

KAAZING WEBSOCKET GATEWAY

PRODUCT OVERVIEW

KAAZING WEBSOCKET GATEWAY

PRODUCT OVERVIEW

INTRODUCTION

Kaazing's flagship product, Kaazing WebSocket Gateway, lets businesses build, deploy and manage Living Web and mobile applications that can support a million concurrent clients and hundreds of thousands of messages per second:

- On any browser or mobile client;
- Using a standard enterprise protocol, custom protocol or HTML5;
- With enterprise-grade reliability and security;
- For one-tenth the cost and infrastructure complexity of competing solutions.

With Kaazing WebSocket Gateway, Rich Internet Applications running on any browser or mobile device can communicate, over the Web, with back-end TCP-based services across a reliable and secure, full-duplex, low latency connection. This makes every mobile device and browser (modern or not) a full-featured enterprise platform and every online application a first-class citizen of an enterprise system.

Kaazing eliminates the need to simulate real-time interactivity using long-polling, reverse AJAX, streaming, server push or any of the other techniques commonly included under the umbrella terms *Comet* and *AJAX*. While these approaches deserve much credit for enabling most of the dynamic content we see on the Web today, Comet and AJAX are nonetheless aging, costly and cumbersome workarounds that are only partially effective at addressing the central failing of HTTP; namely, its request/response model.

Organizations add Kaazing WebSocket Gateway to existing Web infrastructure to enable a whole new class of dynamic, interactive Living Web applications. The most forward-looking organizations go a step further and use Kaazing to eliminate their costly reliance on application servers and load balancers altogether. In either scenario, Kaazing WebSocket Gateway streamlines an organization's online IT architecture and delivers unprecedented improvements in total cost of ownership.

Kaazing WebSocket Gateway requires no browser plug-ins, works with any browser or mobile device, and integrates quickly and easily with back-end systems such as enterprise messaging platforms. Kaazing supports all major client-side technologies including JavaScript, Adobe Flex (Flash), Microsoft .NET/Silverlight, Java/JavaFX, iOS and Android.

Kaazing WebSocket Gateway is based entirely on Web standards including the HTML5 Communication specifications and WebSocket protocol. Together, these standards form the basis of all future communication involving dynamic data over the Web.

Today, companies in a wide range of industries are using Kaazing WebSocket Gateway to slash costs and complexity, reduce time to market for new applications, tap new sources of revenue, and deliver a superior online experience across Web and mobile platforms.

- In **Financial services**, Kaazing is delivering the extreme performance required by modern trading and capital market applications.
- In **Online gaming and auctions**, Kaazing is enabling ubiquitous presence along with support for huge numbers of concurrent users.
- In **E-commerce**, Kaazing is allowing retailers to capture and instantly leverage critical details of online shopping behavior.
- In **Social media**, Kaazing is powering rapid delivery of personal activity streams.
- In **Transportation and logistics**, Kaazing underlies a new generation of high value-add fleet monitoring and management solutions that rely on real-time telemetry and analysis.

THE CHALLENGES OF BUILDING REAL-TIME WEB APPLICATIONS

HTTP made the Web possible. But it was never intended to handle dynamic content.

In a legacy Web or mobile architecture, communication between the client and the application server is completely separate from communication between the application server and the back-end data source. The application server isn't just forwarding traffic – it is actively interpreting, translating and reformatting every message that passes between the front-end client and back-end server. That's because the back end doesn't understand HTTP while the front end understands *only* HTTP. This arrangement is efficient as long as the content being served is primarily static. But efficiency quickly degrades when the data is dynamic, communication is event-driven, and the volume of clients and messages increases. The speed and latency characteristics that make live data *live* can literally be lost in translation.

To sidestep the inherent inability of HTTP to support live data streams, application developers manually implement a Comet or AJAX workaround to simulate real-time interactivity. This approach imposes heavy costs on both the developer and the organization and deprives the end user of a genuinely live experience:

- **Applications scale poorly**, requiring additional hardware which increases cost and complexity of IT infrastructure.
- **Applications consume excessive amounts of network bandwidth**, further inflating infrastructure requirements and IT costs.
- **Applications incur increased latency**, damaging user experience and with it customer satisfaction, loyalty and retention as well as site stickiness and brand equity.
- **Developers are saddled with a complex programming task** that forces them, among other things, to design their own application protocols which must then be mapped onto standard API calls.

