



**A Guide to Diagnosing .NET Application Health  
Problems with Operations Manager 2007**

and the

**AVIcode .NET Management Pack**

## Abstract

The AVIcode .NET Management Pack extends Operations Manager 2007 by enabling a comprehensive view of application behavior and providing real-time diagnosis of code failures and performance bottlenecks. This paper describes how the AVIcode .NET Management Pack for Operations Manager 2007 can be used in conjunction with Microsoft System Center Operations Manager 2007 for complete:

- end-to-end performance monitoring of line of business (LOB) applications,
- service level availability (SLA) monitoring for connections to 3rd party web services,
- analysis of security, connectivity, coding and performance problems,
- pinpointing load balancing issues,
- understanding trends in historical data,
- and protecting sensitive information.

This document shall not be duplicated or used for any purposes other than that for which it is being provided. The information disclosed herein was originated by and is the property of AVIcode Inc. and except for rights expressly granted by written consent, such information shall not be disclosed or disseminated in whole or in part. AVIcode reserves all rights hereto. The word AVIcode and Intercept and the AVIcode and Intercept logos are service marks of AVIcode™ Inc. Throughout this document, other trademarked names may be used. Rather than put a trademark symbol in every occurrence of a trademarked name, we state that we are using the names only in an editorial fashion, and to the benefit of the trademark owner, with no intention of infringement.

©2007 AVIcode. All rights reserved. AVIcode and Intercept Studio are trademarked. All other trademarks are the property of their respective owners. 09/07

---

## Table of Contents

Overview .....	4
The Application .....	4
End-to-end Performance Monitoring for LOB Applications .....	7
SLA Monitoring for Connections to 3 <sup>rd</sup> Party Web Services .....	9
Security Problems: Detection and Root-cause Determination.....	10
Connectivity Problems: Detection and Root-cause Determination .....	11
Coding Problems: Detection and Root-cause Determination .....	11
Performance Problems: Detection and Root-cause Determination .....	12
Slow Database Issues .....	12
Slow Application Code Issues .....	13
Server Resource Utilization Issues .....	14
Monitored Server Key Metrics .....	14
Monitored Server Trends .....	15
Monitored Server Worker Processes .....	16
Application Statistics .....	17
Application Load Balancing and Performance Issues .....	18
Reporting.....	18
Data Protection.....	20
Sensitive Field Recognition .....	20
HTTPS .....	20
Application Groups .....	20
Conclusion .....	21
Other System Center 2007 Solutions from AVicode .....	22
AVicode BizTalk Server 2006 Management Pack.....	22
AVicode Office SharePoint Server 2007 Management Pack.....	22
AVicode Reporting Services Management Pack.....	22

## Overview

In order to meet the demands of a complex IT landscape that requires ever-increasing flexibility and agility, along with careful cost management, Microsoft developed System Center, a family of IT solutions that promotes proactive planning, deployment, management, and optimization of IT environments.

Microsoft System Center Operations Manager 2007 provides a standardized, comprehensive view of the health of your IT environment, including hardware, software and operating systems, no matter how complex or "venderogeneous" the environment may be. The end result is improvement to a company's bottom line and productivity.

The AVIcode .NET Management Pack extends System Center Operations Manager 2007 by enabling a comprehensive view of application behavior and real-time diagnosis of code failures and performance bottlenecks. It delivers a consolidated view of all applications, featuring health status indicators that identify faulty application components and performance degradations. And, it provides a definitive root cause diagnosis of detected problems, enabling better triage and faster resolution.

