Creating the Knowledge about IT Events

Gilad Barash, Ira Cohen, Eli Mordechai, Carl Staelin, Rafael Dakar
HP-Labs Israel

firstname.lastname@hp.com

Abstract

We describe a system which automatically associates documents containing knowledge about problems and their resolutions to IT events. IT professionals use these when troubleshooting problems in the IT environment. The system associates documents from several independent sources, both internal and external to the organization, using machine learning techniques to associate the most relevant and pertinent information.

Problem statement

In today's complex IT environment, millions of events are generated daily by hardware, software, middleware, applications, network, and storage system sources. These events describe, usually in text form, normal behavior as well as abnormal behavior (errors and outages) and are analyzed to alert and to help troubleshoot problems that may affect availability and performance of the business services using the IT infrastructure. IT support engineers are expected to interpret and act upon these quickly. For example, in one of the large financial firms we interviewed, Level 1 IT Support Engineers are required to react to an incoming alert event within two minutes.

Commercial software tools such as HP's OMi or IBM's NetCool, collect, filter and correlate events from the different components of the IT environment. IT organizations leverage these tools to consolidate their operations to a single group of support engineers that monitors events from the entire IT environment. To be able to react quickly to events, the engineers require a breadth of knowledge about the meaning of the events and the remediation procedures required to address them. Companies such as Splunk and LogLogic, collect and index raw event logs, making it easier to search through logs to identify events related to various types of problems, such as performance, security and failures. Significant research efforts have been dedicated in recent years to the analysis of logs, making them accessible for administrators and operators for understanding system behavior (e.g.,[1] [3]).

Over time, knowledge related to events is accumulated, but tends to exist in various decentralized document corpora: technology forums, wikis, product documents and incidents, to name a few. To leverage the existing body of knowledge, support engineers and administrators must *manually* search for the relevant documents, *manually* rank them based on relevancy and quality of information and utilize them to resolve the problem at hand, as search engines produce `noisy` results. There are various reasons for these noisy results: search engines do not know which document corpora (or domains) are relevant, or more importantly, irrelevant to the type of IT events an operator is handling, and the search result rankings do not account for the quality of the information. For example, searching documents containing the text of IT events stemming from an Oracle DB results in many pages that may talk about MySQL, and even pages returned from an Oracle forum, may contain unanswered questions of users, making those pages of low quality to the administrator in need of a solution.

In the following, we describe our system, which automatically collects, ranks, and associates knowledge documents describing IT events, the problems they represent and remediation procedures from various document sources – both intranet and WWW. This knowledge helps the support engineers resolve problems faster, and serves as the stepping stone for creating automated resolution plans for a portion of the problems detected through these events. In creating the system, we developed several novel components for accurately associating documents to events. Specifically, our system ranks documents based on the relevancy their source (e.g, the domain of a webpage), the quality of information, as relevant to solving problems (e.g., is a question in a forum thread answered or not), and the relevancy to the events seen in the system being debugged. In this paper we describe the system, and the algorithms used for each of the ranking mechanism. Our main contributions are as follows:

- Proposed and built a system which automatically associates knowledge articles to IT events, with the stated purpose of aiding administrators understand remediation actions when they appear.
- Developed a novel source ranking algorithm for determining which knowledge sources are most relevant to events coming from a specific system
- Developed a system for determining the quality of information in technology forums. Our system extracts generic attributes related to quality (such as rankings of users and number of posts), and infers, using classifiers, other quality measures, such as whether the question in forum thread was

answered or not.

- Developed a novel algorithm for learning classifiers in the presence of label noise. This was required since training data necessary to infer quality of information measures was noisy in terms of the class label given to the training examples.

In the following sections we'll describe the overall system and its components, provide details of the technology with results on data collected from various IT domains.

Solution Outline

Our solution consists of an event-driven system which automatically creates a knowledge base fed from various disparate content sources and organizing the content in a way that enables IT engineers' quick and efficient access to the knowledge they need to learn about and resolve a problem quickly.

The architecture of the solution is shown in Figure 1, with a detailed description below. The input is a set of IT events. For each IT event, or groups of events known to be correlated, the system performs the composition, search and rank functions. We now describe these steps in more detail.

