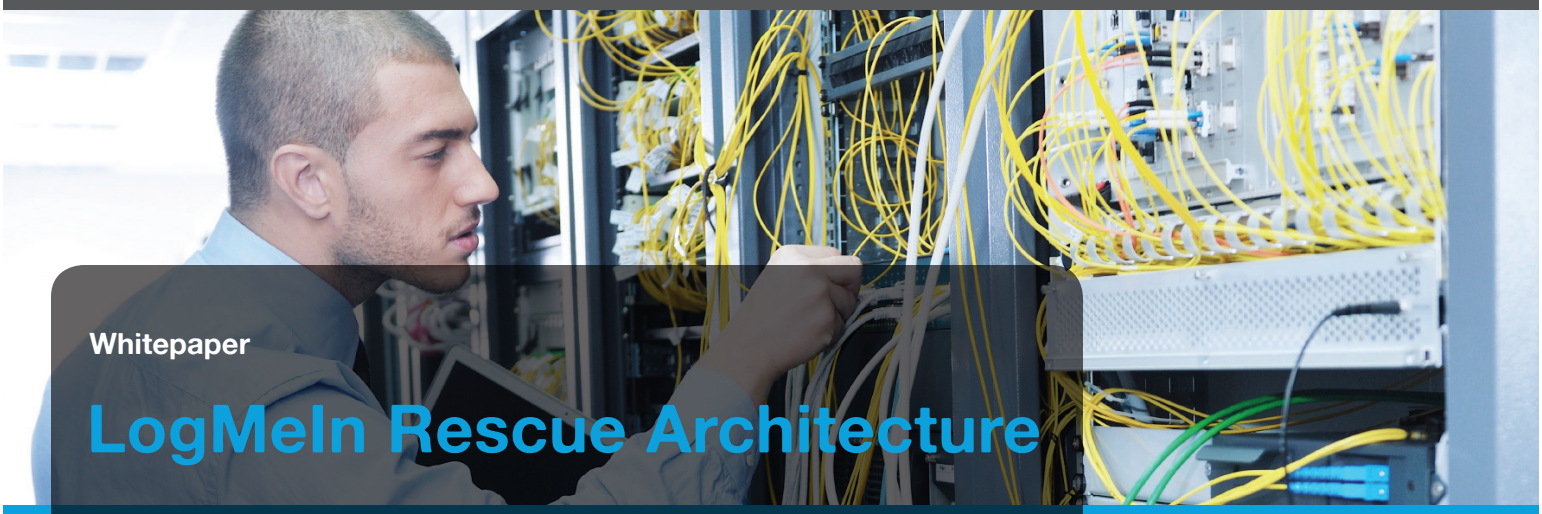


Whitepaper

LogMeIn Rescue Architecture

A technical overview of Rescue's architecture.