In this paper, we will discuss how to use the AVIcode .NET Management Pack in conjunction with Operations Manager 2007 to:

- Isolate issues in distributed transactions through complete end-to-end performance monitoring of LOB applications
- Perform SLA monitoring for connections to 3<sup>rd</sup> party web services
- Analyze security, connectivity, coding and performance problems
- Pinpoint load balancing issues
- Leverage SQL Server Reporting Services to understand trends in historical data
- Collect event data without compromising security for sensitive information

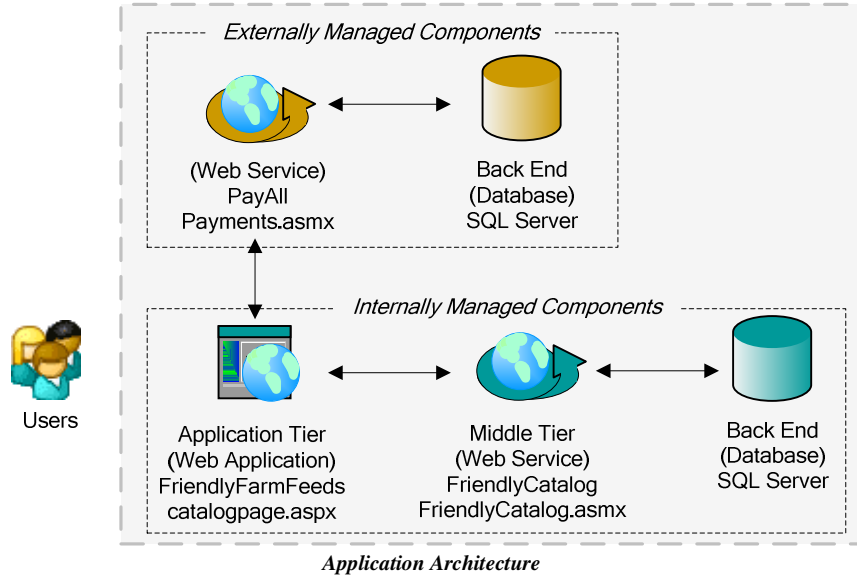
For the sake of discussion, we will look at monitoring a distributed web application developed by the mythical FriendlyCo organization, described in more detail in the next section.

---


## The Application




The FriendlyCo organization has developed a distributed web application for selling feed and supplies to farms. The application has a simple architecture as shown here:

## A Guide to Diagnosing .NET Application Health Problems with Ops Manager 2007 and the AVicode .Net Management Pack




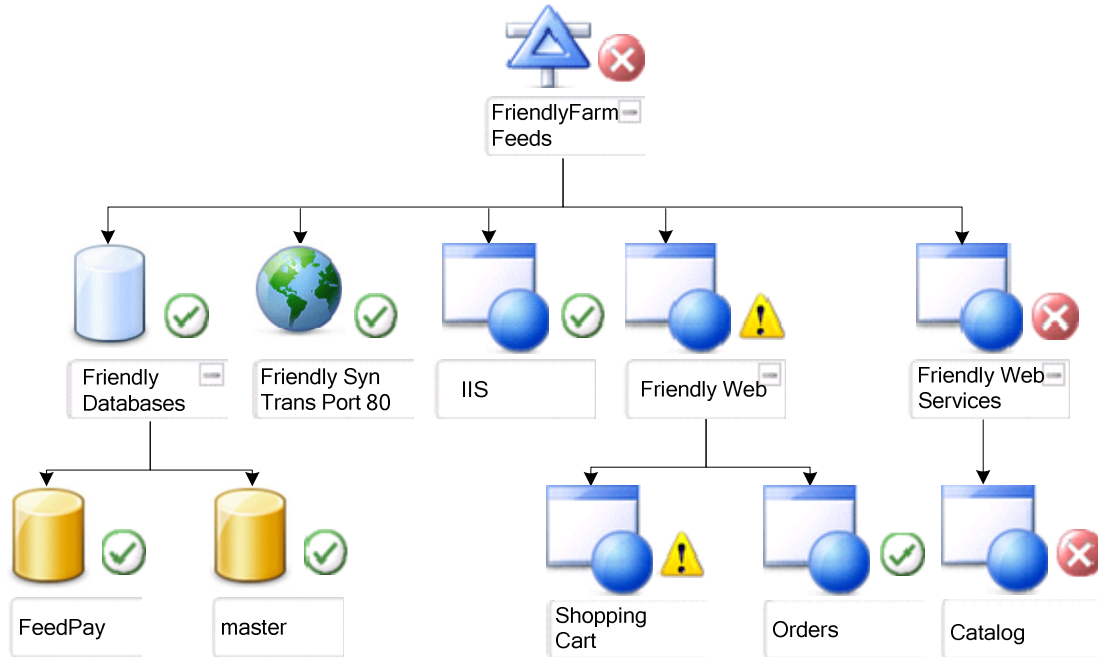
FriendlyCo has strict service level agreements that it must meet in order to keep its customer base. To meet their SLA, FriendlyCo has implemented Operations Manager 2007 and installed the AVicode .NET Management Pack to monitor their applications for slowdowns and errors.

