EA.com Architectural Overview

Produced for
EA.com

Prepared by
Lars Smith
Alexander Scholz

Version 1.0
November 10, 2000

Table of Contents:

Release History

Version	Author	Date
1.0	Lars Smith, Alexander Scholz	12/10/00

Location and Dependencies

Document Location	
Code Location	
Dependencies	

1. Overview

This document provides an overview of EA.com's basic architecture. When we wrote the first draft of this document in October, we said that it was not just aimed at new hires, but at all of EA.com's rapidly expanding 400+ employees. Well, we already have 500+ and are growing fast.

EA.com is a big site, with many features. And none of it existed before last March. If you do not have a clear picture of all of the different facets of EA.com, you are probably not alone. Hopefully this document will help clarify things.

In some cases you might have a very clear idea of some aspect of EA.com's architecture that has been presented here, and feel that you could explain it more clearly. Please feel free to do so. Documents or section rewrites may be submitted to:

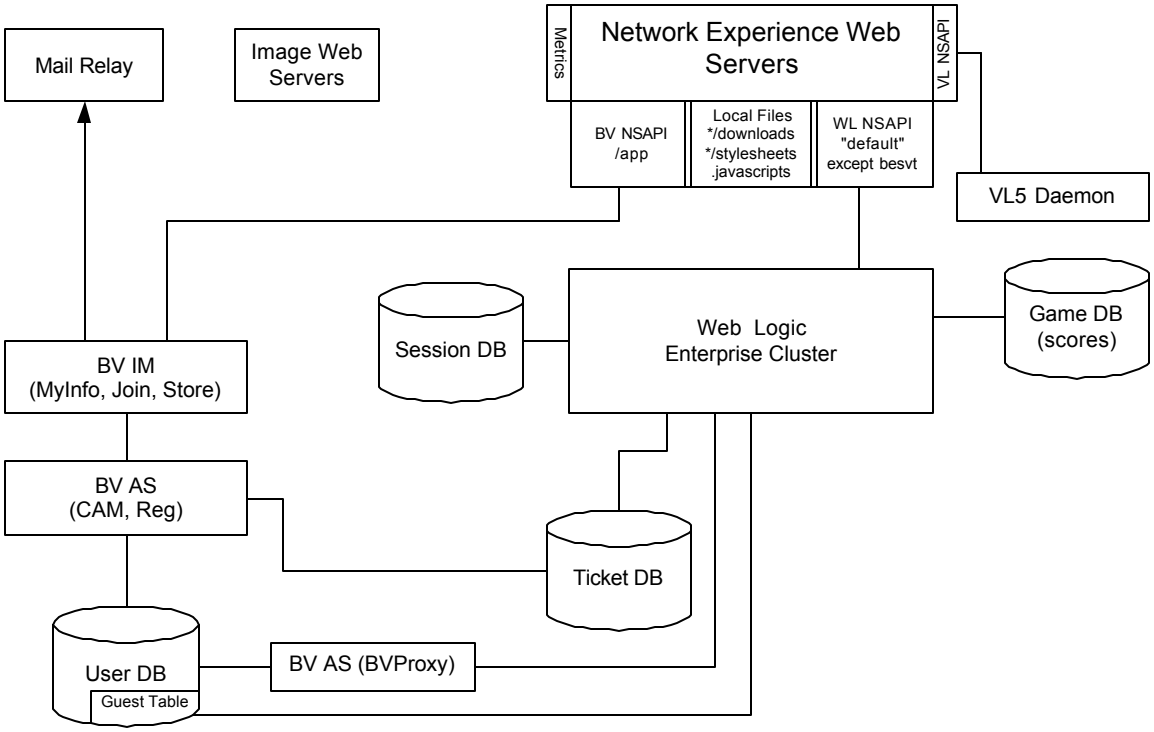
BOS SUBMISSION @ EAHQ

2. URLs

EA.com actually consists of multiple sites, termed "brands," which serve various external entry points to our Web properties. These "brands" share general structure, but differ in appearance and in content. Whether users enter via Netscape, Net Center, AOL.com or EA.com, they will encounter pretty much the same site and functionality, just with a different look and feel.

Brand	**Entry Point**
EA.com	www.EA.com (EA Internet Website)
aol.EA.com	AOL client
aolcom.EA.com	HTTP://www.aol.com/ (AOL Internet Website)
compuserve.EA.com	Compuserve2000 client
cs.EA.com	www.cs.com (CompuServe Internet Website)
netcenter.EA.com, soon to become	www.netcenter.com,
netscape.EA.com	www.netscape.com
icq.EA.com	www.icq.com

3. Web Pages



- HTTP requests are forwarded from the Enterprise Web servers to the WebLogic Application servers via the WebLogic proxy NSAPI (Netscape Application Programming Interface).
- BroadVision pages, indicated by URL, are forwarded to the BroadVision Interaction Manager servers.
- Downloads, style sheets and JavaScript files, also indicated by URL, are served locally from the Web servers.
- All images are served the image Web servers and are referenced using absolute URLs, HTTP://images.EA.com/ .

If a Web page is requested with the URL path subsection **"/app"** it goes to BroadVision, default goes to WebLogic, **"/downloads,"** gets served locally off the Web server. Downloads are all dynamic files. The same thing goes for **"/stylesheets"** or **"/Javascripts."** This site serves little static content. Nearly every page viewed by the customers is a dynamic JSP (Java Server Page) generated by WebLogic.