These problems with Comet or AJAX, and the costs they impose on all the stakeholders of an application, are caused by the inherent inefficiency of the workarounds themselves:

- HTTP headers contain hundreds or even thousands of bytes of metadata. Most interactive AJAX and Comet communication takes place in small batches, making the repeated metadata a significant percentage of each batch. This is why 90% or more of enterprise network bandwidth can be wasted.
- In low-message-rate situations with many concurrent users, the server is continuously processing empty-payload HTTP requests, wasting precious server resources.
- High-message-rate scenarios generate a continuous loop of immediate polls, again wasting valuable server and network resources.
- With secure connections (those using Transport Layer Security), the server must perform unnecessary encryption on the bytes describing the metadata of all those empty-payload request-responses, once more squandering valuable resources.

With the number of Living Web applications already exploding, the fatal flaws of HTTP, AJAX/Comet and legacy Web infrastructure vis-à-vis dynamic content have become unavoidable.

THE FUTURE ARRIVES

Fortunately, the Web standards bodies were aware of this impending dilemma years ago. The creation of the HTML5 standard inspired a complete reworking of the Web network stack to support true event-driven communications. Because the WebSocket and HTML5 communication standards underlie not only Kaazing WebSocket Gateway but all future Web communication involving dynamic data, it is useful to review the key elements of these standards.

WebSocket

Much of the network traffic generated by legacy Web technologies such as AJAX and Comet is metadata – that is, pure overhead. The WebSocket specification addresses this overhead by defining a full-duplex persistent connection between Web or mobile clients and back-end servers over which application data can flow in both directions simultaneously. Once established, this single-socket, very low overhead connection does not require any of the non-payload metadata that can dominate interactive legacy Web communication. It is not uncommon for multiple kilobytes of useless header data to be replaced by just 2-8 *bytes* of overhead when the same message is sent over WebSocket. The resulting reduction in network throughput can be dramatic when the volume of messages or number of connected users is high.

Server-Sent Events

Server-Sent Events (SSE) is a standard that allows servers to initiate data transmission to clients once an initial connection has been established at the client's request. SSE can be used to send message updates or continuous data streams and is well suited to broadcast-style notifications where a publish/subscribe paradigm would require too many active event streams over different HTTP connections.

Cross-Origin Resource Sharing

By allowing a client to incorporate information from many sources, Cross-Origin Resource Sharing (CORS) allows site aggregation to take place on the client side. The result is improved performance, reduced complexity and lower costs in an organization's Web infrastructure. WebSocket and Server-Sent Events are both cross-origin capable, so an application can communicate with a server over multiple channels, with real-time full-duplex message streams separated from other traffic. Cross-origin resource sharing was impossible to achieve in any standard, secure way prior to HTML5.

Cross-Document Messaging

For security and privacy reasons, Web browsers prevent documents from different domains from communicating with one another. While this prohibition against cross-site communication is an important security feature, it prevents pages from different domains from communicating even when they are not hostile. Cross-Document Messaging provides a straightforward, standard and secure way for documents to communicate with one another regardless of their source origin. Among its many applications, Cross-Document Messaging is useful where a portlet (the pluggable user interface software component), and not just the data, is delivered by a domain other than the source domain.

These capabilities make WebSocket and HTML5 Communication the ideal plumbing for Rich Internet Applications. Yet WebSocket and HTML5 are standards. To solve real-world problems, enterprises need a high-performance Web communication *platform* they can use to build solutions. The platform should also satisfy the following requirements:

- **Legacy browser support.** Older browsers including IE6/IE7 do not support WebSockets or HTML5 communication. Because enterprises cannot afford to short-change users based on their choice of browser, the platform must deliver seamless, high performance emulation when any of these features is not supported natively by the user's browser.
- **Proxy and firewall traversal.** Despite the best efforts of the standards bodies, WebSocket traffic is not guaranteed to traverse proxies or firewalls. To give all users a first-rate experience regardless of their physical location, the platform must ensure that WebSocket traffic tunnels through proxies and firewalls.
- **Support for business protocols and client-side technologies.** Many enterprise back-end systems communicate using higher level protocols such as JMS, AMQP, XMPP and a variety of custom formats. In order to extend these back-end systems out to the Web, their protocols must be transported over WebSocket. Additionally, developers must have an easy way to add WebSocket connectivity to their client applications regardless of which front-end technology they prefer to use.
- **Enterprise features.** An enterprise platform must offer levels of reliability and resiliency, performance and scalability, security and manageability that go well beyond the scope of any set of standards, including those for HTML5 Communication and WebSocket.

KAAZING WEBSOCKET GATEWAY

As detailed in the following sections, Kaazing WebSocket Gateway is a high performance Web communication platform that meets or exceeds all of the foregoing requirements. Kaazing extends back-end enterprise systems out to the Web by enabling clients and servers to communicate with any browser or mobile device over a full duplex, single socket connection with sub-millisecond added latency. Leveraging a robust, scalable architecture, Kaazing WebSocket Gateway can handle exceptionally high message volumes and huge numbers of concurrent users with enterprise-grade reliability and security.

