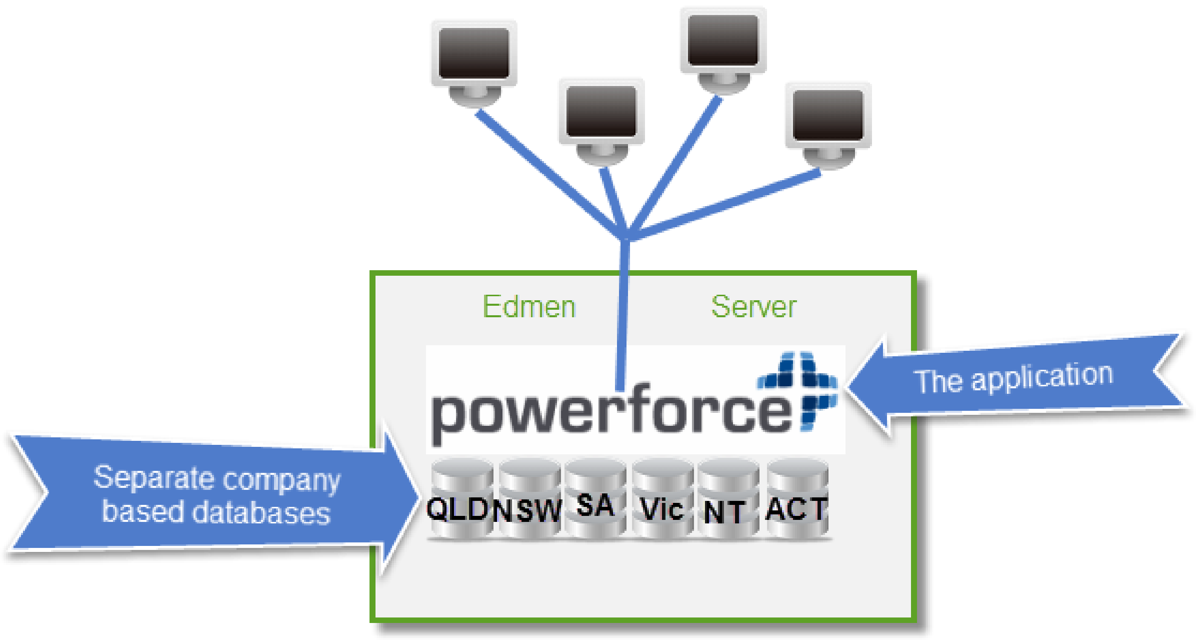
POWERFORCE STRUCTURE OVERVIEW

POWERFORCE AS A DESKTOP APPLICATION

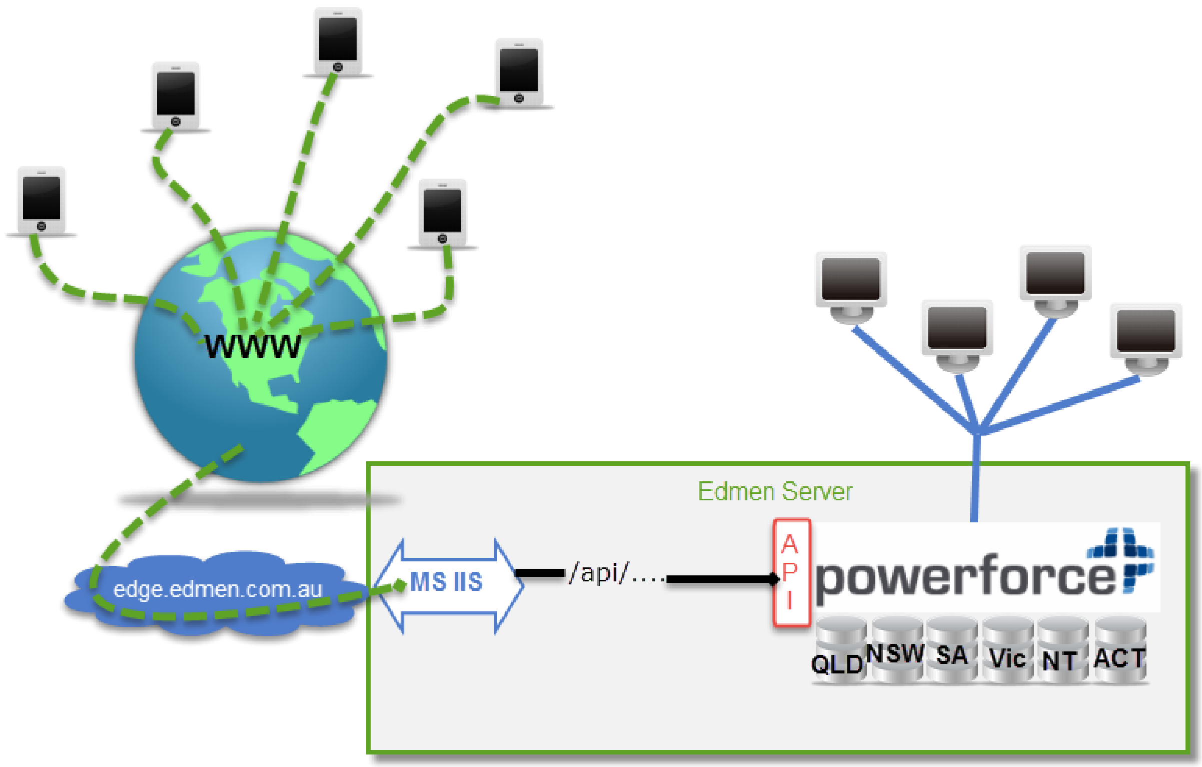
As a desktop application the conceptual structure is fairly straight forward. The application and the database/s live on the server and users generally connect via the internal network.



ENTER THE WORLD WIDE WEB AND MOBILE APPLICATIONS

As we move to the web and mobile applications, the way we interact with Powerforce changes. Application Programming Interfaces (APIs) are built to enable remote devices to interact with Powerforce in a controlled, consistent, standardised manner. A web server such as Microsoft's IIS is used to help manage and direct incoming requests to the appropriate location.

The APIs use the information coming in from the web to determine which database to tell Powerforce to use, whether it wants to retrieve or update information and which specific information that may be. The APIs are able to determine the correct database, based on the web user sending the request. Which information, is determined by the actual request (ie the part of the address that comes after "api/"). What to do with it is determined by the HTTP Verb that is sent with the request. A "GET" tells the APIs that the user wants to get some information. A "POST" tells the APIs the user wants to update some information. The APIs then either retrieve or update the relevant data and then format an appropriate response to return to the calling device/service.