3.1 ClearCase and TeamSite

ClearCase is the content repository for code and TeamSite for development and revision of Web pages. TeamSite has a templating system whereby Web code developers do not build final pages, but templates, which are missing key text and images so content is entered into the pages with a data entry tool that generates JSPs. This allows editorial staff to make changes without breaking the templates, to ensure there is a consistent L-frame on every page with minimal coding, and avoids a lot of dynamic includes when processing a page.

3.2 Network Entry & Login

- WebLogic serves login pages.
- EA.com account names and passwords are passed to the BroadVision Proxy using custom coding that allows WebLogic to communicate with BroadVision.
- The BroadVision Proxy validates using the user database and retrieved information is returned to WebLogic.
- User can set automatic login using a cookie.

When cookies are used to track customers, the cookies are checked against their user database entry, and thus we can validate a given user. Cookies contain basic information that identifies who a user is, such as a hash check sum off their password.

3.3 WebLogic

WebLogic is a Java-based Web application engine complying with the J2EE (Java 2 Platform, Enterprise Edition) standards. The Web pages served up are Java Server Pages, i.e., HTML files which can contain Java code and invoke other WebLogic Java objects. Also accessible to customers are Servlets, that is, Java code accessible with HTTP requests.

Other objects include classes or Enterprise Java Beans (EJBs), which are accessible to the JSPs and Servlets, but not to the customers directly. Java Enterprise Edition defines standard ways that Java code can interact with other pieces of Java or code, such as communication cues, and the ability to invoke Java code, which might or might not be on a different server.

We use NES (Netscape Enterprise Server) to serve connections to the customer because it is more efficient than WebLogic at opening and closing connections to the Internet. There is a WebLogic plug-in for Netscape, the WebLogic NSAPI that allows Netscape to pass on code communication to WebLogic.

In simple terms, a request comes in from one of the various entry points and gets passed on to WebLogic, is processed, and it gets passed back as HTML. This is the basic core of our site, and this is how a customer comes to see Web pages. For a lot of sites this would be enough, but what we are trying to do is provide a more complete experience with login and registration.

3.4 BroadVision

BroadVision is similar to WebLogic in that it is a Web application engine that provides dynamic pages. BroadVision also provides JSPs and also has an *NSAPI*. We have both for historical reasons. EA.com purchased WebLogic this last winter, whereas BroadVision was originally integrated two years previous for the Ultima Online registration site.

The key strength of BroadVision is that it provides a lot of out-of-the-box functionality. It is positioned as an e-commerce or e-marketing tool, and has some of the ideas of customers and a shopping site built into it. We use these BroadVision tools for the Online Store. For the registration system we leveraged some of the preexisting functionality from Ultima Online as well as the built in functionality of BroadVision. These actually provide the backbone to Registration (Join Now), CAM or Customer Account Management (My Info).

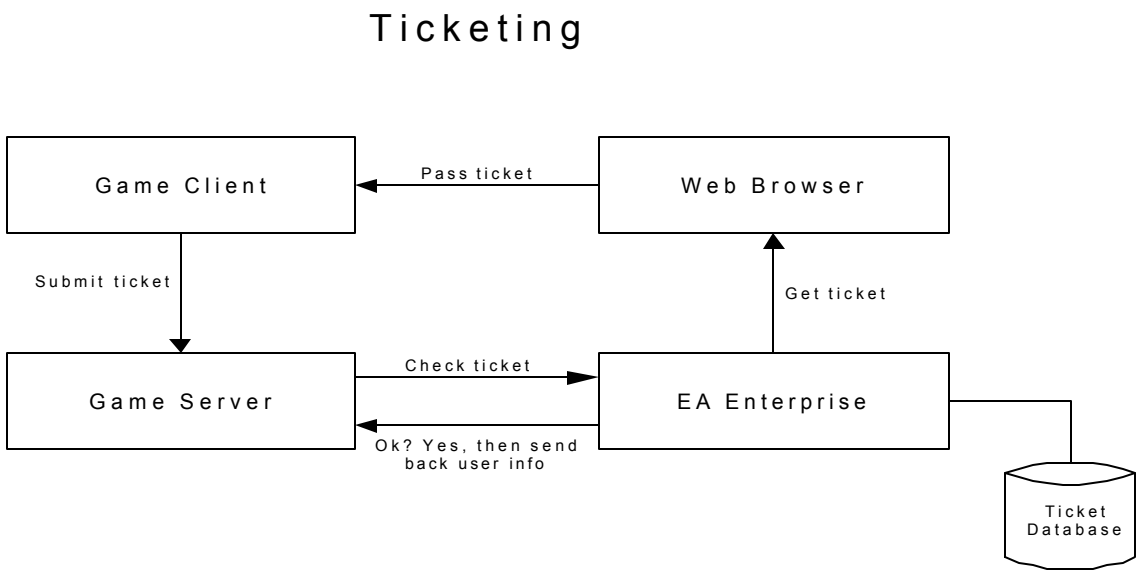
The BroadVision interaction manager, which is sometimes called the session manager, and the BroadVision application interaction manager is a page server and renders JSPs as well, keeps track of user sessions, what pages have been viewed, and what page should be viewed next.

3.5 Transition Between Servers

Transition pages are close to seamless, i.e. from the EA.com welcome page and registration. Users are not redirected to a different URL. Mapping is in the configuration of our Web server, so that when a user goes to certain subdirectories, it sends the request to BroadVision, or WebLogic as necessary. The user goes to the site, looks at the JSPs, clicks registration (Join Now), and there is a redirection. What is important is that after the user has arrived at WebLogic we keep track of him.