Standards Compliance

Kaazing WebSocket Gateway is 100% standards based. It fully implements the W3C HTML5 Communication specifications and IETF WebSocket protocol, including Cross-origin Resource Sharing, Server-Sent Events, Cross-Document Messaging and the WebSocket API.

Support for Legacy Browsers and Web Infrastructure

It will be many years before legacy browsers are replaced by modern versions that natively support HTML5 and WebSocket. For users who remain on older browsers that lack native support for HTML5 or WebSocket, whether by choice or because of their employer's IS policy, Kaazing WebSocket Gateway provides seamless emulation with near-native performance, enabling a genuine Living Web experience for every user.

Kaazing also ensures that Gateway traffic will seamlessly traverse proxy servers and firewalls, even those that might have impeded open source WebSocket traffic. In each case, Kaazing's unique technology ensures native or near-native performance and the best possible connection regardless of whether browsers or intermediate devices support the latest protocols.

Transporting Higher Level Protocols

As revolutionary as it is, WebSocket – like any other socket – is an “empty pipe” when it comes to higher level protocols. In order to extend back-end enterprise systems out to the Web, the business protocols those systems use to communicate must be transported over WebSocket, a requirement not addressed by the specification.

Kaazing fulfills this need with product editions that offer out-of-the-box support for leading standard protocols such as JMS, AMQP and XMPP, and an HTML5 edition that developers can use to transport any other standard or custom protocol over WebSocket. Whatever the protocol, Kaazing's support includes all of the enterprise-grade features described in this document.

Broad Support for Client-side Technologies

Kaazing WebSocket Gateway ships with a client library that makes it easy to develop the client-side portion of JMS, AMQP or XMPP applications simply by including the library in the client application. This vastly reduces the time needed to prototype, build and deploy new services and saves developers from having to build their own enterprise-ready protocols for Web delivery. Support includes JavaScript, Adobe Flex (Flash), Microsoft .NET/Silverlight, Java/JavaFX, iOS and Android.

The ability to enable any connected client in any location to communicate over any higher-level protocol with any back-end service is what we call Universal Architecture. Because of its Universal Architecture, Kaazing WebSocket Gateway vastly simplifies and accelerates the task of prototyping, building and deploying globally scalable, cross-platform applications. So businesses can address shifts in customer behavior and attack new market opportunities with unprecedented speed.

Kaazing's Universal Architecture also enables a streamlined migration strategy. To move from Flash to HTML5, for example, one would first migrate the existing Flash client to Kaazing WebSocket Gateway, reaping significant scalability benefits in the process. One would then add an HTML5 client, further leveraging Kaazing's Universal Architecture, and optionally phase out the Flash client as HTML4 browsers became sufficiently scarce.

Scalability and Performance

Kaazing WebSocket Gateway scales easily to handle enormous numbers of concurrent users and massive throughput requirements with sub-millisecond added latency. Kaazing's unique ability to offload connection overhead and distribute load across a hierarchy of Gateways means that the platform scales architecturally rather than by brute force, allowing far higher capacity deployments than other approaches can deliver. Kaazing's patent-pending WebSocket Acceleration™ technology ensures that applications do not suffer performance degradation even when WebSocket or other advanced communications functionality is emulated.

Peer Load Balancing

Implemented entirely in software, Kaazing's unique peer load balancing capability eliminates the "choke point" bottleneck created by traditional load balancers when used to handle dynamic content. Not only does peer load balancing enable extreme scalability, it significantly reduces total cost of ownership and solution complexity by eliminating hardware load balancers from the architecture entirely.

Reliability and Resiliency

Reliability and resiliency are vital to any online business application, whether it's an e-gaming site where hundreds of thousands of concurrent users place real-time bets, or an online financial platform where a few thousand institutional players engage in high-frequency trading worth billions of dollars. Whatever the application, Kaazing WebSocket Gateway guarantees reliability and resiliency at every level, from individual messages to the organization as a whole. The integrity of individual communications is provided by end-to-end auto reconnect and, where the protocol supports it, guaranteed message delivery. At the organizational level, built-in high availability (HA) via clustering as well as seamless disaster recovery (DR) ensures business continuity with little or no incremental expense or complexity.

Security

Kaazing WebSocket Gateway was built from the ground up for environments with the most stringent security requirements. Kaazing supports standard TLS/SSL encryption for data protection over the wire. It offers basic as well as token-based authentication and supports fine-grained authorization that let management designate permitted operations by user or by role.

