/end date (game only available for a promotion period) and premium/non-premium content.

The top-line features of games are:

- Game Delivery – Flash, Shockwave, Java, Unity, ActiveX, Download (.exe, .osx). Games can be delivered directly, through iframe, or embedded using object/embed tags.
- Leader Boards supporting all-time, monthly, weekly, daily and arbitrary time period aggregation.
- Game groups – two or more games considered together, such as a game suite, where they share a player rank leader board, awards and experience points.
- Site-wide, game group, contest, tournament and single game player ranking.
- Game source, assets, attributes for programmatic (CMS) staging
- Game syndication: feeds to communicate games, game lists, game recommendations, leader boards to 1<sup>st</sup> and 3<sup>rd</sup> party affiliates.
- Game Organization: games organized by genre, popularity, franchise, related games (links, associated shows links to web pages, associated download games) or arbitrary producer-controlled ordered lists.
- Developer attribution.
- Featured games. Site producers can designate specific games as featured.
- Recommended games. Site producers can designate a relationship between games such that when a user expresses interest in a given game a list of other games related to that game are shown.
- User favorite games: users may maintain their own sorted list of favorite games.
- Game ratings (by user votes).
- Game reviews (by users).
- Game event tracking.
- Profile status and avatar integration.
- Awards such as badges, achievements, trophies and temporal awards.
- Challenges where users can pick a game and challenge their friends and maintain a private leader board for a given period of time.
- Gifting. Site producers or game developers can designate gifts that are available for specific games. Gifts can be earned by level, affinity points or reward.
- In-game ad delivery.
- User generated content (UGC), galleries, usage statistics and user ratings.



## Enginesis Platform Overview

---

- Player augmented games (user generated content) – users can design their own characters, enemies, levels, environments, etc. such that either they can restore the game or other players can play their creation.
- Multiplayer services integrating friends, chat, player ratings and other profile services in multiplayer games.

### *Features in Development:*

- Contests: open competition for a fixed period of time tied to prize or digital reward.
- Tournaments: The current tournament system can be expanded to include more complex features.
- Dynamic in-game ad sponsorship: Unlike pre-roll, post-roll or interstitial ads, dynamic in-game ads are deeply integrated into the game. An example would be a Toyota sponsorship in a game featuring cars would swap out a generic car for a Toyota model. Since this is direct interaction with the user a higher value can be placed on the ad impression.

### *Score Submit*

A score is a result of a complete game played by a user at a particular date and time. Given the configuration of the game on the Enginesis servers, the score submission may trigger other events such as experience points, badge awards, challenge reconciliation and various notifications.

Games and their representative leader boards may also have multiple time periods. Therefore a single score submit may have an effect on more than one leader board. Time periods are arbitrary windows of time. All games must use the all-time time period, meaning the scores are managed without regard to time. Most games will use additional time periods such as daily, weekly, monthly and others. As scores are submitted the system checks what time periods are currently active and updates each time period leader board accordingly. If, for example, a game has a daily and a monthly leader board then each score submit affects 3 leader boards: all-time, today's daily and this month's monthly leader boards. In all cases, a given user's score is updated only if they beat their prior score on the given leader board.

The system manages a score history and always keeps the players best score for each game played. The best score in the history may not necessarily be the score that appears on a leader board due to the context of the leader board query. For example, the best score earned 3 months ago may not make the leader board when showing players scores earned for the current month.

### *Leader Board*

A leader board is an ordered list of scores for a particular game in a date range, showing the best score for each user. Any given user appears on the leader board only once, and it always shows the best score earned by that player under the constraints of the query (date range, tournament, and challenge).

## Enginesis Platform Overview



Figure 9: Top-10 Leader Board



Figure 10: Leader Board with Paging Controls

### Player Ranking

Players are ranked on individual game leader board by score. We support both high-score games (the higher the score the better the rank) and low-score games (such as golf where the lower the score the better the rank). In the case of ties the player who submits first is considered the better rank.

### Game Group and Site-Wide Rank

Enginesis has a game group concept where any number of games that appear on a site can be considered together for player ranking. This is similar to the Olympic

## Enginesis Platform Overview

---

Triathlon or Decathlon concept. Players play each game and are ranked according to how they perform in all games in the group.

Ranking for game suites such as a game group or site-wide (all games on a site) do not consider the score in individual games but rather use the player's ranking on each leader board. If a player does not appear on a leader board for a certain game then their rank is the worst possible position in that game's ranking. A score is assigned to the player based on the sum of their ranking on all the leader boards in the game group, and then each player is assigned a rank considering this score. This system rewards players for participating in many games as opposed to doing well in just one game.