From page to page, we know that it is the same customer, and thereby maintain some sense of state on WebLogic. We know the last page viewed, we know where they are on the site. If a user goes to play poker, and is not a registered customer, and not at the EA.com registration site, he cannot play until he has registered. This means that instead of seeing a button that says, "Play," he will see a button that says, "Register." When he clicks that button, it will take him to the registration page, and when he has registered, it will take him back to the same game page that he was at before.

4. Ticketing



4.1 Registration and Account Management

- WebLogic generates a ticket in the ticket database and redirects users to a BroadVision page, passing the ticket context ID on the redirection URL.
- The BroadVision application server uses the context ID, essentially the number generated for a ticket transaction, to look up the ticket and identify the customer.
- After the customer has complete registration or maintenance activities, the customer is redirected back to a WebLogic page.

The way we keep track of our customers is through a process called ticketing—one of the key functions within the EA.com site architecture. The general concept of ticketing is that at any given point we have a certain knowledge of a given user at our site that may include sensitive information such as authentication or knowledge of what customer they are. This information needs to be passed between applications. Maybe it is a question of the Web server talking to another Web server, maybe a client talking to another application or a game server.

An example of this would be that a customer is at a JSP, clicks a play button to play a game, and WebLogic generates a ticket which is passed to a Web browser, then from the Web browser to the game client, then from the game client back to EA.com. The back end portion of the game server goes against WebLogic and figures out if it is a valid ticket. The basic premise is, if a piece of information is given out on the Internet, and that same piece of information comes back from another entity, the user's identity is thereby authenticated.

Other parts or our site are not really central parts of our service, but is still integrated via our ticketing system. WebCrossing, which handles our message boards, passes a ticket that is essentially a context ID identified by a long number. That ticket gets passed to the Web browser, which passes it to WebCrossing, which takes it the EA.com enterprise database, which gives back all the information. Tickets verify the validity of the service being requested. If a user has not logged in, a ticket can not be generated. *Refer to the sections dealing with Auth/Auth/Ident and AOL for more information on Registration.*

4.2 Entitlements and Validations

Tickets also are generated listing entitlements. A user can not go to the message boards and get a ticket and then go use it to play a premium game. Information is passed to Enterprise and WebCrossing only via the ticketing mechanism. What is stored in the user database is information like the user name and address, as well as a concept of entitlements, which essentially controls which people are allowed to play what games.

Ticketing is also used for actual game play, but functions in a slightly different fashion. When a customer goes to play a game, say Bunny Luv or Pro-3-Point, and clicks the play button, the browser generates a dynamic JSP file that embeds the game. That dynamic JSP file provides information such as where the game is, how to load the game, and the ticket to load that game. The game starts with a number of arguments, one of which is the ticket. The game has an API that allows it to send that ticket back against our Web servers, and that is how a game like Bunny Luv gets a customer's screen name to show up. That is also how score reporting uses tickets to validate that a score is properly associated with a user.

The ticket validates the user, and then further validates that reported scores match that user. Importantly, tickets prevent the user from manipulating the game into making it think that the user is someone else, and also make sure that a user cannot take the game and play it when not dialed up to EA.com. A user cannot just take Bunny Luv and put it on some other game server and play it, because it will generate a ticket that cannot be validated. The same is true if a user tries to run Bunny Luv from their Web server cache. It will fail to start, because it will not be able to contact the EA.com server to authenticate the ticket.

5. AOL

5.1 Registration & AIM

As a partner of AOL.com we are integrated with their namespace system. When users register for an EA.com account they are simultaneously being registered for an AOL AIM (AOL Instant Messenger) account via EWOKS (External Web Oscar Knowledge Server), which is the registration system for AIM. Users talk to BroadVision, BroadVision talks to EWOKS. If a user has an existing name and password from registration on AIM, he can actually give us his AIM name and password and the EA.com registration system will be able to authenticate it.

5.2 VL5

- AOL client and CompuServe client customers are identified using VL5 (Virtual Lock 5).
- AOL caching Web proxies, known as spiders, add encrypted HTTP headers containing the customer information.
- The Netex Web servers pass the headers for certain URLs to the VL5 NSAPI, which uses the VL Daemon to decrypt the headers and pass the customer information to WebLogic with the request.
- The AOL client shares cookies between multiple screen names, so VL5 is essential for logging in these customers.

Virtual lock VL5 was originally designed to integrate the customer base from CompuServe and AOL. We use the encrypted information passed on by these VL5 headers passed on in HTTP requests. Our Web servers encrypt that information using the VL Daemon, inserting the actual AOL header before sending it to WebLogic, which means that AOL customers do not have to login because we can already tell who they are from the AOL headers. We keep track of AOL customers that are not registered in the guest table, which keeps track of who comes as an AOL guest to play. For convenience and storage the guest table sits underneath, and is part of, the user database. It is unclear what amount of traffic Netscape will draw, but we do expect that AOL will represent about 2/3's of our user base. Because of this, a lot of care has gone into making it easy for AOL customers because they represent the Lion's share of our customers.

There are other reasons that EA.com uses the VL5 headers. One is that they work well with AOL browser, which most people do not realize is actually Internet Explorer 5.0. Another is because AOL allows multiple screen names, the idea being that every member in a family can have a screen name. When users go to EA.com it saves their screen name as a cookie, which is a common enough idea. The problem is that since AOL is a browser, no matter what screen name a user is using, he has the same set of cookies. If a user "A" goes to a site that sets a cookie, and then lets someone else, User "B," on the same account switch to their screen name without shutting down, they have the same cookie. Accordingly, if the user "B" goes to the same site, they show up as user "A." It is easy to imagine how this could create havoc with issues such as score reporting. We solve that problem by AOL customers using VL5 headers, and not cookies, as a login mechanism. This way we know if they have switched screen names.