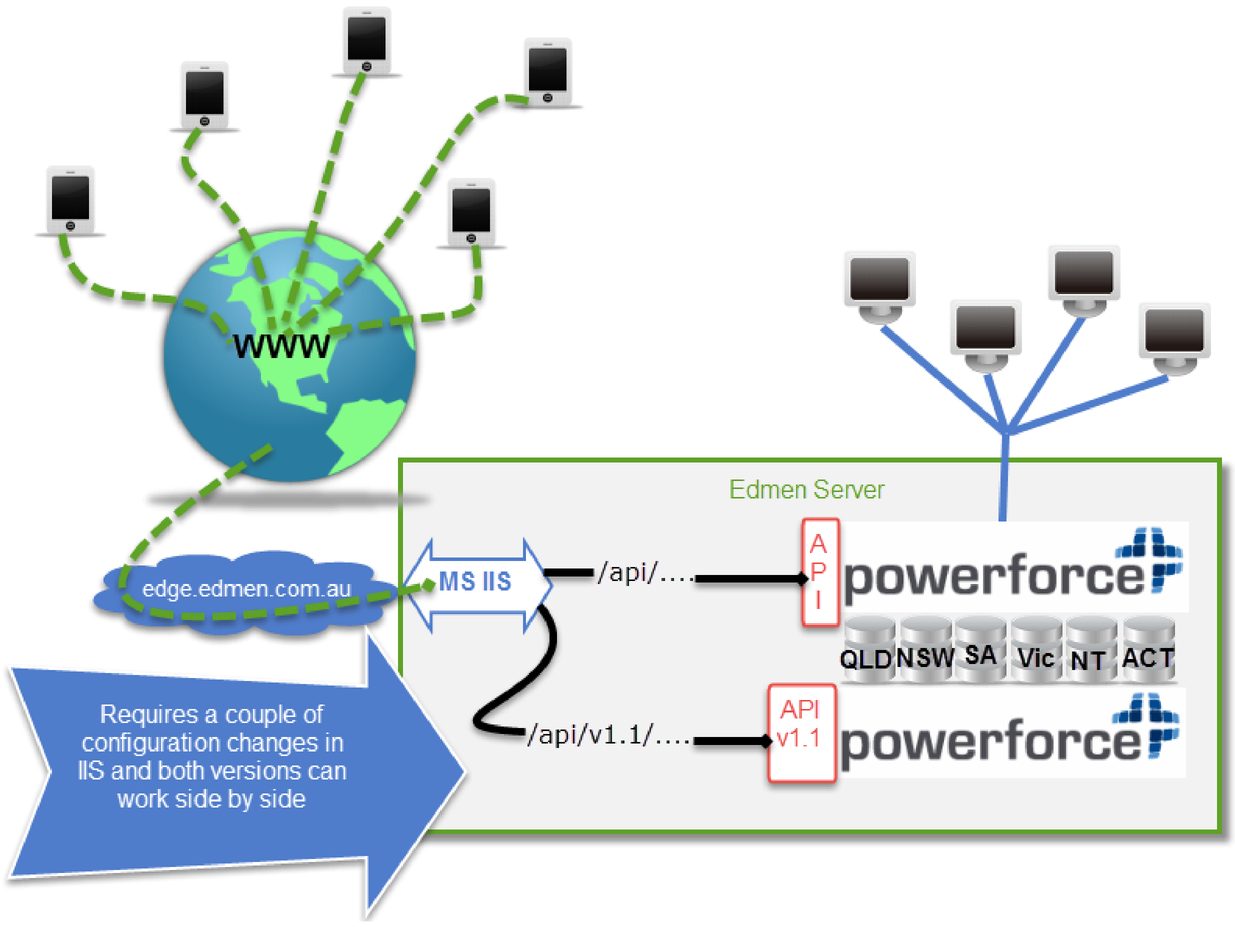


FUTURE EXPANSION. NEW FEATURES AND FUNCTIONALITY

Moving forward there will undoubtedly be enhancements in the form of new features and functionalities and improved ways of performing existing tasks. This will result in the addition of new API's and potentially some changes to the way existing APIs are structured.