### *Player Rating System*

Players are assigned ratings in individual multiplayer games using a rating algorithm called Elo. The Elo rating system is a method for calculating the relative skill levels of players in two-player games such as chess and Go. The rating system confers player status and identifies like-capable players. This system enables fair player matching for multiplayer games and tournaments. The rating system takes into account experience (prior games played), wins, losses, and draws such that novices and less capable players are identified from experts.

### *Awards and Badges*

Badges are collectable digital awards that a player earns simply by playing games, participating in tournaments and other site wide events. They represent skill and perseverance in each game played and activities participated in.

Specific badges will be defined by game. Some badges, such as mastery, can be won by playing a single game, round or level. Others, such as Lifetime Total Score, must be earned over multiple (or even many) rounds: the more you play and the better you do, the more prestigious your badges become. In addition, rare and unique badges may become available only for a short time, and can be won by completing certain challenges within a single game or tournament.

All badges won are stored with the user's profile. The user can always view their badges, see what badges are available. Badges are also on public display as a community building bragging right and challenge incentive. Once a badge is awarded it stays with the user forever.

Badges are only one form of digital reward. Another form may be to earn rights to a video or music clip, special locked content on the web site, affinity points (tokens), game micro-transaction items or other in game special features such as power ups, locked levels, extra lives and so on.

Sample list of events that generate a reward:

- Achieving a threshold score in a particular game (for example: earn more than 3,000,000 in Breakout and earn a "breakout killer badge".)
- Win a particular tournament and earn a badge.

## Enginesis Platform Overview

---

- Play a certain game 25 times.
- Complete a particular game.

### *Temporal awards*

Certain awards are earned by a player but only held for a finite period of time. Usually the award is relinquished to another player. Examples of this type of award are:

1. Top-10 Zuma Player in August;
2. Highest badge award winner for last week;
3. Most active team member for the ZZ-Top Tournament.

The award is recorded in player history and newsfeeds.

### *Contest*

A contest is a period of time a game is played to declare one or more winners for a pre-defined prize. For example, “play Arcadia this month and the top 3 scores will win an I Love The 80’s Toy”.

### *Tournament*

Tournaments allow either individual players or teams to compete in a game or suite of games over a period of time. Tournaments vary in structure: there may be ladder or round-robin. It may be too early to discuss exactly how tournaments will work but we know we need to organize players into groups around a game (possibly more than one game) with a leader board containing just those results. The tournament is open for a finite period of time.

### *Challenge*

A challenge is an ad-hoc tournament set up by the web site administration or an individual player for a particular game or group of games. It takes place in a finite period of time and has a private leader board to track the results. A player sets up a challenge by identifying the game (or games) and who can participate (public is open to anyone on the site, private requires a list of emails or registered users who are allowed in) and an end date after which the challenge is closed. An email is sent to the participants (except in a public challenge) containing information to locate the challenge. Some examples:

- A challenge to a buddy list to see if you can beat my score;
- A public challenge to play a newly designed level to see who can get the best score;
- A triathlon to play a 3 game suite to see who scores the best.

### *Reviews*

Users can review games and the reviews are visible by other site participants. Reviews require content moderation by a site administrator.

## Enginesis Platform Overview

---

### *Gallery of User Generated Content*

Certain applications gather user generated content. Examples are the Wrestler Society X characters, Mencia Synth compositions, user level designs in a game that supports user generated content, and so on. A gallery is an application that enumerates user generated content by predetermined criteria, such as most popular, most voted, best, worst, and presents this list to other users such that they can browse and experience the content themselves.

## Multiplayer Services



Games allowing more than one player to compete head-to-head in either turn based or real-time. Multiple gaming models will be supported: player vs. player (e.g. Sumo Volleyball, Redneck Games), multiplayer (e.g. Horse Race Trivia, Space Arena, and Sinners Isle), massively multiplayer contests (e.g. Trivia Dome), MMOG style (we haven't done one yet) and tournaments of multiplayer games.

Multiplayer services include the following features:

- Multiplayer gaming API allowing game developers (internal and third party) to work with our multiplayer server.
- Game lobby, complete with chards, avatars, presence, player status.
- Public and private rooms.
- Spectators.
- Player matching, find an opponent based on experience or rating.
- Tournament seeding.
- Chat, chat functions such as game groups, buddy lists, whisper and block.
- Voting.
- Influence – possibly tied in with status, influence increases/reduces one player's or team's ability based on observer and other player's consensus.
- Lobby and player enumeration functions, XML feeds to communicate games, lobbies, active users to 1<sup>st</sup> and 3<sup>rd</sup> party affiliates.
- Ability for players and spectators to view and comment on games (e.g., the final outcome of a tournament) using comments and forums.