6. WebLogic Sessions

- A WebLogic session is created for each visitor.
- A cookie ensures that the WebLogic NSAPI forwards the request to the correct WebLogic server.
- Core session information is also put in the session database.
- After the user is idle, the session is passivated from memory to disk.
- After more idle time, the session is deleted from disk.
- If the customer returns, the session is recreated using the core session information from the session database.
- A batch job deletes old session information from the session database.
- When a new session is created, WebLogic checks to see if the user already has another session using the session database.
- An already existing session is deleted, ensuring that a user is logged in only once.

URL	<HTTP://www.ea.com>
Path/Extension in URL	Content Type
/ (default)	WebLogic
/javascripts/	static, local file
/stylesheets/	static, local file
/downloads/	static, local file
/app/	Broadvision
/besvrt/	blocked

E.g.: HTTP://www.ea.com/javascripts/play.js is a static file served from NES.
HTTP://www.ea.com/lounge/home.jsp matches the default and is served from WebLogic.

When users come to WebLogic, WebLogic creates a session. EA.com does not want to keep that information around forever, but it is impossible to know when someone stops looking at Web pages, because no one uses a log out button. We have to therefore have a time-out that says if a user does not access those pages in a certain amount of time, clear out that piece of memory. The problem is that with a lot of the games, like Bunny Luv, there is no communication with the Web server. What happens when they play a game and then they come back and they want to play another game? We did not want them to have to log-in again, so we created a mechanism whereby they can get cleared from the memory, but they can come back and get invisibly logged-in again.

We do this by having a session database that contains information to the effect of what person was on what WebLogic server. The server must be specified, because there is not one WebLogic server but a two clusters of WebLogic servers, fifteen WebLogic servers on the East Coast and eleven on the West Coast. Each runs two copies of WebLogic, which makes for a total of 52 instances. When a user accesses a WebLogic page for the first time, it generates a cookie that tells this NSAPI to send a request to a particular WebLogic Server. The 2nd request that goes back to EA.com, it actually contains a header that associates a user with a Web server. Accordingly, if we shut down one WebLogic server, clients that had open sessions with that WebLogic server will be treated as new and have to go through that original logic sequence again.

7. Session Database

Our session database tracks what customers are where on the site, and what WebLogic system they were on. In simple terms a user plays Bunny Luv via a WebLogic server, but after fifteen minutes the WebLogic server has cleared its record of that user. When the user goes back, WebLogic has no record, but the browser has a cookie that says that there was a running session on that WebLogic server. WebLogic then refers to the database where that last session is recorded. The database says, yes that user was there, but they were cleared, and then recreates that EA session object. Of course the system has to be purged to keep from over-loading, because every player ends up in the interim database. The way we do this so that sessions are maintained, is that every four hours sessions that are more than eight hours old are purged, the idea being that no one is going to play Bunny Luv more than eight hours. And if they do, that probably has rewards of its own.

The session database also prevents multiple players from using the same password. If more than one user logs on with the same password, than the first login is invalidated. When the first user of a given password then goes to view a different page, they are shown as not being logged on. So to speak, the last login wins, and deletes out any other session entries.

7.1 Session Database vs. Ticket Database

The session database and ticket database are actually part of the same database, just with different database tables. We did this because they had similar performance profiles, both storing lots of small bits of data that get interacted with frequently.

Once a user subscribes, there is an entry in the user database around the actual account. This translates to many more levels of entitlement. Now we have only anonymous entitlement, guest entitlement and member entitlement. When we start having special promotions the session database will see if a given user is someone with entitlement at that level, and then he can play those games allowed by that special offer. When a session in WebLogic is created, a page will dynamically render based on the information in a given user's session, determining whether that user sees a play button or register button.

7.2 Score Reporting Database

When a user reports a score back from a game, it goes through a Servlet. The user hands that Servlet their score, the Servlet submits a ticket to verify that the user is submitting a score for the game he was supposed to be playing, (A little anti-hack to make sure a 3-Point-Shoot -Out score is not submitted for Bunny Luv) and that gets reported to the scoring server database which determines if that score is good enough to be in the top 100. If it is, it gets added, if not, thrown away. Leaderboards get their information from the score reporting database. We are still working on someway to cache that in memory, but at the moment it is a database transaction every time someone goes to a game hub page.

8. Authentication/Authorization/Identification

- A block of data is stored in the ticket database and a corresponding context ID is passed to the browser.
- The browser is directed to pass that context ID to either another Web server or to another client application.
- If the context ID is used to transition to another Web server, the context ID is then passed back to EA.com or the request URL.
- If the context ID is passed to another client application, that application then passes the context ID either to its server component or the Authorize Servlet.
- The EA.com component receiving the context ID uses WebLogic to validate the ticket.
- WebLogic can then pass back identification information to the Game Server.
- The ticket is good for only one use.

There are two different levels of ticketing. There is the actual ticketing mechanism and the authorization system (Auth/Auth/Ident). The Auth/Auth/Ident system provides authentication, authorization, and identification services to game clients and to other Game Technology or Enterprise Group systems via a specific API. The system does not provide initial login or authentication for players on EA.com; it provides additional authentication and authorization services for players who are already logged on to the Web site such as score reporting and time and title sensitive information.

Games talk to Auth/Auth/Ident which in turn talks to the ticketing mechanism, because Auth/Auth/Ident can only retrieve a much more limited set of user data than the full ticketing mechanism: name, city. Auth/Auth/Ident filters that information without returning it all back like the regular ticketing mechanism. This is done because tickets for game play are only in the database for game play for a few minutes, but are kept around for score reporting purposes for eight hours.