Rescue by LogMeIn

Rescue architecture

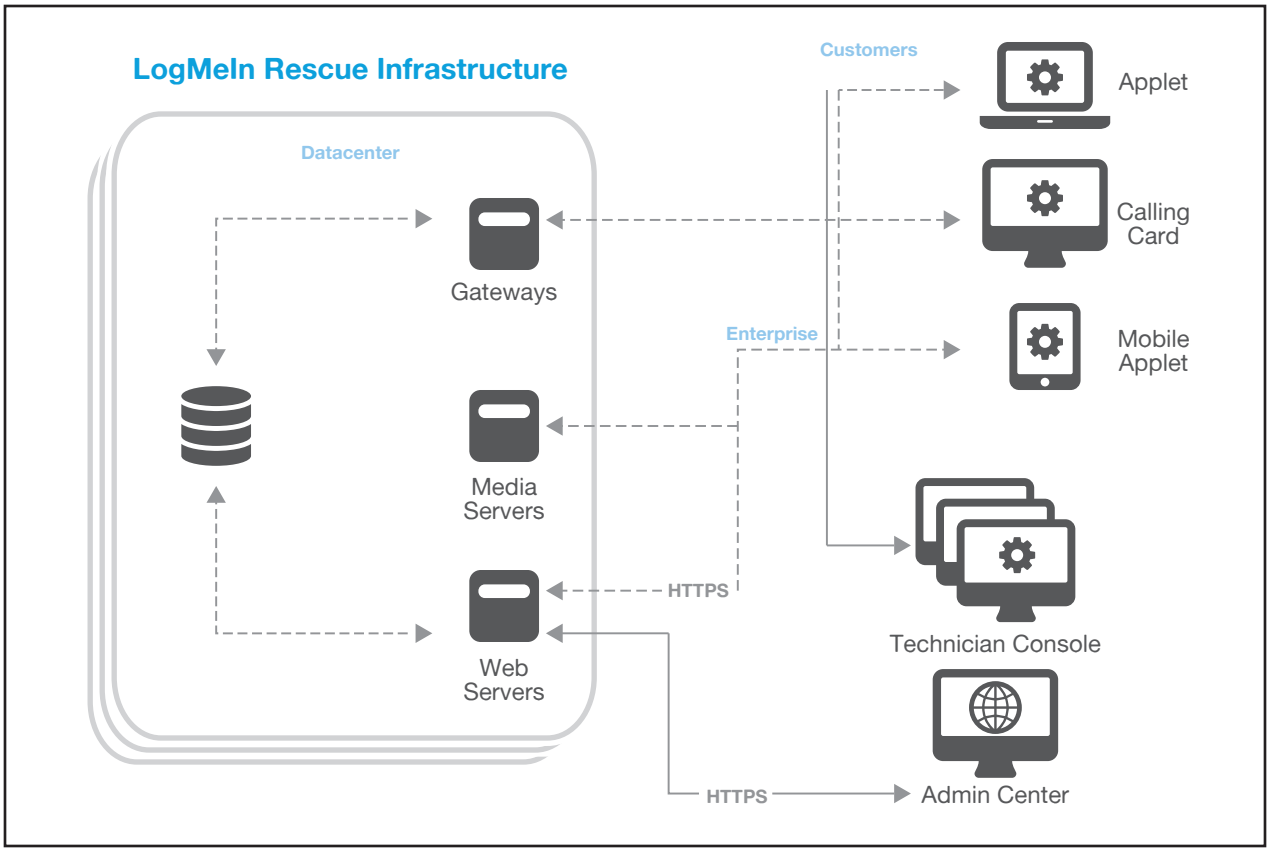
Introduction

Scalability, security, reliability, ease of use. These four characteristics are what describe a great remote support solution, but they do not always go hand-in-hand. It's easy to find a remote support solution that provides two or maybe three of the above criteria, but a solution that delivers all four is rare. LogMeIn, Inc. provides just such a solution with LogMeIn Rescue.

Scalability. Whether you have a single technician or a call center with ten thousand employees, Rescue gets the job done.

Security. Support sessions are protected with end-to-end 256-bit AES encryption. Support operations must be permitted by the end user before the technician can perform them. Support session logs are stored in a database in encrypted format and can be queried later. Remote control sessions can be recorded to a video file.

Reliability. Rescue is hosted in five carrier-grade datacenters with a fully redundant infrastructure.
Ease of use. Your technicians will be up and running in a matter of hours. Your supported end users will get help with a few clicks. No software has to be installed by either party.



Data Confidentiality

Often security is equated to data confidentiality, data confidentiality is equated to encryption, and encryption is characterized by the symmetric cipher used and its key length. These misconceptions lead to misnomers such as “256-bit AES secure.” Needless to say, this is misleading.
A secure online system should always meet the following objectives:

- Authentication of the communicating parties
- Negotiation of encryption keys without a man-in-the-middle intercepting them
- Exchange messages confidentially
- Detect if a message has been modified in transit

SSL/TLS, short for Secure Sockets Layer and Transport Layer Security, has been designed to provide support for the above steps. It is the de-facto standard for secure communications over the Internet, and has been endorsed by Visa, MasterCard and American Express.