### *Lobby*

The multiplayer games have a lobby system that defines available rooms, players associated to each room and available seats. A lobby may contain players from other web sites hosting the same game (for example, a Texas Hold'em poker game on Comedy Central, Spike and CMT all pull their players together to host a more active lobby than any one could do on its own.)

### *Public and Private Rooms*

With multiplayer games a public room is open to all site users. A private room is available only to those players who know about it requiring prior knowledge of the room and name password. A moderator is assigned to the room to control access and

## Enginesis Platform Overview

---

set attributes. Private rooms may or may not be visible in lobby listings (moderator defined setting).

### *Spectator*

A spectator is a participant in a multiplayer game who can only observe the game play performed by the active players but not participate in the game. Depending on game setup observers may be able to chat.

## Chat Service



Chat is a complicated topic that will require ongoing research. We have implemented chat in multiplayer games using our built-in multiplayer services.

We have deployed site-wide chat using Meebo. Using both internal multiplayer and Electrotank we provide a comprehensive service tied in with other Enginesis services such as profiles, avatars, friends, leader boards, badges and player ratings.

We know we want services such as site-wide chat such that players not currently in a game can find other users and chat with them for any reason including coordinating a new game. Web site community service could offer chat functionality with similar requirements without a tie-in to online games. This should be tied together into one site-wide chat with the added ability to offer chat within a particular game. The chat feature will require features such as buddy lists, block and whisper. The platform will expose a chat API such that third party developers can easily build games utilizing chat.

Current chat requirements:

- Presence service (player is currently logged in) and status;
- Avatar and other profile attributes;
- Blocking;
- Individual (whisper) and group chats;
- Spectator chat;
- Chat amongst participants in a given game or game lobby;
- Private table vs. public table chat;
- Message center (leave message for particular user to be picked up at next login) ;
- Tie-in with mobile devices;
- Kid-friendly chat (i.e., select phrases from a pre-canned list);
- VoIP – live chat while playing a game;
- Buddy list – friends & foes.

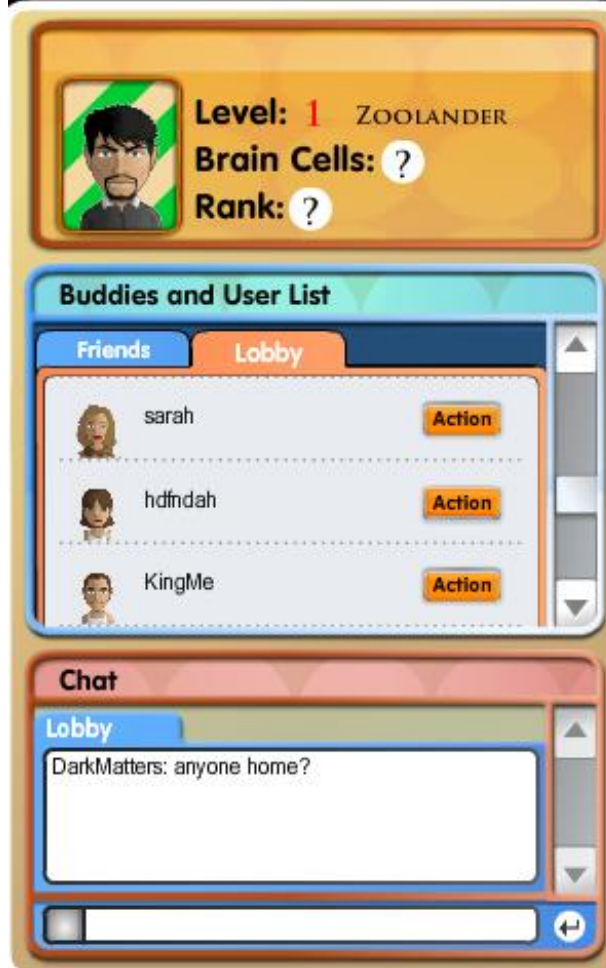


Figure 11: Chat Widget on worldofpop.com

## Notification Services



A critical service of a community is communication. Users will require notification of important events of interest to them. The service will also notify users of news, new events, marketing campaigns and things we haven't thought of today. User's profiles will include switches to indicate the information of interest to them.

Samples of email services include the following:

- Tell A Friend;
- Site support services such as registration confirmation and forgot password;
- Game Challenge;
- Tournament/Challenge initiation, progress, results;
- Leader board turnover;
- Newsletter (or interface with 3<sup>rd</sup> party providers such as CheetahMail);
- Alerts.