The eight hour limit was chosen somewhat arbitrarily, because that is what we expect to be the time limit that most people would play, though in theory someone might play something like Jungle Strike 18 hours beginning to end. Auth/Auth/Ident also determines whether a ticket should be routed to the East or West Coast based on the context ID.

One last important aspect of authentication is that with Auth/Auth/v2 it is not necessary to start a game from the Web site. A ticket is generated with Auth/Auth/Ident only when a game is started. Auth/Auth/V2 allows a user to start a game from their computer and the Web server is contacted with logon information, and then generates tickets so that they can use that ticket to authorize a game without being on the Web.

9. Domestic Network Infrastructure

- AboveNet is our ISP (Internet Service Provider).
- We have an AboveNet in San Jose, abn-sjc, and AboveNet, Virginia, abn-iad.
- Our VPN (Virtual Private Network) connects us directly to both.
- Traffic is paid for in both directions, which consists mostly of replication for message boards and ticket checking.

Our deal with AOL gives us as much free bandwidth to their customers as we want, as well as free rack space. In theory we could relocate all of our servers to AOL, but AboveNet is much more responsive to our needs. Direct fiber connects us to AOL, and can be upgrades for bandwidth as needed. Right now we have a pare of OC48's to AOL's network, something like 48 times more capacity than the DS3 to HQ, so that we could handle a hundred-fold increase in traffic over what we see right now without having to upgrade. Everything is redundant, and we are connected to AOL on the east and west coasts so that we can even route from west to east via AOL in need.

When a user goes from one ISP to another it normally has to go through a public exchange. We use a private provider, PIX (the Palo Alto Internet Exchange) to avoid low performance public networks. Routing based on context IDs facilitates load balancing.

We do not have two separate copies of member data, because that would require synchronization and unnecessarily increase costs. What we do have is a stand-by warm copy. The east coast is the main facility, and the west coast serves more of a backup facilities role. If a transaction is made, it does not get replicated instantaneously, but quickly. Because information from the user database is business critical, we replicate it transaction by transaction. This might lead one to wonder that because we do not synchronize data, if it is possible to play on one user name on the east coast and west coast concurrently. Yes, two different people could in principle play on the same user name at the same time, if they tricked their browser into sending traffic to the server on the other side, because clusters do not interact. But ultimately users could only be logged on twice to the same password, not a third time.

Along the lines of our domestic network infrastructure, it should be noted that TeamSite, which we use for the development and templating of Web pages, has a partner, Open Deploy, which is pretty much a file copying deploy. Pages are sent to open deploy and served to east and west coasts more or less at the same time to avoid a lot of call traffic from the east to the west coast.

10. Game Play

Game archetypes allow us to group our games into functionalities. Most games fall into one type of archetypes. If it were not this way, maintenance and development costs would be excessive.

10.1 Gamettes

A gamette has no client server interaction, runs in a browser, or is a lightweight install. It maybe uses score reporting, maybe it does authentication. Examples are Pro 3-point, Bunny Luv, Trap Shoot, Street Slider.

10.2 Gils and JVGAS Games

One of the most important questions EA.com faced with AOL was game integration. How can EA.com integrate games that are already on AOL with the least amount of work? GAS (Game Authentication Server), which was already part of AOL, would communicate with the game server to reserve a spot. Different than ticketing, GAS reserves a spot for a user for an amount of time based on an IP address, and decides where to get the package and what are the arguments too feed in, in order to load. It used DILS (Download, Install and Launch Services) to install zipped files from AOL, and was a fairly large program that was installed from the AOL CD. We designed JVGAS (Joint Venture Game Authentication Server), and GILS (Game, Install and Launch Services) to replace GAS and DILS.

JVGAS now talks to the Web server, which talks to WebLogic via COMIC (Communication Interface Component), which saves WebLogic the trouble of translating C to Java. To avoid COMIC becoming a bottleneck, WebLogic talks to COMIC via JMS (Java Messaging Service). Unfortunately, JMS is proving buggy, and we are considering placing COMIC directly inside WebLogic as a way to cut out the JMS.

GILS is the bootstrap that knows how to install games. GILS is installed via an ActiveX Netscape control called the bootstrap which saves the customer having to determine if they want it installed and how to install it. GILS then talks to NES, gets a ticket, caches the ticket and returns the information that says here is what needs to be installed, and here is what needs to be done to launch it. GILS goes and retrieves those packages from our Web server as zip files.

Even though we designed GILS to help us with the AOL transition, it will stay. We have another custom client in addition to GILS for our Marimba game server called DIMPLE (Download, Install, Maintain, Patch, Launch and Execute) but DIMPLE and GILS have their distinct advantages. GILS has the concept of dependencies and packages, which give the user a choice between some optional packages he may or may not want to download, and GILS can also install things in the system directory, which DIMPLE cannot. DIMPLE on the other hand does bit-wise patching which allows downloading of the minimum number of bits for a given game.

10.3 Voltron Server Games

Voltron games have a prefabricated architecture. They have all the communications, all the concepts about users and data storage abstractions: pretty much a complete client-server game, and all that is missing is the game. This makes it easy and cheap to create simple parlor games like card games, or chat games like Iflirt. Developers take the monolithic code base and they just have to extend it to their C++ classes to extend the code functionality.