Unique to Kaazing WebSocket Gateway is its support for SPNEGO-based Kerberos security across a WebSocket connection. Kaazing users can integrate Kerberos with their existing infrastructure to provide Single Sign-On (SSO) capability over the Web, or they can use the token-based authentication to integrate with any third party or custom SSO framework.

Another innovation unique to Kaazing is automatic revalidation, a WebSocket extension which ensures – without introducing any service interruption – that WebSocket connections do not outlive their authorizing credentials. Kaazing is also alone in allowing customers to set a strict upper limit on connection time, after which the WebSocket connection is guaranteed to close. Although the time limit can be set sufficiently higher never to interfere with an actual application, its presence allows security experts in the enterprise to better model the system's overall security profile.

Binary Support

The Kaazing WebSocket Gateway supports binary protocols enabling raw TCP communication between client and server, or between two servers. While the WebSocket protocol specification also supports binary data, the Web standards bodies have yet to formalize an API for sending and receiving it. Until a standard becomes available, the Kaazing Gateway offers developers an API for sending true binary data over the web. In addition, since each browser vendor has its own preferred representation of binary data, the Kaazing library abstracts the difference for a variety of client technologies including Flash and Silverlight. This advantage makes binary communication work everywhere, with any legacy or modern browser.

Virtual Private Connection

Many people think of WebSocket as only a client-server technology, but it is much more than that. The full potential of WebSocket also includes enabling server-to-server, device-to-device, intra-cloud and cloud-to-cloud communication over the Web. It is easy and inexpensive to create a high-performance, low latency, full duplex Virtual Private Connection (VPC) between any two servers by positioning a Kaazing WebSocket Gateway in front of each one. (VPC is currently available with the JMS Edition, but broader support is forthcoming.) Although pairs of servers can already establish a TCP socket connection, only Kaazing WebSocket Gateway permits traffic across that socket to traverse proxies and firewalls – that is, to traverse the entire Web – seamlessly.

Manageability

Kaazing WebSocket Gateway exposes a set of Java Management eXtension (JMX) MBeans that allow administrators to manage and monitor the internal state of the Gateway. Management tools compatible with these protocols recognize Kaazing WebSocket Gateway and connected Gateway clients as part of their management topography. This lets IT teams use existing management tools to oversee both LAN-based and mobile/browser-attached users

SUMMARY

Until now, it has been impractical to establish full-duplex connectivity between servers, clients or devices over the legacy Web. At best, one could simulate real-time interactivity using protocol trickery and inefficient communication models. This limitation has impeded advances in the development of Rich Internet Applications.

HTML5 communication standards and the WebSocket specification provide the building blocks to address this fundamental failing of the legacy Web and to usher in the era of the Living Web. But standards and specifications are not actionable technologies, and certainly not enterprise-ready solutions. To realize the full promise of the Living Web, Kaazing WebSocket Gateway was created.

Kaazing WebSocket Gateway makes it easy to build JavaScript, Java/JavaFX, Adobe Flex (Flash), Microsoft .NET/Silverlight, iOS and Android applications that communicate directly with back-end services over standard enterprise protocols such as JMS, AMQP and XMPP, or any custom protocol, or native TCP. By extending back-end enterprise systems out to Web and mobile clients, Kaazing WebSocket Gateway radically simplifies the convoluted J2EE architectures that currently dominate legacy Web infrastructure, eliminating the need for application servers, load balancers and the collection of software tricks currently used to simulate real-time interactivity. Both IT costs and the time required to build and deploy dynamic, globally scalable applications are slashed.

For the first time in history, browser-based and mobile applications have become first-class citizens of network communications, a benefit long enjoyed only by desktop applications. Kaazing WebSocket Gateway scales easily to support massive numbers of concurrent users and huge message volumes. Most importantly for organizations seeking a ubiquitous online presence, Kaazing supports every browser (modern or not), mobile client or other connected device, regardless of location or network topology. Kaazing tunnels through proxies and firewalls, so no client is ever out of reach. Moreover, if a legacy browser or other client device does not support WebSockets or HTML5, Kaazing's patent-pending WebSocket Acceleration technology emulates the full suite of next-generation communication functionality – WebSockets, Server Sent Events, Cross-Origin Resource Sharing and Cross-Document Messaging – with near-native performance.

Kaazing WebSocket Gateway is a robust enterprise-grade platform with a broad suite of features and capabilities that make every Kaazing-powered application reliable, resilient, secure and easily manageable.

To learn more about Kaazing WebSocket Gateway, visit us at www.kaazing.com.