## Enginesis Platform Overview


---

Enginesis also provides a short message notification system. Each profile has a notification queue. This works similar to the Facebook newsfeed. Notifications that may appear in a user's notification queue include:

- Challenge event (you have been challenged by *so-and-so...*);
- Leader board turn-over (you are no longer in the top-10...);
- Buddy acceptance;
- Comment (*so-and-so* added a comment to your profile);
- Friend and watcher in-game events (*so-and-so* just beat your high score in *this-game*.)

Short message notifications can also be published to external sources such as Facebook and SMS text messaging.

## Tracking & Metrics

 Tracking is an important module because we require a firm understanding of usage patterns, game play metrics and ROI. Typical web tracking does not support gaming well because many of the usage events important to our understanding are outside traditional web logging.

Tracking metrics include:

- Game downloaded;
- Game played;
- Game completed;
- Time spent in game (average by game, group of games);
- Aggregates by game, game group, user, genre;
- Referrers to game;
- Game milestones achieved (defined by game, for example level 1 completed, boss beaten, pick-up earned).

### *Tracking Milestones*

The tracking interface is designed to communicate certain game milestones to the tracking server (Omniture). This allows metrics to be recorded for important game events used to determine the effectiveness of the game with the intended audience. The game milestones are:

- |             |   |
|-------------|---|
| <b>Load</b> | This event is triggered when the game load is complete and the game is ready to play. This event is different than the load request event on the web server (triggered when the web server initiates the download of the content) because it signals all assets are loaded and the game is ready to play. Typically there is only one load event per game session.                                |
| <b>Play</b> | The game play event is triggered each time a new game is begun. Usually this is triggered by the user clicking a <i>start game</i> button or similar GUI element but some games automatically start play once loaded. It is up to the game developer to call this milestone at the proper time. Depending on game design, there is at least one play event per session, possible more if the game |



## Enginesis Platform Overview

---

design allows the player to start a new game.

The play event takes an additional parameter used to indicate which game component was played in the case of game suites (not levels – see Zone Change).

### Game Over

The game over event is triggered when the game ends. This signals a completed game and is very important to gauge game play behavior. There should be a play event to match any game over event, but there may not be a game over event matching a play event if the player doesn't finish the game.

The game over event takes an additional string data parameter. The value of this parameter is completely up to the game as to how it is used. In most cases set it to an empty string. In games where a win/lose scenario makes sense this parameter should be set to either "WIN" or "LOSE".

### Zone Change

The zone change event is game specific. A zone change is a game event such as a new level or a new environment or some other significant event. This event requires an additional parameter to uniquely identify the event to the tracking system. The data parameter to this event uniquely defines the milestone such that the tracking team can query the event and build reports.

## Ecommerce



The ecommerce service supports a variety of transaction services utilizing cash payments, currency (our affinity points) and trading. We identified some of the ecommerce functionality as the following:

- Trade in accumulated points (currency) for digital or physical rewards. Digital rewards could be exclusive music videos, free downloads of premium content, in-game level access, in-game power-ups, or other digital swank to place on your personal profile page or avatar. Physical rewards could be DVDs, electronics, apparel etc. when linked together with a fulfillment provider.
- Purchase premium games (pay to play casual download games) or premium game plays (in games that support pay-to-play).
- Purchase in-game items (in-game items are defined by each game, such as special weapons, extra lives, locked levels, wardrobe enhancements, etc.)
- Purchase digital rewards or purchase items that come with points to fund power ups, characters, avatars, locked levels, etc.
- Premium membership, subscriptions to web sites and online services.

The e-commerce system is scheduled for future development. Currently there are no projects with an e-commerce requirement. However, there is a lot of discussion of adding in-game items in our multiplayer games.

### Social Services

Online games these days are all about connecting with user's social networks. We want to be where the users are as opposed to waiting for the user to show up at our online games web site. To facilitate networking Enginesis supports other social networking web sites API and aggregation widgets.



The Enginesis Flash SDK utilizes the gigya widget for sharing. If the developer chooses to do so, games can be easily shared and embedded in an ever-expanding world of social and network contexts to bring the game experience to the user.

#### Facebook

Enginesis supports Facebook Connect, Facebook profile linking, newsfeed updates, using Facebook friends inside a game, and rendering games as a Facebook application. There are several examples of these features in existing games.

Open Social integration is not yet available but is under consideration.

Flux is supported for profile connection, co-registration, friends and profile image.