10.4 Hi/Q Browser Titles

There are a lot of interesting ideas that get nixed, like Cyberpark, 3-D Avetar Chat, CyberWorld. But that is also one of the strengths of the management, to decide when to say no, and have the ability to focus. A good example is Play Station 2. We have put a lot of resources out there, but we are coming out with more games than anybody else. Ultimately what is important is business, and that allows us to concentrate. It also means that some of the cooler things get pushed off because we cannot allocate the appropriate bandwidth, or because they will not be as profitable as the Hi/Q browser (High Quality browser) titles.

With Hi/Q browser titles we strip out the sound track, the video motion, a few game features, pare it down and make it run inside a browser. Right now they have FIFA 2000 running in a Web browser with hardware acceleration and direct draw circuits inside a browser. They managed to pare down a 600Mb game to a four Mb download, which is pretty incredible. The FIFA 2000 Hi/Q title looks as good as the PC title: it has all the same textures, all the same zoom angles, weather effects, lighting effects, night games, everything. That's basically the Hi/Q browser concept. Rather than going out and buying a $45 game, a customer can access a half dozen titles from last year for $5 a month. And we think a lot of people would be willing to pay for this.

With Hi/Q browser games, the key functionality is on the page. The game page itself and play page launch the actual game. All the work that went into the game page, partially because they removed all the settings and options, and what is left is the game engine. To start the game engine the user has to define the parameters: what players, what game settings, if it is a racing game, what car and track. When they go to launch that game, there is a massive amount of information that has to get on that page as a parameter to that ActiveX or Netscape plug-in.

What gets emphasized a lot at EA.com is the network experience. Network experience is making it easy, making it TV simple. We want the play button to mean play and not mean wait there for a two-minute or twenty-minute download. We want to tell the customer upfront what they can and cannot do. The way we get FIFA down to two games is to reduce it to four teams. The user can play with more teams, but then it is a bigger download. Nascar for example has a core download of 4 MB, but if a customer downloads all the cars and all the tracks, it is 20 MB. The page is going to have options that indicate that choosing those options requires additional downloads. If they take the minimum, they can play right now.

The way it works is that there is an ActiveX control called Snoopy and Sniffer, which examines the users' hard drive to see what they have. Snoopy and Sniffer can interact with Java script control and examine the hard drive for components that have been selected, ask the user if he wants to install them and if he agrees, uses DIMPLE to go and get it from Marimba. There is really not a lot of complexity for people responsible for Hi/Q games' backend: ticketing and score reporting. There is a lot of complexity around the browser development: the use of Snoopy and Sniffer, the complexity of bugs that might cause crashing, KMS (Kesmai Matchmaking System) clients that can crash computers and WIN32 code. This is made more complicated because many are multiplayer titles (FIFA, Java Golf, Nascar, Need for Speed).

10.5 Emulators

Emulators allow EA.com to integrate games cheaply relative to developing them internally. We coded a generator for the sake of Genesis, because EA had already paid to have it developed. When we did that, we signed an agreement with Sega to design the emulator in a way that it could not be used to play non-EA games. We were able to do that partially by setting up the emulator to only run and start when it is run and start against EA.com.

Emulator was developed by a company, Game Nation, and acquired because it complemented our strategy to build open interfaces into our system so that we can host other games with our ticketing method without

them necessarily being EA games. An example is BoxerJam, a pre-EA company that already had an agreement with AOL. We signed a deal with them that they would continue to provide BoxerJam, but engineering-wise we would figure out how to integrate them into our ticket mechanism. This allows a user to take a ticket and seamlessly move into Boxer Jam, even though it is a different company the ticketing method carries through.

EA grows by acquisition and is working on becoming a game publishing platform. We are trying to become something like the AOL of online gaming. Our deal with AOL is that we are the only way they can get games onto their site. It is not at all inconceivable that we could host other games on our site if they want to pay us, the same way we publish other games.

With an emulator title there is one emulator, called a ROM, somewhat like the cartridge on the old Genesis. Rather that using an authorize Servlet that issues a ticket, and putting it against the game that is to be played to request authorization, an emulator has to say not just here is the ticket, but here is the game to be played, because an emulator can start up a variety of games with the same ticket.

An emulator does not know in advance what type of game the user wants to play, whereas a game like Bunny Luv does. That means a little more information has to be passed into the game. It is possible that there would need to be different emulators based on authentication for various titles. The Genesis emulator is an ActiveX control or a Netscape plug-in depending on what browser is being used.

Our files are encrypted and get decrypted first inside the emulator to prevent hacks, and also to prevent people from learning how they are encrypted They are never stored on the client computer. This means that every time the game is started the game file has to be retrieved. We use a product form the company Marimba called Castinet. It started as a content server, and then became a push technology and now it is being used for patching applications.