The TLS implementation used by LogMeIn Rescue is OpenSSL (http://www.openssl.org). LogMeIn always uses the latest available version. At the time of publication, the version employed by Rescue is 1.0.2d.

Key Agreement

When a support session starts and a connection is established between the supported user and the technician, their computers must agree on an encryption algorithm and a corresponding key to be used for the duration of the session. The importance of this step is often overlooked, and this is somewhat understandable: it seems like a mundane task that should be simple and straightforward.

It is, however, anything but simple: Certificates must be employed to counter so-called man-in-the-middle attacks (where computer C would position itself between computer A and B and impersonate the other party to both A and B). Since neither the technician nor the end user have server software and an SSL certificate installed on their computers, they both turn to one of the LogMeIn Rescue servers to perform the initial phase of the key agreement. Verification of the certificate by both the Technician Console and the end user applet ensures that only a Rescue server can mediate the process.

Message Exchange

TLS allows for a wide range of cipher suites to be used. Communicating parties can agree on an encryption scheme they both support. This has two primary purposes: first, the protocol can be extended with new cipher suites without breaking backwards compatibility, and second, newer implementations can drop support for suites that are known to contain cryptographical weaknesses.

Since all three components of the LogMeIn Rescue communications system are under LogMeIn's control, the cipher suite used by these components is always the same: AES256-SHA in cipher-block chaining mode with RSA key agreement. This means the following:

- The encryption keys are exchanged using RSA private/public key pairs, as described in the previous section
- AES, short for Advanced Encryption Standard, is used as the encryption/decryption algorithm
- The encryption key is 256 bits long
- SHA-2 is used as the basis of message authentication codes (MACs). A MAC is a short piece of information used to authenticate a message. The MAC value protects both a message's integrity as well as its authenticity, by allowing the communicating parties to detect any changes to the message.
- Cipher-block chaining (CBC) mode ensures that each ciphertext block is dependent on the plaintext blocks up to that point, and that similar messages cannot be distinguished on the network.

The above ensures that data traveling between the supported end user and the technician are encrypted end-to-end, and only the respective parties have access to the information contained within the message stream.

Authentication and Authorization

Authentication and authorization in LogMeIn Rescue serves two distinct purposes.

Authentication ensures that the technician or administrator logging in to the Rescue system is in fact who he claims to be. In Rescue, authentication is handled in a very straightforward manner: Technicians are assigned login IDs (usually matching their email addresses) and corresponding passwords by their administrators. These credentials are entered into the Login form on the LogMeIn Rescue website at the start of a technician workday.

In LogMeIn Rescue, the Rescue system is first authenticated to the technician (or rather, the technician's web browser) with its 2048-bit premium RSA SSL certificate. This ensures that the technician will be entering his username/password into the right website. The technician then logs in to the system with his credentials.

LogMeIn Rescue gives administrators a number of options for password policy:

- Administrators can enforce a minimum required password strength and a maximum password age – a built-in meter shows administrators and technicians the strength of the chosen password
- Technicians can be forced to change their Rescue password upon their next login

LogMeIn Rescue also allows Administrators to implement a Single Sign-On (SSO) policy. The Security Assertion Markup Language (SAML) is employed and is an XML standard for exchanging authentication and authorization data between security domains, that is, between an identity provider and a service provider. Technicians then have access only to pre-defined applications and a single SSO ID to log in to those applications. At the flick of a switch, a technician's SSO ID can be disabled.

Authorization, on the other hand, happens very frequently – at least once during every remote support session. The supported end user, after downloading and running the support applet, will be contacted by a technician. The technician can chat with the end user via the applet, but any further action, such as sending a file or viewing the end user's desktop, requires express permission from the user. A "single prompt" can also be implemented. This is intended for lengthy remote support work where the customer might not be present for the entire duration of the session. If this flag is enabled for a Technician Group, then the technicians in that group can request a "global" permission from the customer, and, if granted, will be able to perform actions such as viewing system information or entering a remote control session without being further authorized by the end user.