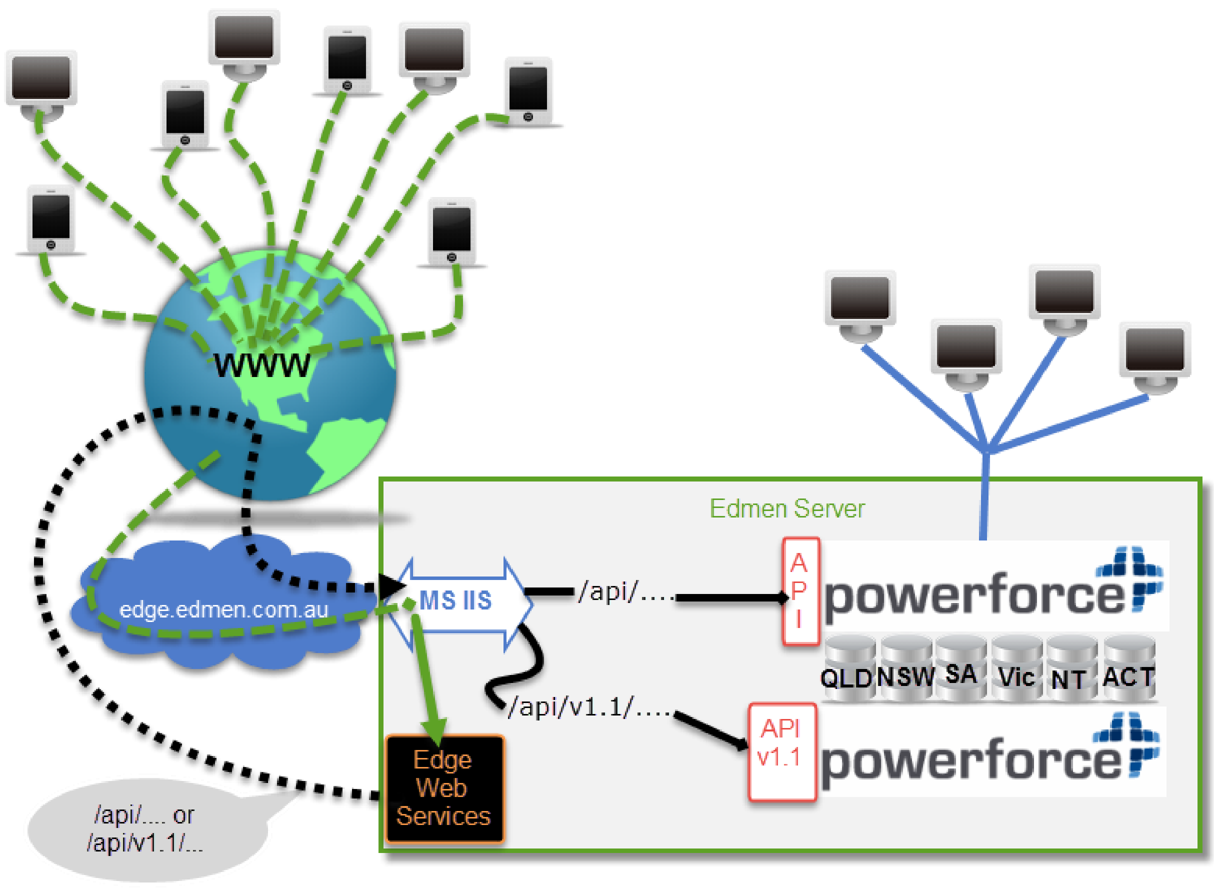
In order to roll these out with reduced or even no downtime and without breaking any existing mobile applications we will be able to place an additional copy of Powerforce in a different directory on the same server. The new copy will have the new APIs and can run concurrently with the existing system.

The differentiation between the two will be the API call from the remote device. For Example: the first release of the APIs are called simply by using something like *edge.edmen.com.au/api/employees*. When a new release (let's call it version 1.1) is exposed, the new call is similar but with the addition of the version number. Eg *edge.edmen.com.au/api/v1.1/ employees.* For this to happen all that is needed is a couple of minor configuration changes in the web server in order to redirect each call to the appropriate place. This way different devices can be on different versions and both continue to work until such time as a business decision is made to end support for the original version.



EDGE IMPLEMENTATION

As part of the complete Edge implementation there are a few more moving parts involved which fall outside of the intent of this document. Some of those are provided by CompuEase and therefore I cannot speak authoritatively about them. The diagram below does however provide an overview of how they fit into the overall picture.



The web services will make the calls to the Powerforce API and in return will receive and interpret the response, then send it onto the relevant device. It will therefore be the responsibility of the web service to form the correct call ie to choose *edge.edmen.com.au/api/* or *edge.edmen.com.au/api/v1.1* etc.

