



BUGSPLAT™

Turn Key Crash Reporting

Return on Investment Analysis Introduction

Software crashes are common - and frustrating. Regardless of the application, operating system, and your development team's best efforts, from time to time your users will experience a product crash. These issues can be due to bugs in the code, unexpected side effects caused by other applications or drivers, or discrepancies between the end user environment and internal testing environments.

Software users understand that software crashes, but they are usually too busy to report the problem. Often, they will:

- Ignore the crash, restart the application and continue working.
- Wonder whether your software's benefits outweigh the problems.
- Lose confidence in your product and question the reliability of your software.
- Consider investigating other products to replace yours.

And software crashes are expensive and time-consuming problems for your development team. Not only is it costly to identify and fix the problem, but if your engineers and developers are devoting time to these issues, they'll have less time to devote to developing new features or products.

Products that have a reputation for being "buggy" and unreliable don't last long in today's competitive marketplace. Your customers have many software options and they expect cost-effective products that work effectively. And if they do encounter a problem, they expect a quick, easy fix. They're often too busy to spend time with technical support resolving issues related to bugs.

In our connected world of online forums and blogs, customers share their opinions openly, and reports of unreliability will decrease your product's credibility and cause market share decline. However, the reverse is also true; a reputation of quick response and extraordinary customer service will be instrumental in increasing your customer base.

Crash Reporting Alternatives

When your customer experiences a crash using your product, there are a number of approaches you could take:

- Traditional support - your customer contacts customer support by email or phone when their system crashes. Historically, this has been the standard level of service for desktop applications.
- Create a crash report automatically, and launch the customer's email with the report as an attachment.
- Subscribe to Microsoft's error reporting service, WinQual.
- Use Google Breakpad.

- Create your own crash reporting solution.
- Integrate BugSplat into your application. BugSplat offers significant benefits over these other options.

Let's examine each of these alternatives in turn.

Traditional Support

This approach is the default solution provided by most companies. To do it well, you must have a strong technical support presence, including email and telephone, to resolve and track issues. There are many drawbacks to this approach:

- Analyzing a crash by phone is an expensive and sporadic proposition.
- Sometimes you will be able to reproduce your customer's crash by loading their data files, but often it takes many hours to diagnose the issue. The problems are often related to a series of specific steps that can be difficult to replicate.
- Not all defects are reported, as many users will find it too frustrating or time-consuming to initiate a technical support request.
- The crash report data you receive will likely be fragmented and unreliable. You may not know about all of the product crashes, so it will be difficult to trend the data and determine whether your product is becoming more reliable.
- With this system, you can't easily notify customers who are experiencing problems to let them know it has been resolved in a new version or service pack.
- Telephone support is typically a time-consuming proposition, and if your product crashes often enough, your customers will migrate to a more reliable competitive product rather than spend additional time speaking with technical support to resolve on-going problems with yours.
- An automated crash reporting system won't replace your technical support team, but it can lower the costs of providing technical support and significantly improve your customers' experience when a failure occurs.

Automated Crash Reporting Through Email

This approach involves capturing a crash report (for example, on a Windows system, you might create a minidump file) and attaching this report to a template email using your customer's email client. You're able to obtain quantitative data about the reliability of your software without investing in any third-party service or backend infrastructure. And your customer is more likely to report errors knowing that this information will go directly to the company creating their software package. However, there are significant disadvantages:

- You're using the customer's email client, which might cause your user some concern.
- Email is filtered and restricted by a wide range of security software.
- An email-based solution makes automated crash processing of reports very difficult.

- Your team needs to manually ‘debug’ every crash that is reported. Until this is done, there’s no way of knowing if the defect causing the crash has been investigated. In other words, you may waste time debugging something that is already fixed.
- You won’t get any automatic trending data. A small percentage of bugs cause a significant amount of crashes, so you may be fixing a crash that only impacts a single customer.
- If you decide to track the results, you’ll need to add the overhead of manually entering the data into your database and defect tracking software.

Microsoft’s WinQual Service

Another option to consider is Microsoft’s error reporting service. The only real advantage of using the Microsoft solution is that they will bucket your crash reports based on the module crashing, and the offset within that module, providing you with some level of trend data. But this approach has significant disadvantages:

- This option only works for native Windows applications.
- While “free,” as in you don’t need to pay Microsoft, there are still a number of real and assumed costs.
- Your end customer sees the generic Microsoft error reporting dialog - they have no way of knowing that your company will see the report.
- End users have little incentive to report the error. They naturally wonder why Microsoft would fix a problem in some other company’s application.
- A custom-built crash reporting dialog with your company’s identity is a powerful message to your customer that you stand behind your software. Users are far more likely to send crash report data with a custom-branded crash report dialog.
- Because of Microsoft’s extraordinarily restrictive privacy policy, it’s impossible to determine the identity of the customer reporting the crash, so you have no opportunity to follow up with proactive customer support.
- You need to create a Verisign certificate to use the service (around \$400 per year).
- You still need to manually debug each minidump against a local Symbol Server to get the full set of information from the minidump file. You’ll incur costs for both the Symbol Server setup and also 30-60 minutes of staff time per minidump to debug.
- Microsoft only retains a statistical sampling of minidump files received, whereas a custom or BugSplat solution will retain them all.
- You have no opportunity to gather your customer’s contact information or determine what they were doing prior to the crash.

