HOW MIDDLEWARE WORKS

Middleware first gained popularity in the 1980s. Deploying middleware then allowed new applications and services to access older legacy back-end systems. This allowed developers to ensure that new interfaces are capable of receiving data from older back-end technology.

Sometimes referred to as "software glue", middleware continues to be used today, as software that mediates between two separate pieces of software. It underpins the architecture philosophy on the backend and allows an operating system to communicate with applications.

Today's middleware is less about connecting to legacy systems and more about overall access and communication. Let's take a look.

What is middleware?

Middleware is software that acts as an intermediate between the backend and the front end. It is a runner between the two platforms that allow users to access otherwise inaccessible functionality. For example:

- Middleware may run between a Windows machine and a Linux back-end.
- Middleware could be the piece that allows enterprise employees to use remote applications.

Middleware can be easily understood as the "to" in phrases like peer-to-peer—it is middleware software that enables the data to travel.

Middleware is also important for application development and delivery. By understand the operating systems that a company uses to underpin its operations, a software developer can account for the way in which an application will be distributed.

How middleware works

Allowing applications to communicate through middleware means greater longevity of the operating system architecture, thanks to integration middleware aiding with communication.

Various types of middleware are necessary for performing the functions we expect of the applications that we use in business and personal settings. In fact, they are far more common than most people realize.

For example, Android users rely on middleware every time they use any phone application. The Android operating system is built on a modified Linux kernel, so developers need to build the application with the need to communicate with Linux in mind.

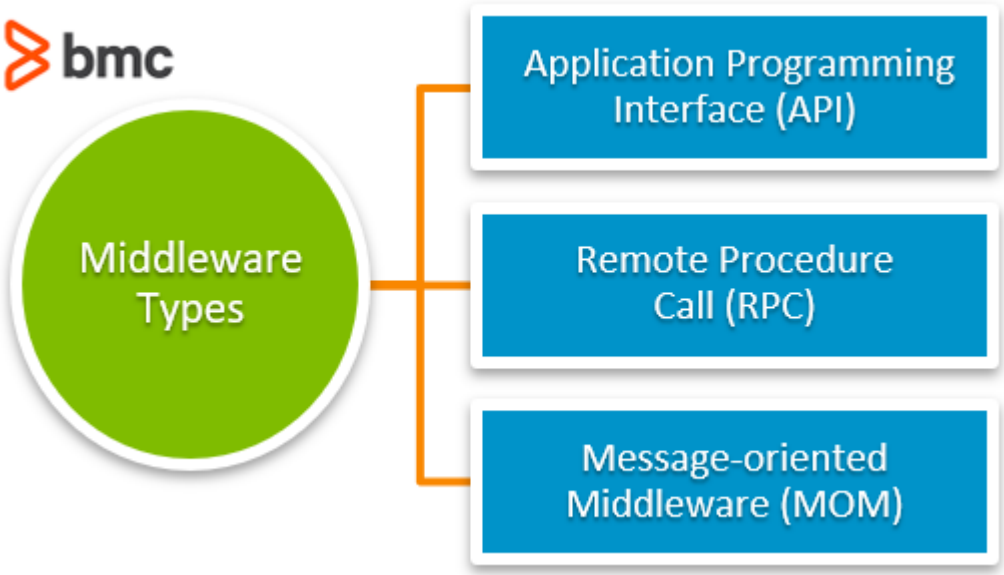
Linux needs to communicate with an app, but it needs to use the Android OS as middleware in order to successfully do so. As you can see, the request from the application needs to communicate with the back end. In order to do that, the Android operating system will:

1. Send the request back.
2. Receive the data in response.
3. Transform the data for the application.

Where is middleware useful?

Everywhere! Middleware is useful absolutely everywhere, but especially in enterprise settings.

Middleware and middleware developers support application development by allowing back-ends to become front-end agnostic, in some cases. Middleware enables operating systems of all kinds to be linked up to all kinds of front-end applications, including everything from web servers to database middleware.



Types of middleware

Middleware is not one type of software. Middleware comes in many forms and each type has a different use case which is useful to improving productivity and accessibility to apps.

In fact, there are many broad types of middleware that each serve different purposes. Depending on a business's needs, they may only need a type of middleware that will lock the client machine when it is accessing remote programs. Other enterprises might need to concurrently use both local and remote functionality, meaning that they need a different type.

Here are some examples of middleware, including examples you will have encountered in your day-to-day life.

