**Getafix – Actionable and Repeatable Knowledge Base System**

Why Getafix

Being part of vRealize suite of products, the developers are involved in day-to-day support calls related to the products that they work for.

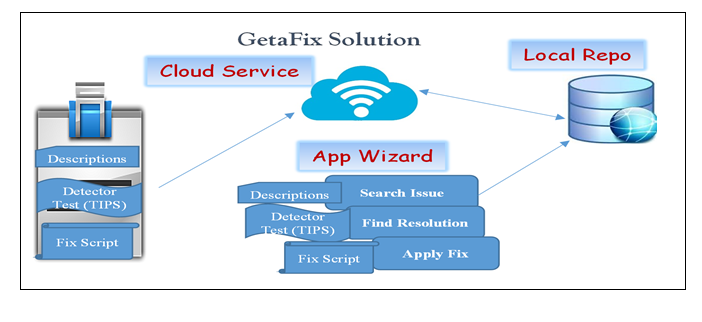
1. Roughly 70-80% of the customer cases were due to issues in environment/deployment and not code. i.e., the issues were resolved by adjusting the environment to match the assumptions in code, than vice versa.
2. Environment issues were fixed by having a call between the customer, the support engineer and authors, where the environment would be adjusted to meet the assumptions of the product.
3. Of the code issues, at least about 50% of them had a work-around that would make the product work. Typically, such workaround would be applied on a WebEx involving the customer, support engineer and sometimes developer.
4. Code issues that did not have a workaround and were very urgent would be fixed by providing hot patches in the form of jar files or war files2. Instructions would be provided on deployment of these hot fixes which would be followed by support, and in some cases developers would be involved.
5. Of all kinds of issues, roughly 3 out of 4 customer cases that developers get involved in, would demonstrate one or a combination of issues that were already diagnosed and fixed.

The repeating cycle of support experience with above observations gave rise to the thought that there must be a part of this cycle that can be easily automated and operationalized.

What is Getafix?

Figure 1 Getafix Overview

On the developer site, whenever a developer resolves a customer issue, he contributes to the Getafix issue database server (Cloud Service) by adding the test case to diagnose, and the resolution script that fixes the issue, along with descriptions and symptoms. This way, Getafix server will have a growing list of ‘detected issues’, ‘test cases’ to diagnose them, and corresponding ‘fix scripts’ to fix the issue.

On the customer site, these ‘issue updates’ will be downloaded and kept updated. Thus, when a customer observes a problem, he can run a ‘Diagnostic scan’ similar to that of windows but using an ever updated issue list, on a specific area that he is facing a problem in. Since the issue database is populated with issues from other customers, there is a higher likelihood that a customer will find a resolution to his problems.

The developer end of the solution. Once a customer issue is diagnosed and fixed, developer would like to capture a couple of artifacts describing the case. A developer can also choose ‘test cases’ and ‘fix scripts’ from existing files in the system. To provide a fix to the given problem, developer can write simple shell script to fix the issue. Script or even a Program that can correct the issues.

The solution at customer end comprise of a local database which is in sync with the central server and gets updated with new issues and resolutions when any developer uploads it. So, whenever a new issue is updated in the central server, the database of all other customers are updated with it.

It is powered by Keyword based searching mechanism to get a better match of problem from the database when the customer enters the problem he is facing.

Customer can select one or more symptoms from the list and enters the target environment details to run the diagnosis. It then triggers and executes the test case associated with those symptoms. If any of the test case fails, it will indicate the root cause of the problem. As every test case is also associated with the resolution description and resolution script, the customer is then alerted with these resolutions which also gives an idea of its impact on the environment if the fix is executed. Customer can decide at this point to run the script to fix the issue. It is possible at this point that the customer may choose to file a support request to confirm this diagnosis and resolution. A customer can also choose to just apply the fix if he feels the fix is not very risky. He provides the inputs required by the fix script and Getafix runs it for him on the target environment to fix the issue.

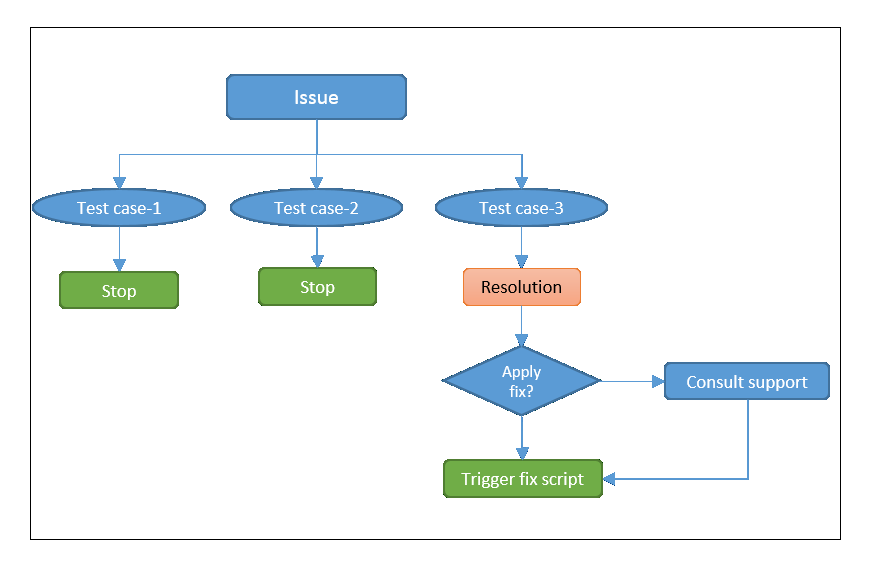


Figure 2 Flow of Getafix

Conclusion

Getafix is a platform where the customer searches his problem using his own words and got to a solution. This shows how the customer could perform a self-service support without having to rely on documentation for troubleshooting. Further, the customer did not have to align his terminology with that of products before looking for a known problem. A customer can apply the fix with the help of the onsite tool without having to contact support. This shows that precious time of customer was saved by not having him to go through the support cycle. It also shows that the developer’s and support engineer’s time were saved since they did not have to deal with a repeating issue again.

This way, Getafix will solve all the problems that were identified and stated in a support scenario.

Further Improvements

1. We will use a WorldNet based dictionary for search purpose. In addition, we would embeds business domain specific equivalents of the terms to help this search.
2. We also plan to introduce ‘Confidence Badge’ for the resolutions suggested by Getafix which is based on the statistical analysis of success rate of a particular resolution. This will help customers to gain confidence while applying the fix suggested by Getafix