Composer: The composer takes as input the text of an event, or group of events, and creates a ranked set of search queries, starting from queries that are constrained to include all words in the events in sequence, reducing the constraints in subsequent queries. Removal of constraints includes allowing flexibility in word ordering, removal of words that contain numbers of special characters, all the way to removal of words from the queries.

Searcher: The searcher traverses the set of queries, starting from the most restrictive, passes it to the available search engines, and collects the search results from each query in their ranked order. The searcher stops when a sufficient amount of results are returned (e.g., 20 pages). The searcher searches both the intranet (internal knowledge articles, sharepoints, wikis, online product documentation) and the web (technical websites, user forums, etc.) to find data that pertains to the events.

Ranker: The ranker goes over the set of search results, computes rank scores based on various criteria and combines the different rank scores using weighting of importance. The criteria are as follows:

1) **Quality of Information (QoI)**: Ranking pages based on the quality of information is a critical and innovative aspect of the system. We extract attributes from the retrieved documents and employ machine learning techniques for classifying and ranking the documents based on their quality. In documents such as technology forums (or incidents), where questions are asked by users, and answered by a set of other users, we extract quality attributes such as the expertise of the users answering/asking the questions, number of replies to a question, thread duration, and linguistic attributes. These attributes are also used as input features to classifiers that decide whether the question was actually answered, and thus should be ranked higher as a useful document.

2) **Content source ranking**: When searching the internet and intranet, there are many sources of document corpora (e.g., the Oracle technology forum,

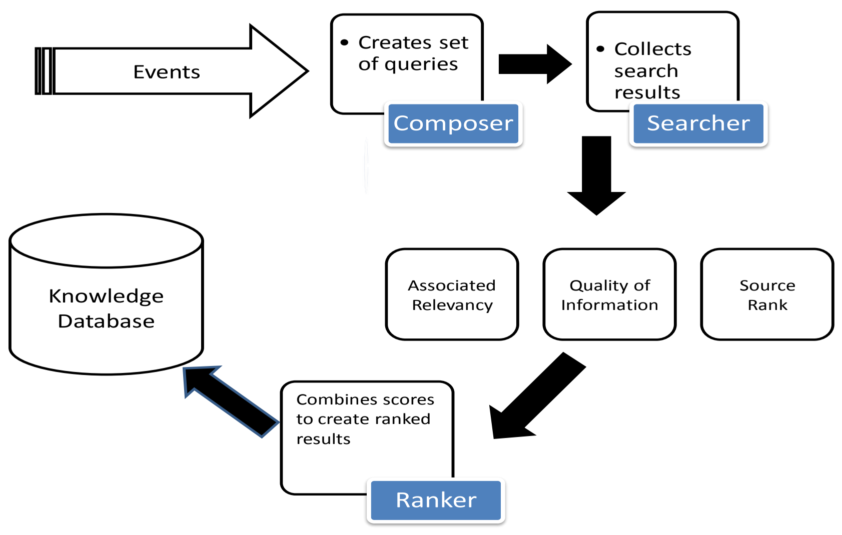


Figure 1: *Architecture of solution*

forums.oracle.com) that may be more or less relevant to the types of events seen in a system, e.g., an Apache forum site is less relevant compared the Oracle technology forum if the events come from a system running an Oracle DB. We developed a method for automatically ranking sources of documents based on the type of events being queried.

3) **Content relevancy ranking**: We compute the relevancy of each document to the IT events that were part of the query and problem description (if it exists). The relevancy computation includes the edit distance between the IT event text and the document (seeing how much of the event text actually appeared in the document), where in the document it appears and by whom (if it is a forum or incident, was it written by the user with the problem?). Additional attributes making up relevance are determined by the creation date of the document, determining whether it is still relevant to the version of the system/software which created the event (e.g., events from an Oracle 8 may be less relevant to an event generated by a system with Oracle 10).

   In the next sections we expand on each of the ranking technologies listed above.