WHAT IF THE "…." BIT OF "API/…." CHANGES?

Firstly, an explanation. The "…." bit that has been ignored in this document so far, is the part of the call/address that dictates specifically what to do. For the following examples we will assume that these are all "GET" requests which means we only want to see or retrieve some information.

api/employees/<employeeid> - the "*/employees*" part indicates that we want some information about an employee. The "*/<employeeid>"* tells us which employee we want.

Let's go further. Now we want to specify exactly the sort of information we're looking for.

api/employees/<employeeid>/profile - the "*/profile*" part on the end indicates that we want to know the profile for this employee which currently consists of their skills and their likes and dislikes.

How does the web services know which call to make?

1. Documentation. Each call will be documented to list what they are and what they're for.
2. HATEOAS. This is a principle that ensures that the most current link information is always available to third parties. In short, it means that with each response from a call, there is a list of other relevant calls that can be made, complete with the current link/address. Based on the previous examples, the call to *api/employees/<employeeid>* will return the basic information about the employee but also some other information that tells them they can now call *api/employees/<employeeid>/profile* to retrieve the skills etc.

So what if "/profile" changes?

There may come a time where the "/profile" is no longer appropriate for the information required. For example it may be a business decision to separate skills from likes & dislikes. This may result in two new calls like this "*.../<employeeid>/skills"* and "*.../<employeeid>/likes".*

If the remote devices continue to call the *"/profile"* address, they will no longer work.

Documentation can help with this except for two concerns. One, despite best intentions, documentation is always delayed primarily because it can't be started until the work is finished. Secondly, it relies on third party developers continuously monitoring documentation for changes just in case.

This is where HATEOAS comes to the fore. The links within responses are always current because they are updated as the functionality is built. Each link has a corresponding label (or in the Powerforce case, an "object") which indicates what the link is for. As long as the labels or objects don't change, then it is easier to keep up to date.

So for this example a request for "*api/employees/<employeeid>"* would return the name, address etc and a number of links, one of which would say:

"profile":

    "href":"edge.edmen.com.au/api/employees/<employeeid>/profile"

"href" is a term that indicates this is a link.

After the changes the "profile" object would remain but with the new link and an additional object would appear.

"profile":

    "href":"edge.edmen.com.au/api/employees/<employeeid>/likes"

"skills":

    "href":"edge.edmen.com.au/api/employees/<employeeid>/skills"

In this scenario, there could be a couple of outcomes depending on the third party implementation.

If the web services had been developed to always call "*/<employeeid>/profile*" either based on the original documentation or based on what was originally seen in the HATEOAS links, then the application would just stop working or at least would receive a 'not valid' error response.

If on the other hand, the web services were developed to always look for the "profile" object to determine the next step then the services would at least still retrieve the employee likes and dislikes without any changes having to be made. There would still need to be an enhancement of some sort to retrieve the skills as this is completely new.

IN SUMMARY

The Powerforce APIs are being built with future growth in mind. They are intended to be as robust, flexible and scalable as possible. New features or modifications to existing features will inevitably require changes to third party services that interact with the APIs.

What changes might break or hinder mobile or web applications?

There are three key areas outlined in this document that will require modifications to external applications such as the Edge web services. The extent of these modifications will be dependent upon the original implementation of the services.

1. **New features**. This is a given. New features and functionality will naturally require enhancements to both sides of the equation.
2. **A collection of changes and new features necessitating a new version.** This is where the generic component of the HTTP call will change. Eg. From edge.edmen.com.au/api/... to edge.edmen.com.au/api/v1.1/....
3. **Changes to existing features.** This is where an existing feature/function is modified for either business or efficiency reasons resulting in a change in the specific component of the HTTP call. Eg. From api/employees/<empid>/profile to api/employees/<empid>/likes.

The success of this project both now and into the future relies on ongoing collaboration between all parties and won't be without hiccups. They should however be minimised with effective detailed consultation, planning and design.