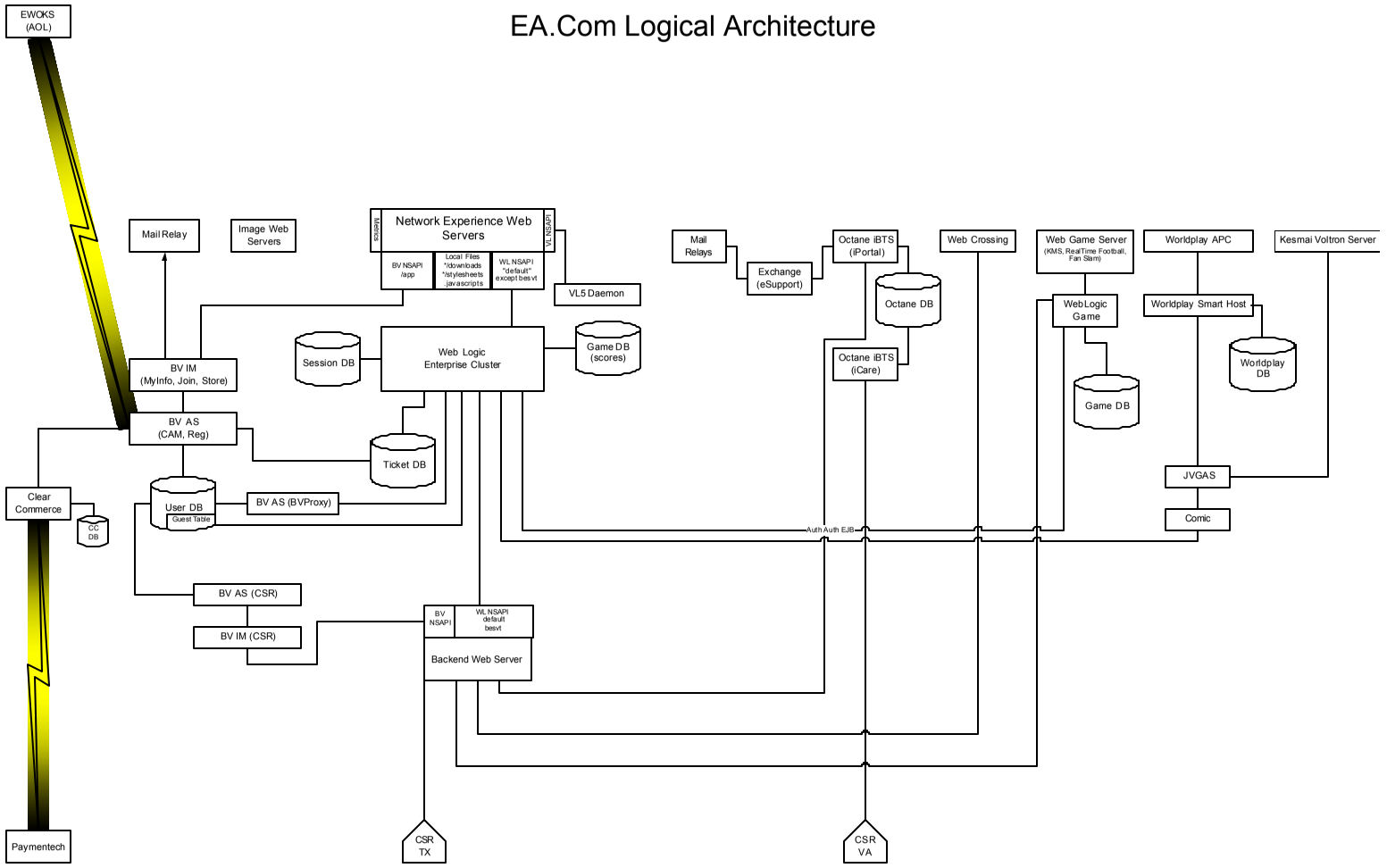
10.6 Matchmaking

Matchmaker is an application that matches two or more people together, for the purpose of playing a particular game on EA.com. The combination of the KMS (Kesmai Matchmaking System) backend and the Matchmaker client make up the EA.com Matchmaking System. Matchmaker provides basic player chat with a robust set of matchmaking features integrated into the EA.com Enterprise Infrastructure. The system features a standard matchmaking Graphical User Interface, complete with game lobbies, game rooms, game settings, and chat. Those Hi/Q titles that use KMS: FIFA 2001 and NBA Live 2001, are also using Matchmaker. the EA.com Matchmaking System also supports a number of retail box titles, including NHL 2001, NBA Live 2001 and FIFA 2001.