Content Source Ranking

When our system queries knowledge corpora in regards to events, additional information can be leveraged to improve the search results. We sometimes know what the source of the events is (e.g., Apache, Oracle DB, j2ee based system, etc), and more often can identify that a group of events came from the same type of system, even without knowing its name, simply because the events came from the same log file. We leverage this information, coupled with the fact that we have many types of events on which we query each such source to rank different sources of documents, such as forum domains, internal knowledge databases, and any general internet domain. In a nutshell, we perform many search queries based on a large set of events from the same subsystem source (e.g., apache). We then collect all of the results, strip them to the origin domain (in the case of internet search), and count appearances and rank of each domain, to produce a ranked list of domains. The method works is described in Figure 2.

To demonstrate the results of the source ranking, we ran tests on two types of logs. The first was a windows logs containing mainly multiple printer events (HP printers). The second is a combination of log events from multiple components of HP Software's Business Availability Center software (in-house events, java events, jboss, apache, database events, etc). The top eight results, with their rank are shown in Table 1. As can be seen, hp.com ranks at the top in both cases, with enterprise related sites ranked high in the case of BAC's logs, and consumer related technology help sites in the case of the printer logs. The score of the ranking is shown, where we see that there is a quick drop in rank between the first rank and the 8'th rank.

Quality of information based ranking

Quality of Information is a concept that measures how fit certain data is for a use. In other words, it quantifies whether the correct information is being used to make a decision or to take a certain action. Defining, quantizing and standardizing QOI metrics is an effort that requires much thought and domain knowledge. There tends to be roughly two kinds of QOI metrics: ones that are innate to the data source and are context-free (such as date of creation, completeness), and those that are content-specific, which must be extracted from the unstructured text data within the data source. It is the latter type of metrics that pose the biggest research challenge due to their elusive and unstructured nature.

```
Algorithm 1: Content Source Rank

Input: Set of events E from the same subsystem source (e.g., Oracle DB events)
Initialize DomainHash  as Hashtable of domain names
For each event in E do
        Run the composer to create a set of queries
        Run the searcher on each query
        Collect the top K search results for each query with its search result rank i
        For each search result in their ranked order  (i=1,…,K)
                CurrentDomain ← Domain name from the full URL (e.g., hp.com)
                If the CurrentDomain did not appear in a higher rank
                \\ Add the inverse rank of the domain in the current search to previous counts
                        DomainHash(CurrentDomain) += 1/i;
Extract RankedList with weight of domains from DomainHash
```

Figure 2: Source Ranking Algorithm

Table 1 Source ranking results

Printer log events source ranking	BAC log events source ranking
1 hp.com 119.157375	1 hp.com 59.369148
2 ibm.com 65.521497	2 microsoft.com 51.931098
3 microsoft.com 43.811642	3 eggheadcafe.com 36.795678
4 oracle.com 42.549713	4 experts-exchange.com 33.695516
5 apache.org 36.819445	5 forums.techarena.in 25.293438
6 sun.com 28.664713	6 pcreview.co.uk 14.205668
7 scribd.com 25.912615	7 tech-archive.net 14.055154
8 jboss.org 25.084923	8 soft32.com 13.247566

In order to find the QOI of user forum threads, we extracted the metrics, both innate and content-specific. A partial list appears below:

- Whether the thread is marked as "Answered" or "Not Answered".
- The date the thread was last modified.
- Number of replies to the original poster (OP).
- Total number of words in the thread.
- Thread duration (in days).
- Highest rank of any poster in the thread.
- Is the last post written by the original poster?
- Does the last post include a derivation of the words "thank you"?
- Does the last post include a question mark?
- Is the last post written by the original poster and includes a derivation of the words "thank you"?
- Is the last post written by the original poster and a question mark?
- Is the last post written by the original poster and derivation of the words "thank you" and a question mark?
- Number of links

In order to extract these QOI metrics we implemented a utility that was capable of downloading forum threads and through mechanisms of screen scraping either found or calculated the above metrics for each forum thread. The utility supported scraping HP forums, as well as those of IBM and Oracle.

For the purpose of ranking user forum threads, one of the main QOI metrics is whether or not a thread is considered as having been answered, since these would be the results we would want to rank higher. Some forums contain this label, but many do not. We designed a system for inferring this measure from the other QoI measures.