Administrators can also impose IP address restrictions on their technicians. When selected, the IP addresses available can be restricted to a very narrow list. Technicians assigned to a particular task can then only access Rescue from pre-approved IP addresses for that task.

The administrator of a Technician Group can also disable certain features in the Administration Center. For example, members of a Technician Group can be prevented from receiving files from end users. Here are some of the permissions an Administrator can grant or deny:

- Launch remote control
- Reboot
- Launch Desktop Viewing
- Record sessions
- Send and receive files
- Start private sessions
- Launch File Manager
- Request Windows credentials
- Send URLs
- Allow clipboard synchronization
- View system information
- Deploy scripts
- Use single prompt for all permissions
- Transfer sessions
- Allow screen sharing with customers

The Rescue system is also authenticated to the supported end user. The applet, downloaded and run by the user is signed with LogMeIn's code signing certificate (based on a 2048-bit RSA key), and this information is typically displayed to the user by their web browser when they are about to run the software.

The supported user is not authenticated. It is up to the technician to determine who the user is, either via chat or a telephone conversation. The Rescue system does provide authentication-like mechanisms such as unique PIN codes, but these are used for routing the support session to the correct private or shared queue, and should not be construed as an authentication system.

Auditing and Logging

Any remote support solution must place strong emphasis on accountability. LogMeIn Rescue provides two distinct auditing features.

First, the so-called "Chat log" is saved in the Rescue database. The "Chat log" is transmitted to the Rescue servers by the Technician Console in real time, and contains events as well as chat messages that pertain to a particular support session. For example, a log file would display when a remote control session is started or ended, or when a file is sent by the technician to the end user. Accompanying metadata, such as the name and MD5 Hash thumbprint of a transmitted file, is also included in the log when applicable. The "Chat log" database can be queried from the Administration center. At the time of writing, LogMeIn's data retention policies stipulate that the contents of the logs will be made available online for two years after the end of a remote support session and archived for two years after that. To facilitate integration with CRM systems, LogMeIn Rescue can post session details to a URL. Administrators can choose whether to allow chat text to be excluded from these details. Additionally, all records of chat texts between technicians and clients can automatically be omitted from the session details stored at the Rescue Data Center.

Second, LogMeIn Rescue allows the technicians to record the events that transpire during a desktop viewing or remote control session into a video file. This is a very important feature for accountability and liability reasons. The recording files are stored in a directory specified by the technician. In the case of a large support organization, this location should be on a network server. The disk space taken up by these recordings varies widely, and depends entirely on the contents (and compressibility) of the supported end user's desktop, but based on an analysis of millions of remote control sessions utilizing LogMeIn's technology, the average disk space requirement for one minute of remote control data is between 372 and 1024 Kbytes. The recordings are stored direct to AVI or in an intermediate LogMeIn proprietary format that can be converted to standard AVI files by the "Rescue AVI Converter" application downloadable from the Support section of the LogMeIn Rescue website. The LogMeIn proprietary format, called RCREC, can cut recording size by about 10%.

Data Center Architecture

LogMeIn Rescue is hosted in state-of-the-art, secure data centers with the following features:

- Multi-layer security control procedures, biometric entry systems, and 24/7 closed-circuit video and alarm monitoring
- Uninterruptible redundant AC and DC power, onsite backup power generators
- HVAC redundant design with air distribution under raised flooring for maximum temperature control
- Smoke detection system above and below raised floor; double-interlock, pre-action, dry-pipe fire suppression

The LogMeIn Rescue infrastructure itself is highly secure and reliable:

- Redundancy on the server component level: redundant power supplies and fans, RAID-1 mirrored harddisks
- Redundancy on the server level: depending on role, active/passive or active/active clusters
- Redundancy on the datacenter level: Five datacenters (US West Coast, US Central, US South-Central, US East Coast and London, UK) with near-instant failover capabilities
- Dual redundant firewalls with only ports 80 and 443 open
- Active/passive database clusters
- Redundant load balancers including SSL
- Load-balanced and redundant web and application server clusters
- Load-balanced and redundant gateway server clusters

LogMeIn Rescue HIPAA Considerations

Although LogMeIn cannot control the content shared by users during a support session, the LogMeIn Rescue service is designed to meet strict security standards, which allows HIPAA regulated entities to meet regulatory guidelines set forth by HIPAA.

Access Controls

- Define permission-based access on a granular level (such as permitting some technicians with remote view only, but not remote control; or some technicians with no file transfer rights)
- No data from remote PCs are stored on LogMeIn data center servers (only session and chat data are stored). In addition, chat text logs can be removed from session details.
- Permissions can be set so that Technicians do not have file transfer rights, eliminating their ability to take files from remote PCs.
- End user must be present at the remote machine, and permit remote access
- End user maintains control, and can terminate the session at any time
- Permissions can be set so that end user must explicitly allow a technician to use specific functions (remote control, desktop view, file transfer, system information, and reboot & reconnect)
- Access rights are automatically revoked when session is terminated
- Predetermined time of inactivity forces automatic logoff
- Hosted at redundant leading, carrier-grade data centers with restricted, secured access

Audit Controls

- Option for forced session recording, with ability to store audit files on secure network share
- Technician sessions and remote session activity is logged on the host computer to ensure security and maintain quality control (successful logins, unsuccessful logins, remote control started, remote control ended, reboot initiated, logout)
- Person or entity authentication
- The technician's identity is defined by a unique email address, or via an SSO ID, and the technician must be authenticated
- Excessive number of unsuccessful login attempts (five unsuccessful attempts) will lock the account
- IP address restrictions limit the access technicians have to only those specified.

Transmission Security

- End-to-end 256-bit AES encryption of all data
- MD5 Hash for enhanced traceability of file transfers

An Overview of the LogMeIn Rescue Gateway Hand-off Process

When the digitally signed Rescue applet is started on a machine:

- It contains a session authentication GUID (Globally Unique Identifier), which has been embedded in the .exe file as a resource by the site when it was downloaded
- It then downloads a list of available gateways from secure.logmeinrescue.com
- It picks a gateway from the list and connects to it using TLS ; the gateway is authenticated by the applet using its SSl Certificate
- The gateway authenticates the applet in the database with the GUID and registers that the user is waiting for a technician

When a session is picked up in the Rescue Technician Console:

- A request is sent to the gateway with the session authentication GUID to relay connections between the Technician Console and the client applet
- The gateway authenticates the connection and starts relaying data at the transport level (it does not decrypt relayed data)

When a connection relay is started, the parties try to establish a peer-to-peer (P2P) connection:

- The applet starts listening for a TCP connection on a port assigned by Windows
- If the TCP connection cannot be established within a time limit (10 seconds), an attempt is made to establish a UDP connection with the help of the gateway
- If either a TCP or a UDP connection is established, the parties authenticate the P2P channel (using the session authentication GUID), and it takes over traffic from the relayed connection
- If a UDP connection has been set up, TCP is emulated on top of the UDP datagrams using XTCP, a LogMeIn-proprietary protocol based on the BSD TCP stack

Every connection is secured with the TLS protocol (using AES256 encryption with SHA256 MAC). The Session Authentication GUID is a 128-bit, cryptographically-random integer value.

Rescue Media Architecture

The Rescue Media Service (also called Video Service, Lens Media, or just Media) is a WebRTC based, separated, standalone service that enables video streaming for the Rescue Lens feature. Its main purpose is to manage so called “conferences” (essentially Rescue Lens sessions within Rescue sessions).