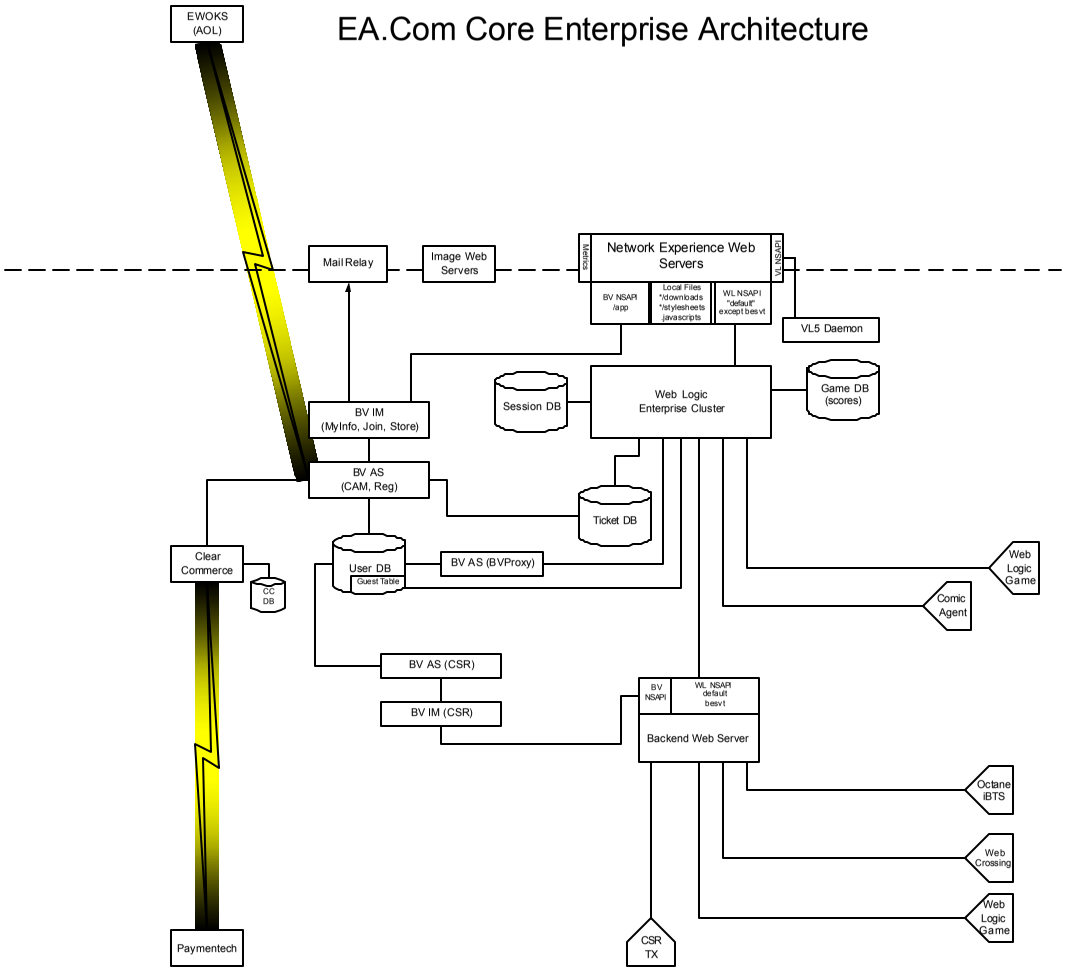
The user clicks a button that says match-up. In the background Snoopy and Sniffer checks that he has the CD in his drive, that he has KMS, which is an ActiveX control and is often included on the CD. It also checks that a user has the latest versions of whatever software. (Players are not allowed to play unless they have the latest versions of everything. Otherwise it would be a nightmare for customer support.) Finally it presents a list of Matchmaking lobbies and then the user gets passed off to the actual KMS client were the Matchmaking occurs. When the user goes to the KMS page a ticket gets passed on to the KMS client.

We have two Marimba servers for downloads: an EA.com Marimba as well as a package Marimba. Download.EA.com is EA and Trans.EA.com is for actual packaged goods. We do this because packaged goods are huge, and we do not want to clog up things, for example with the EA.com game play functionality, like the Genesis titles downloading the latest patch for NBA live. We try and minimize the impact on the customer load.

11. Visio: EA.com Logical Architecture

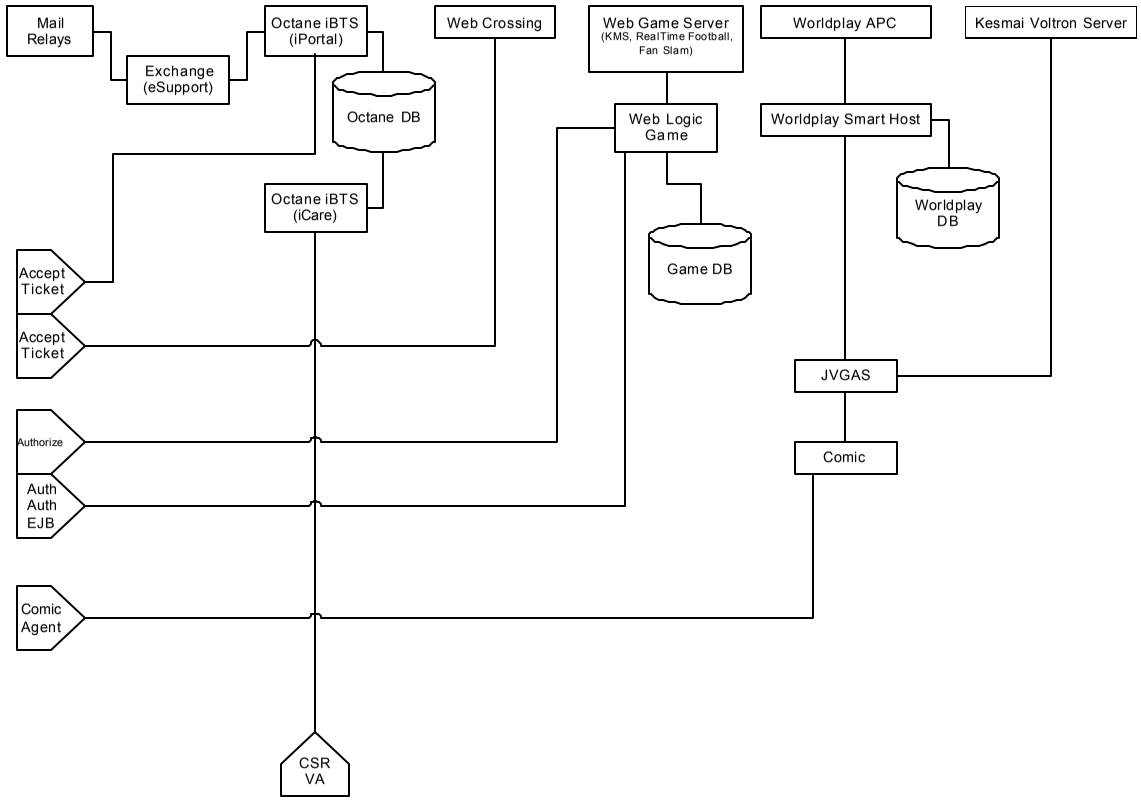


12. Visio: EA.com Core Enterprise Architecture



13. Visio: EA.com Ancillary Enterprise Architecture

EA.Com Ancilliary Enterprise Architecture



14. Ticketing Variations