In some forums, albeit not all, the notion of whether a thread was "answered" or "not answered" is marked by the original poster. However, even in these cases, we found that many threads can be inaccurately designated, leading to a problem of noisy labels. This is caused by the fact that marking a thread as "answered" is a manual operation that must be performed by the original user who posted the question, when and if his question is answered. All too often, users will find their question answered in the thread but neglect to check the box that designates the thread as "answered". Therefore, while threads that were designated as "answered" tended to always really have the answers in them, those that were left as "unanswered" many times did indeed include the answers to the original poster's question.

The research question then became two-fold:

a) Is it possible to learn whether a thread was answered or not based on the other features that we extracted?

b) Would the learning result from one forum domain (such as HP) be useful for forums in other domains (such as IBM or Oracle).

To test the ability to classify threads as "answered" or "not answered" we gathered user threads from two different public forums: Oracle (5500 threads) and IBM (1200 threads), and extracted 10 quality related attributes.

These sites were chosen because they provide a label to each thread on whether the question is answered or not. We trained decision tree classifiers (found to work the best) on the threads from the different sites. We tested the classifiers on the threads from the same site (using cross validation) and from the different site. The results are shown in Table 2. They show that while the accuracy is reduced when transferring a classifier from one forum to another (due to label noise), it remains fairly high, giving credence to generalization of the classifiers.

Table 2 Classification accuracy on IBM/Oracle forums

Train\Test	Oracle	IBM

To handle noisy labels, we use an ensemble-classifier training algorithm to help identify and remove misclassified labels [3], utilizing a concept of majority vote wherein different classifiers vote on whether or not the training sample is correctly labeled or not and the votes are tallied accordingly. This alleviates possibilities of over-fitting by one or more classifiers. We extended and improved the algorithm introduced in [3] by changing the step which discards training samples. We assumed that if a majority of classifiers misclassified the sample, there was a good chance that if we flipped its label it would then be correctly classified. Rather than discarding training samples that were voted on as having been misclassified, we flipped their labels and added them back into the new training set, thus boosting correct classification without throwing out any samples. The algorithm is described in Figure 3.

We tested out the revised algorithm on datasets from the UCI repository with a varying amount of artificially-places noisy labels in order to see how robustly it deals with them. For each dataset we randomly flipped between 10 – 40% of labels so that they became noisy prior to classifier training.

The classification accuracy results of running both methods on the Forest Cover dataset for each of five noise levels appear in Table 3 below.

Table 3 Accuracy results for methods handling label noise

Method \ %Noise	0%	10%	20%	30%	40%
Classification without filtering	0.7794	0.7527	0.7341	0.6877	0.6464
Ensemble Filter	0.7813	0.7681	0.7488	0.7237	0.6912
Ensemble Flip Filter	0.7805	0.7680	0.7499	0.7258	0.6940
Noiseless data	0.7794	0.7794	0.7794	0.7794	0.7794

Oracle	90%	85%
IBM	79%	97%

```
Algorithm 2: Ensemble classification method
• For every sample
    o For every classifier
        ▪ Classify sample with classifier
        ▪ If given label equals classified labels, return 1
        ▪ Else return -1
    o Aggregate results of all classifiers for each sample
    o If result > 0 (i.e. majority of classifiers classified sample correctly), add sample to new
      training data
    o If result < 0 (i.e. majority of classifiers misclassified sample)
        ▪ Randomly flip label and add to new training data if two random coin tosses are
          true: one based on weight of sample and the second based on the average
          predicted class distribution of opposite class
        ▪ Else discard sample
• Train all classifiers with new training data
```

Figure 3: Ensemble method for handling noisy labels

Based on the results we made the following observations:

- The ensemble method is robust to different noise percentages.
- Flipping the noisy labels seems to always be a good idea and improves the classification ability.
- Multiple flip iterations did not seem to improve the results.
- Less restrictive classifiers (Nearest Neighbor / Decision Trees) were more sensitive to label

noise.

- False negatives (left-over noisy samples) have more detrimental effect compared to false positives (non-noisy samples with low weights).