At the highest monitoring level, the FriendlyCo NOC staff look for problems in their managed applications from the  Distributed Applications View in Operations Manager 2007. From this view, they see that the FriendlyFarmFeeds application is currently in a critical state, indicating that at least one of the monitored FriendlyFarmFeeds servers is in a critical state:


State	 Maintenance Mode	Name
 Critical		FriendlyFarmFeeds
 Healthy		FriendlyCatalog

*Distributed Applications View*









By drilling down to the  Diagram View, the operations staff can identify which of the monitored servers is in the critical state:



*Diagram View*

The server state indicates that enough events have occurred over a defined time period to warrant escalating the severity from healthy to critical. Now that they know that their SLA has been breached, FriendlyCo NOC staff need to find out whether the violation was due to code exceptions, performance problems, or both. To see what alerts put the monitored application into this critical state, they go to the  Alerts Views from the diagram node with the critical alert:

## A Guide to Diagnosing .NET Application Health Problems with Ops Manager 2007 and the AVicode .Net Management Pack

	Severity	Source	Description	Last Modified	Repeat Count
[-] Name: ASP.NET Application Performance Exception Generated (2)					
	Warning	FriendlyFarmFeeds	/FriendlyFarmFeeds/catalogpage.aspx	7/16/2007 11:24:08 AM	3
	Warning	ReportServer	Unexpected Application pool failure. A failure was encountered while launching the application.	7/17/2007 10:03:21 AM	2
	Warning	FriendlyFarmFeeds	/FriendlyFarmFeeds/shoppingcart.aspx	7/16/2007 11:16:27 AM	2
[-] Name: ASP.NET Application Exception Generated (3)					
	Critical	ReportServer	Microsoft Windows Internet Information Services 2003 Application Pool is Unavailable	7/17/2007 10:03:21 AM	2
	Critical	FriendlyFarmFeeds	/FriendlyFarmFeeds/QuestionControl.ContinueButton_Click failed in /FriendlyFarmFeeds/QuestionControl.ContinueButton_Click [System.Threading.ThreadAbortException]	7/16/2007 11:20:27 AM	5
	Critical	FriendlyFarmFeeds	/FriendlyFarmFeeds/orderpage.aspx failed in System.Web.UI.Page.ProcessRequestMain [System.Threading.ThreadAbortException]	7/16/2007 11:19:13 AM	1
	Critical	FriendlyFarmFeeds	/FriendlyFarmFeeds/shoppingcart.aspx	7/16/2007 11:18:03 AM	7

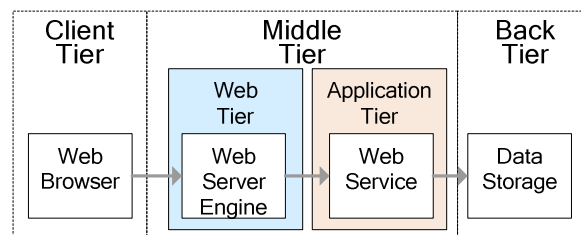
*Application Alerts View*

Now that the NOC staff knows that they have both performance and exception problems on this server, they can begin the troubleshooting process outlined in the next sections of this document.

Notice that the Diagram and Alerts views also notify the NOC staff of other infrastructure problems that have been identified by other management packs, including events from the operating system, hardware, IIS, etc. Keep in mind that these events, while falling outside of the scope of this document, all play an important role in diagnosing health problems in enterprise applications.