### In-Game Advertising

Usually games are supported with advertising shown in the periphery, such as on the hosting HTML page. For games this is suboptimal for a few reasons:

- Limited targeting, such as matching the ad with in-game sponsorship or theme;
- Performance issues, such as Flash contending with Flash or other CPU intensive plug-ins;
- No context sensitive control, such as showing ads at game pause, game over, level up, etc.;
- Poor ad placement and interrupted user experience.
- Cannot deliver ads to games that are embedded in blogs or social media sites.

An in-game advertising solution is powerful yet has many issues of its own. However, most online business is based on delivering advertising and optimizing impressions. In-game advertising is an important feature to leverage current web site ad sales initiatives, improve ROI on games, and track ad delivery within the context of user behavior playing online games. Individual games should have the ability to define ad placement and campaigns. Enginesis delivers on this requirement with the client-side SDK and server-side support for ad integration.

## Enginesis Platform Overview



Figure 12: Example Pre-roll Static Image Ad 300 x 250

Enginesis provides a simple API for developers to call out ads and describe to the ad serving engine the type of ads spots that are available. Definitions include pre-roll (ad shown before game play begins), interstitial (ad impression between levels or between games), and post-roll (ad display at the end of a game). A provision exists for in-game ad placement (such as in-game billboard or product placement) but this is very game specific and difficult to deliver from the ad sales and delivery side. The game database will provide the rest of the information necessary to match ad campaigns with game context and send back accurate ad tracking metrics.



Figure 13: Sample in-game Sponsor Logo Placement

## Enginesis Platform Overview

---

The ad service can link to a few existing ad providers such as Double-Click, DART compatible ad servers, Yumi and CPMStar. Other providers can be integrated as necessary.

### Localization

Many of the server-based services are localizable. The system tracks a user-session based language code and many service APIs will return human-readable data translated to the selected language. This will work for all system related message, but not for user data. For example a user's name will not be translated into multiple languages.

### Search



The search service allows a user to search web site content such as user profiles, game titles, game descriptions and most user generated content (reviews, forum comments, etc.)

The search facility currently supports finding users or games by partial keyword matches.

To support Search Engine Optimization (SEO) we capture keywords for each game. These keywords are typically managed by the site producer. When games are requested with our variety of listing APIs the keywords are sent along to be consumed as fit by the hosting web site.

### Quiz Widget

The Quiz Widget is a special game for flexible quiz games. Most of the definition and configuration of quizzes is done on the server utilizing a user-friendly Content Management System. A Flash client presents the quiz to the end user in a customizable widget. Quizzes are not limited to Flash presentation: all quiz services are provided through the Enginesis API and can be utilized in any presentation medium, such as Flash, Ajax, HTML or any other technology discussed elsewhere in this document.

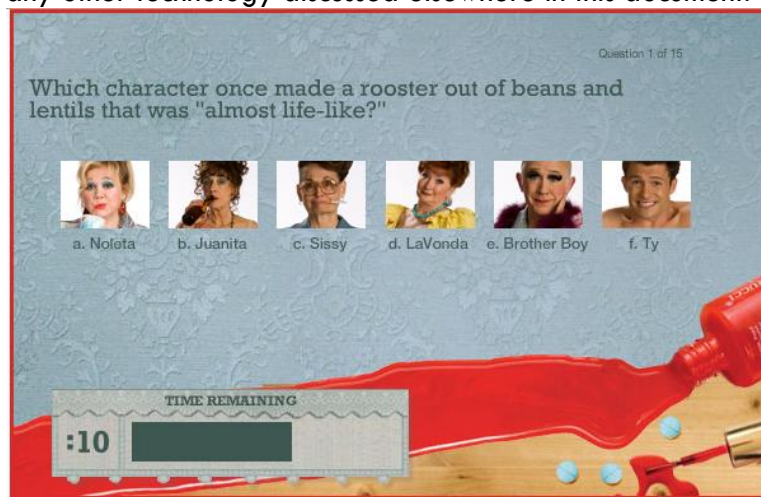


Figure 14: Example Quiz Widget



## Enginesis Platform Overview

---

### Client Side Developer SDK

We provide client-side Software Developer Kits (SDK) to facilitate easy integration of Enginesis services into games and applications.

The Enginesis Flash SDK currently supports the AS3 development environment. An AS2 SDK is under consideration depending on specific requests.

The Enginesis PHP SDK facilitates web site development where PHP would be used (Apache, IIS) and quickly building Facebook applications utilizing Enginesis services.

The iOS SDK will be entering beta soon.

SDKs for Unity, Android and Windows 7 are under consideration.