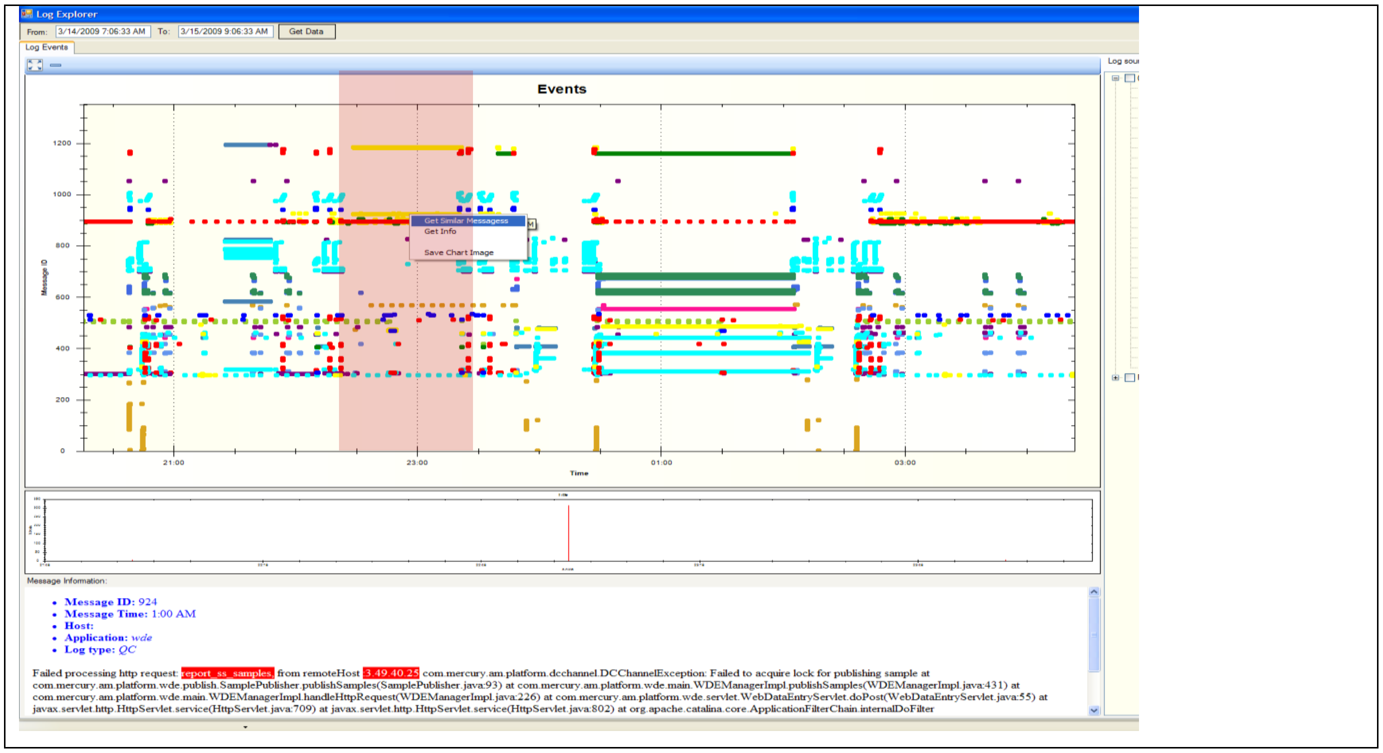
We also created a modified version of the ensemble learning algorithm with weighted sampled that iteratively reduced mislabeled samples' weights. This was shown to perform better with higher levels of noise in the data, but was less robust to low level of noise. Given that the level of noisy labels in the data is unknown, we opted to use the ensemble method with the flipping mechanism for our system.

Combined Ranking

We combine the results of the various ranking methods in three steps. First, the search engine returns a ranked list of results. In our initial implementation we used the google search API, and therefore the ranking is based on the google search engine. Second, for each search result we extract the domain, and average the google rank of the page with the source ranking position (weighted by the source rank weight). This produces a new ordering of the search results. Lastly, we change the order of the results based on the quality of information measures, when the results are forum threads – giving highest weight to whether there is an answer or not, and lower weights to the other quality measures. The ranked results, together with all of the separate ranking measures are stored in the DB. This gives the flexibility to re-rank the order based on inputs from the users on the quality of the search results, giving higher or lower weights to the various ranking measures.