Components

There are three main components of the media service: the MediaSDK, the Session Manager and the Streaming Server. These components open/close conferences and manage peers joining/leaving conferences.

The Service was built on the top of WebRTC, so it requires a signaling layer. The signaling layer uses Rescue’s current communication channels between the Technician Console and the website and between the Lens App and the Website.

MediaSDK (C++, Html5)

The Media Service was built on top of the WebRTC open-source project. The MediaSDK is a thin library that hides most of the WebRTC’s features but expresses only those functions and objects on its interface that are meaningful for peers. This MediaSDK must be used by peers in their code.

The code itself is written in C++ and is compiled for Windows, MacOS, iOS and Android. For Android there is also a Java interface which is a wrapper on top of the C++ interface.

There is a Html5 SDK also which has nothing to do with the C++ code. It has a similar interface but 100% different implementation. This is SDK is used by BoldChat’s Video service.

Signaling

By design the WebRTC has no restriction nor implementation about how peers are exchanging the so called SDP messages. These are text based messages that describe the “state” of the conference. For example which peers are still in the conference, which peers are sending/receiving what kind of streams. The Rescue Media Service is not implementing any signaling layer but it has an interface (INetwork) in the MediaSDK which must be implemented by the peers itself.
In Rescue Lens the existing Rescue websites are used to exchange the SDP messages between the peers. This signaling layer is shown red in the figure below.

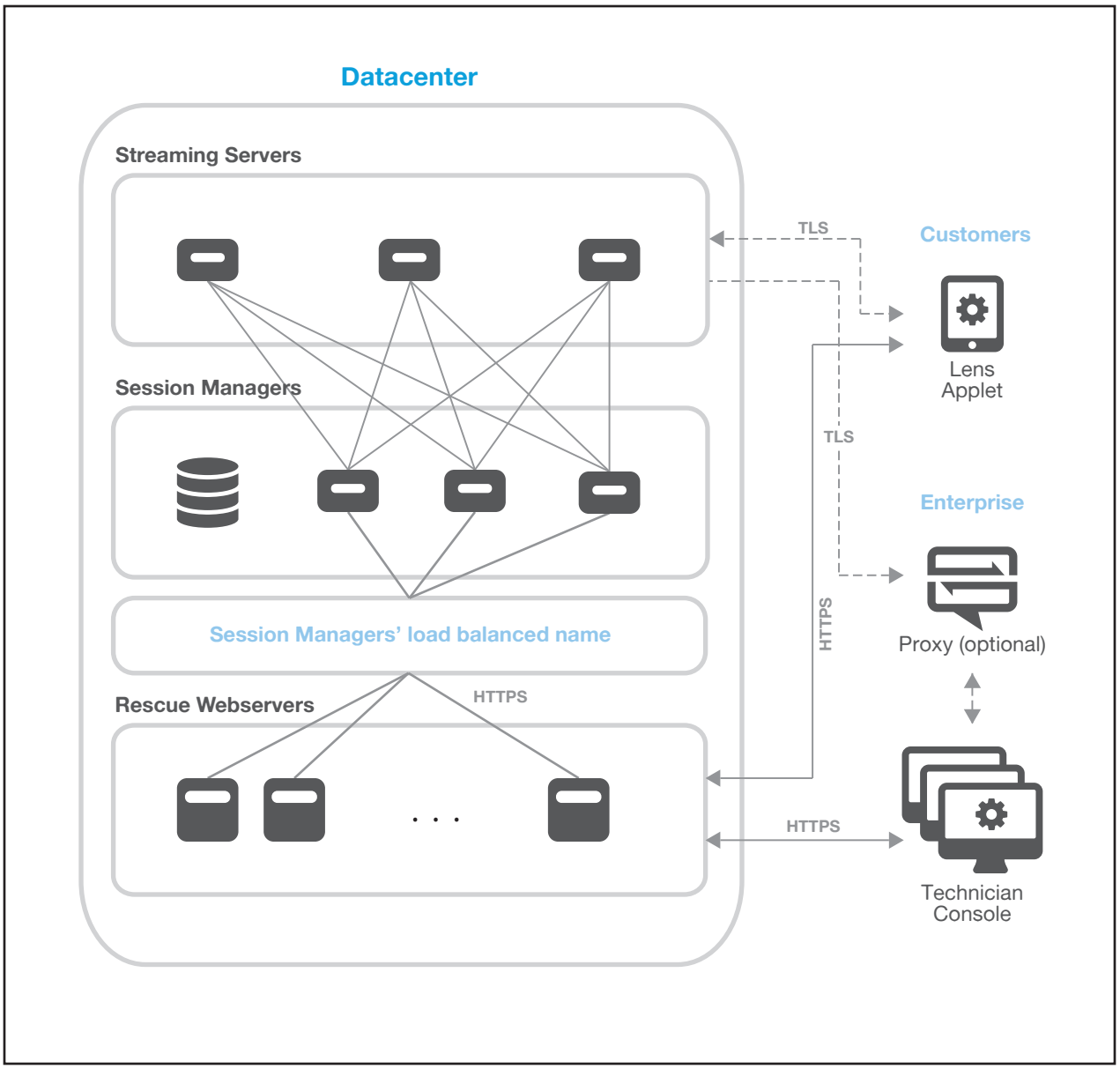
Streaming Servers (Jitsi)