Google's BreakPad Service

Google provides an open source error reporting solution called Breakpad. Detailed information can be found at the website <http://code.google.com/p/google-breakpad>. Breakpad applications can create minidump files that are transferred by a user's email back to the software publisher. This approach has the following limitations:

- Customers must elect to send the crash report with their own email system, rather than simply selecting "OK" on a crash report dialog box.
- There is no automated server to collect the crash reports. Instead they are typically sent to an email account at your site.
- As with WinQual, there is no automated processing of crash reports that your company receives. A developer must setup symbol servers and run a debugger to decode.
- No historic crash reporting features that show stability trends over time.
- Breakpad requires significant software development resources to setup and maintain.

Create Your Own Solution

Creating your own crash reporting system is tempting, as it may appear to provide cost savings. However, this option means you'll:

- Incur high costs, including investments in development, infrastructure, hardware and maintenance that could add up to more than \$1 million.
- Need to design and build the system.
- Need to write a front end for the database that collects the error report data, which takes significant time and money.

The BugSplat Solution

BugSplat has developed a sophisticated and highly scalable infrastructure for handling crash reports from many different platforms. We believe there are significant advantages to using our approach, including:

- Many thousands of developer hours have gone into developing the BugSplat solution. It's continually enhanced based upon customer requests.
- The development cost is split across many customers, so your company will receive a huge amount of functionality at low cost.
- The BugSplat solution is ready now. No significant lead time is required to get this solution into your customer's hands.
- We reduce the debugging time from 30-60 minutes (the length of time it would take a developer to debug) to a mere 30-60 seconds, saving your company significant time and money.

- Since BugSplat crash reports are automatically processed, they are available to your entire development team. Software developers, qa staff, tech support staff, and engineering managers all routinely inspect BugSplat databases to gain insight into their applications stability and support issues.
- BugSplat works with an impressive - and always expanding - list of platforms, which are outlined at <http://www.bugsplatsoftware.com>.
- We're able to collect and report your customer's crash data in various ways. For example, you can get a crash summary report that groups the same crashes together so you can quickly determine which problems are impacting your users the most.
- Each crash is independently and automatically 'debugged', and a call stack is generated along with function names, line numbers, modules loaded and value.
- We write our own debugging service to automatically calculate call stack and other information from a minidump file using Symbol Servers. This is performed by a sophisticated farm of machines, keeping the load off the main server and ensuring quick response time.
- BugSplat continues to scale our backend, as necessary, to ensure a high uptime for our customers and their customers.
- BugSplat implementations require a minimal amount of developer time to integrate. Integration can be as simple as adding a line of code and linking with the BugSplat library.
- BugSplat provides crash-specific technical support responses to any reported crash. The next time a customer experiences the same crash, they can immediately be presented with a webpage that includes your corporate identity and your crash-specific response. Instructions for upgrading your product or working around the problem are delivered just when a user needs them.
- We charge a modest annual fee for using the service. Charges are based on the number of defects that are reported to our site. And as your application gets more stable, your costs will decrease. If after a new release you wish to lower your costs further, you can easily mark earlier versions as retired so you no longer receive (or pay for) those reports.

Estimated Cost Summary

For this analysis, we've assumed that you have a fairly large user base that would report about 5,000 software crashes a month across your customer base. This is not uncommon for a product with 20,000 - 30,000 users.

We have also assumed that the top four lines of code that cause a crash are responsible for more than 25% of the defects, and the top twelve lines of code that cause a crash are responsible for 50% of the defects.

These assumptions are based on real world data from a broad spectrum of existing BugSplat customers. The values below will vary based on a number of factors such as customer loyalty, availability of customer support and the experience of your software development team.

	Approach 1: Traditional	Approach 2: Email	Approach 3: WinQual	Approach 4: Breakpad	Approach 5: Roll your own	Approach 6: BugSplat
Total # user crashes	5000/mo	5000/mo	5000/mo	5000/mo	5000/mo	5000/mo
Total # user crashes reported per month	~50	~2500	~1000 ¹	~2500	~5000	~5000
Reports processed to achieve significant reliability gain and trending data.	50	100	25	100	25	25
Average manpower spent per report	CS ² : 120 min SWD ³ : 45 min	SWD: 45 min	SWD: 45 min	SWD: 45 min	SWD: 1 min	SWD: 1 min
Average manpower cost per report	\$70	\$30	\$30	\$30	<\$1	<\$1
Total report processing cost per month	\$2,500	\$3,000	\$750	\$3,000	\$40	\$40
Crashes prevented for each defect fix	~5%	~10%	25-50%	~10%	25-50%	25-50%
Reliability increases per major release	Minimal	Slight	Better	Slight	Excellent	Excellent
Initial startup cost	\$0	\$5000 ⁴	\$250 ⁵	\$2500 ⁶	\$100,000 ⁷	\$500 ⁸
Regular monthly cost of analyzing specific crashes	\$2,500	\$3,000	\$750	\$3,000	\$40	\$40
Monthly maintenance cost	\$0	\$500	\$500 ⁹	\$500	\$1000 ¹⁰	\$500
Total Annual Cost	\$30,000	\$42,000	\$15,000	42,000	\$12,480	\$6,480

1 Customers will not send defect reports to Microsoft as often as they will send them directly to your company.

2 Customer Service, assuming \$20 per hour.

3 Software Development, assuming \$40 per hour.

4 Assumes development time required to implement email system, and setup appropriate aliases.

5 Assumes development time configuring Verisign certificate.

6 Assumes development time required to implement email system, and setup appropriate aliases.

7 Assumes initial hardware costs, as well as total cost of development of the custom solution.

8 Assumes development time integrating BugSplat library and modifying resource library.

9 Includes symbol server support for all supported operating systems and all application versions.

10 Includes allowance for monthly storage, bandwidth and backup costs.