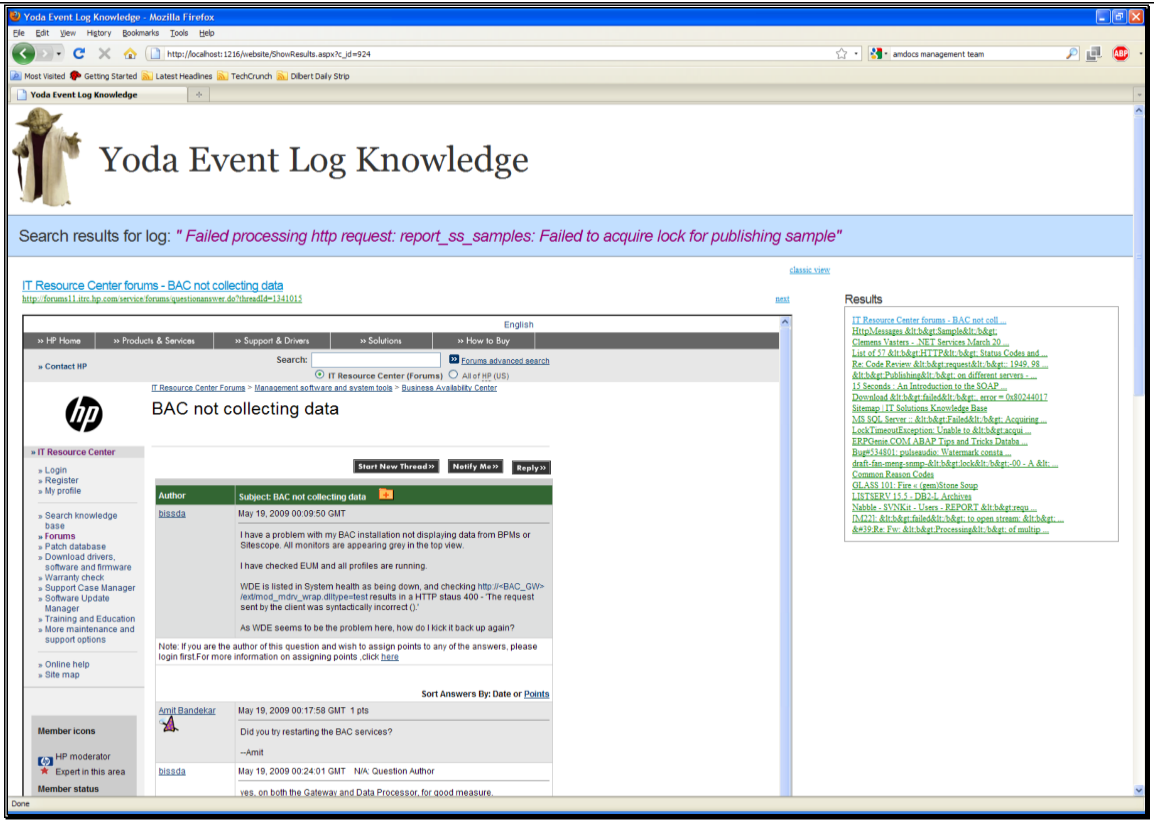
Demonstration of the system

A prototype of the proposed system was implemented and has been tested using events from several systems. In the first test, we collected events (200,000) from an HP BAC product installed at a customer site. During the collection period, BAC suffered a data collection problem, which was reflected in a subset of the events, discovered by our event correlation system. This period is shown as a shaded period in the top view of Figure 4. As can be seen in the bottom image of Figure 4, our system associated with these events threads from the HP technology forum which described a similar problem encountered by a different customer, and its workaround. The solution described in the forum allowed the engineers to quickly workaround the problem, until a code fix was produced.