Application programming interface (API)

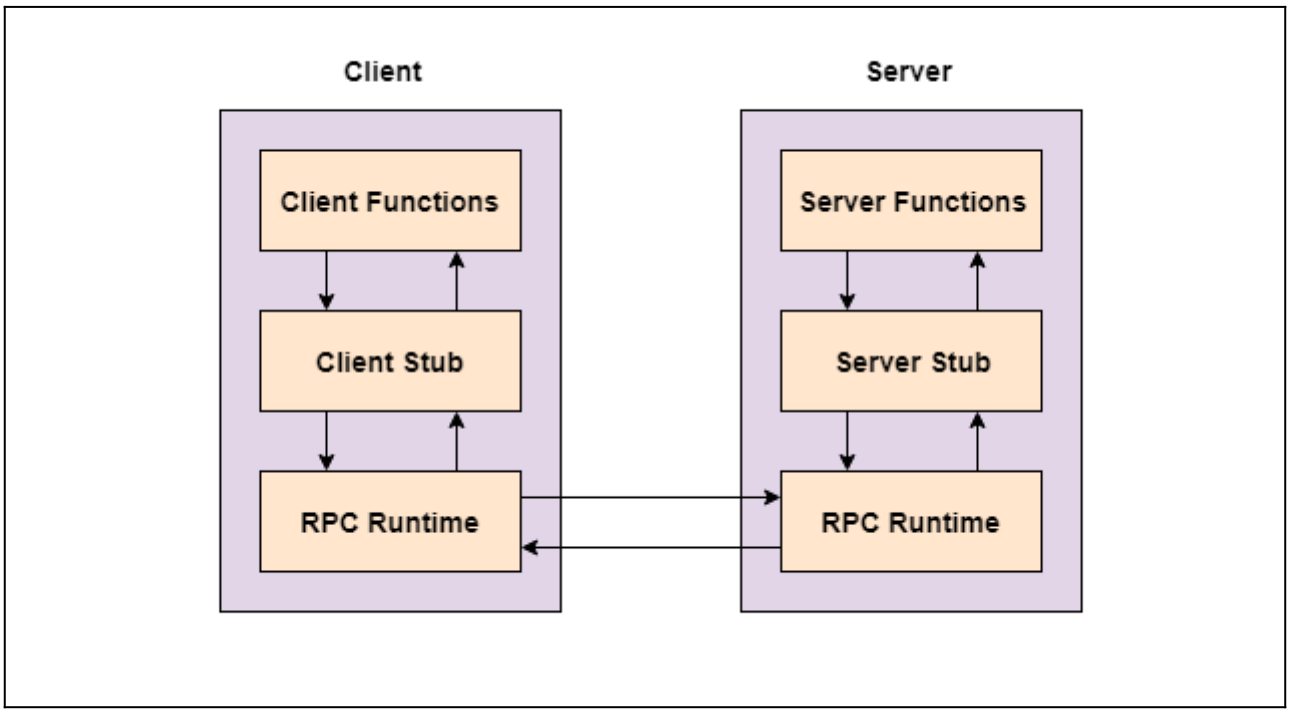
An API is middleware software that communicates information between a front end and a back end. Technically, APIs aren't middleware, but they serve a similar purpose.

A common area that people will find APIs is in headless platforms—the back-end only communicates with the APIs, which then run the data to the front-end. This allows web services to be completely customizable, instead of being locked into a rigid, monolithic provision.

(Learn how to build API portals devs will love.)

Remote procedure call (RPC)

When working with distributed computing setups, an enterprise application might rely on a specific kind of middleware called Remote Procedure Call (RPC).



RPC in action (Source)

When a computer program starts a procedure on a remote machine, it is beginning a client-to-server process. In order to do this, the local computer starts to use middleware to allow the server to perform the procedure.

This allows an end-user to perform remote produce as if it was local and to make the most of distributed systems and any enterprise application that the user needs. Without RPC, it would be difficult to run thin-client machines.

Message-oriented middleware (MOM)

MOM is similar to RPC in that it allows users to take advantage of distributed systems. Intended for applications that span multiple operating systems, it takes the messages that are outputted by software components such as applications and allows for platform-agnostic communication.

Whereas RPC needs the called procedure to be returned in order for the client to begin working again, MOM allows for loose coupling of components. A message broker (or the middleware) provides translation services back to the end-user without the need to lock the systems together. Through messaging, the middleware platform translates information back to the end-user.

Middleware is necessary

Whether you are working in cloud computing or in another area that requires distributed applications, middleware software provides a range of tools to developers for creating application servers and other tools that are useful in an enterprise environment.

Applications can work together with a range of back-end technology to ensure that having a diverse computer environment does not hold back an organization's ability to access enterprise applications.

In fact, middleware provides a wide range of uses to any business.

Related reading

- BMC DevOps Blog
- What Is Pub/Sub? Publish/Subscribe Messaging Explained
- What Is a Service Mesh?
- Microservices vs APIs: What's The Difference?
- Application Mapping: Concepts & Best Practices for the Enterprise
- OSI Model: The 7 Layers of Network Architecture