The Jitsi open source project is used to manage conferences. Peers are connected to each other directly, but they send their audio/video stream to Jitsi. Other peers receive content from the Jitsi. Jitsi works like a relay server between the peers.

Session Managers

The Session Manager is a .Net based website which main purpose is to hide the Jitsi interface and expose some API functions to manage conferences.

With these API calls the peers can create/join/leave a conference. The session manager is using an AppFabric NoSQL database to store info about the running sessions (these connections are not shown in the figure).



Lens Session Flow

1. The technician creates a Lens session.
   a. The session is registered in Rescue's main database; at this point the Media Service is not affected in any way.
   b. Technician shares the Lens session's PIN with the customer.

2. The customer enters the PIN code in the Lens app.
   a. The Lens app communicates with the website and refers to the Lens session created by technician above.
   b. The Lens session becomes visible on the technician's screen.

3. Technician starts the session.
   a. The Technician Console asks for a new session from the Session Managers through the website.
   b. The Session Manager relays the request to one of the Jitsis, where the conference is created. Jitsi returns an "offer" SDP which must be delivered to customer's app through the signaling layer.
   c. The Session Manager performs some changes in the SDP and returns the modified "offer" SDP to the website.
   d. The website returns the SDP as the result to the Technician Console.
   e. Lens receives a message from the website that it should apply the "offer" SDP.

4. The Lens app connects to the Jitsi
   a. Lens app is feeding the MediaSDK with the returned "offer" SDP.
   b. The MediaSDK starts working and generates an "answer" SDP which is sent to the Jitsi through the Lens' website connection (signaling).
   c. The MediaSDK connects to the Jitsi on its external interface. The external interface's address was mentioned in the "offer" SDP.
   d. After successful connection to the Jitsi, the MediaSDK starts sending the video stream.

5. The Technician Console also connects to the Jitsi (at the same time as the app)
   a. The Technician Console feeds the MediaSDK with the returned "offer" SDP.
   b. The MediaSDK starts working and generates an "answer" SDP, which is sent to the Jitsi through the same signaling as before.
   c. The MediaSDK connects to the Jitsi on its external interface. The external interface's address was mentioned in the "offer" SDP.
   d. After successful connection to the Jitsi, the MediaSDK starts listening for the incoming video stream.

Conclusion

Choosing a remote support solution is often a decision based on features and pricing. If you are reading this document, then it is likely that LogMeIn Rescue has met your needs in these categories. With the information set forth above, we believe we were able to prove that the architecture behind Rescue provides the right levels of scalability, security, reliability and ease of use.