- **Message ID:** 924
- **Message Time:** 1:00 AM
- **Host:**
- **Application:** *wde*
- **Log type:** *QC*

```
Failed processing http request: report_ss_samples, from remoteHost 3.49.40.25 com.mercury.am.platform.dcchannel.DCChannelException: Failed to acquire lock for publishing sample at
com.mercury.am.platform.wde.publish.SamplePublisher.publishSamples(SamplePublisher.java:93) at com.mercury.am.platform.wde.main.WDEManagerImpl.publishSamples(WDEManagerImpl.java:431) at
com.mercury.am.platform.wde.main.WDEManagerImpl.handleHttpRequest(WDEManagerImpl.java:226) at com.mercury.am.platform.wde.servlet.WebDataEntryServlet.doPost(WebDataEntryServlet.java:55) at
javax.servlet.http.HttpServlet.service(HttpServlet.java:709) at javax.servlet.http.HttpServlet.service(HttpServlet.java:802) at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter
```



http://localhost:1216/website/ShowResults.aspx?c_id=924

Yoda Event Log Knowledge

Search results for log: *" Failed processing http request: report_ss_samples: Failed to acquire lock for publishing sample"*

IT Resource Center forums - BAC not collecting data
http://forums11.itrc.hp.com/service/forums/questionanswer.do?threadId=1341015

IT Resource Center Forums > Management software and system tools > Business Availability Center

BAC not collecting data

Author	Subject: BAC not collecting data
bissda	May 19, 2009 00:09:50 GMT I have a problem with my BAC installation not displaying data from BPMs or Sitescope. All monitors are appearing grey in the top view. I have checked EUM and all profiles are running. WDE is listed in System health as being down, and checking http://<BAC_GW>/ext/mod_mdrv_wrap.dlltype=test results in a HTTP staus 400 - 'The request sent by the client was syntactically incorrect ().' As WDE seems to be the problem here, how do I kick it back up again?

Note: If you are the author of this question and wish to assign points to any of the answers, please login first.For more information on assigning points ,click here

Sort Answers By: Date or Points

Amit Bandekar	May 19, 2009 00:17:58 GMT 1 pts Did you try restarting the BAC services? --Amit
bissda	May 19, 2009 00:24:01 GMT N/A: Question Author yes, on both the Gateway and Data Processor, for good measure.

Results

IT Resource Center forums - BAC not coll ...
HttpMessages &lt;b&gt;Sample&lt;/b&gt;
Clemens Vasters - .NET Services March 20 ...
List of 57 &lt;b&gt;HTTP&lt;/b&gt; Status Codes and ...
Re: Code Review &lt;b&gt;request&lt;/b&gt;: 1949, 98 ...
&lt;b&gt;Publishing&lt;/b&gt; on different servers - ...
15 Seconds : An Introduction to the SOAP ...
Download &lt;b&gt;failed&lt;/b&gt;, error = 0x80244017
Sitemap | IT Solutions Knowledge Base
MS SQL Server :: &lt;b&gt;Failed&lt;/b&gt; Acquiring ...
LockTimeoutException: Unable to &lt;b&gt;acqui ...
ERPGenie.COM ABAP Tips and Tricks Databa ...
Bug#534801: pulseaudio: Watermark consta ...
draft-fan-meng-snmp-&lt;b&gt;lock&lt;/b&gt;-00 - A &lt; ...
Common Reason Codes
GLASS 101: Fire « (gem)Stone Soup
LISTSERV 15.5 - DB2-L Archives
Nabble - SVNKit - Users - REPORT &lt;b&gt;requ ...
[M22]: &lt;b&gt;failed&lt;/b&gt; to open stream: &lt;b&gt; ...
&#39;Re: Fw: &lt;b&gt;Processing&lt;/b&gt; of multip ...

Figure 4 Snapshot of system

Summary

We presented a system which links documents to IT events, to create a knowledge base linking the events to relevant problems which were previously documented. Our system leverages search engine technology, enhanced with rankings based on content source and quality of information relevancy, to produce a list of documents relevant to an event or group of events. We introduced the research problems related to these ranking problems, and suggested solutions, in form of machine learning algorithms.

References

[1] Aharon, M. Barash, G. Cohen, I., Mordechai, E. **One Graph Is Worth a Thousand Logs: Uncovering Hidden Structures in Massive System Event Logs**. European Conference on Machine Learning 2009.

[2] Brodley, C. Freidl, L. **Identifying Mislabeled Training Data,** Journal Of Articial Intelligence Research 11, 1999.

[3] Wei Xu, Ling Huang, Armando Fox, David Patterson, Michael Jordan. *Large-Scale System Problem Detection by Mining Console Logs.* Proc. SOSP 2009