---


## End-to-end Performance Monitoring for LOB Applications



## A Guide to Diagnosing .NET Application Health Problems with Ops Manager 2007 and the AVicode .Net Management Pack

FriendlyCo has a service level agreement with its end users that requires that the application respond within about 5 seconds. The first step is to quantify acceptable application response times and to configure the monitor to trigger alerts when those times are exceeded. They know that the default response time for FriendlyCo must be 5 seconds or less, but certain pages have more specific requirements: the home page and the login page should respond within 2 seconds, and reporting pages should respond within 15 seconds. This means that the SLA will be defined on a broad scope at 5 seconds, but special overrides (like 2 seconds for the login page or 15 seconds for the reporting pages) must be configured for specific pages and web service calls.

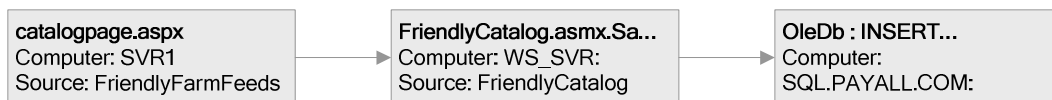
Once the SLA has been configured in System Center Operations Manager 2007 Operations Console, any breach of the agreement will generate a performance violation event via the AVicode .NET Management Pack:

Level	Date and Time	Name	Logging Computer	Description
	7/7/2007 5:37:25 PM	FriendlyFarmFeeds	SVR1	/FriendlyFarmFeeds/catalogpage.aspx?page=2&root=bovine [16,648 ms] slow at /FriendlyCatalog/FriendlyCatalog.asmx.SaveItems [16,638 ms]






*Application Event View*

Alerts represent a roll-up of all events with the same root-cause. So let's look at what information will be reported in such an event, and how that information will help FriendlyCo find where bottlenecks are occurring.

They can see that the SLA of 5 seconds has been far exceeded, because the catalog page execution took over 16 seconds. But how will FriendlyCo determine where within the transaction the slowdown occurs? The transaction starts from catalogpage.aspx, which calls SaveItems() in the FriendlyCatalog.asmx web service, which makes an OleDb call to store the information in a SQL database (as shown here):



To see the complete breakdown of the time that was spent in each tier we look at the details pane of the associated event:

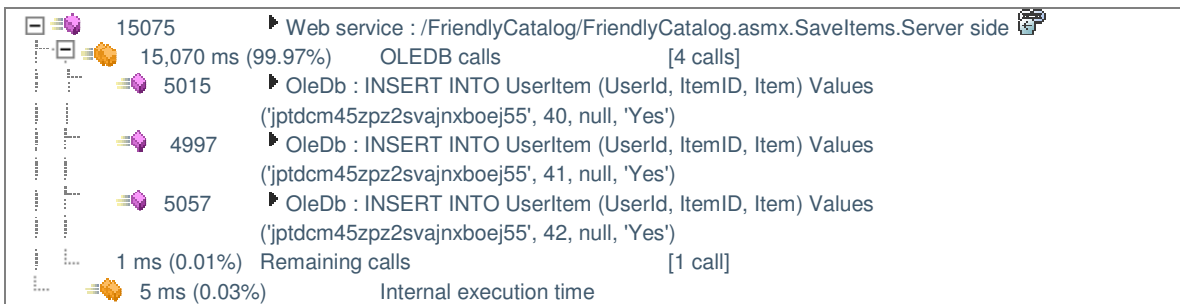
	16648	▶ ASP page : /FriendlyFarmFeeds/catalogpage.aspx	
	16,638 ms (99.94%)	Web Services calls [1 call]	
	16638	▶ Web service : /FriendlyCatalog.asmx.SaveItems.Client side	
	10 ms (0.06%)	Internal execution time	

*Application Event Details - call to local Web Service*

The event details show that one web service call to FriendlyCatalog took up 16,638 ms, or 99.94%, of the total 16648 ms page execution time. That shows that the problem isn't in FriendlyFarmFeeds itself, but somewhere between the start and end of the web service call.

## A Guide to Diagnosing .NET Application Health Problems with Ops Manager 2007 and the AVicode .Net Management Pack

So the next step is to look at the event that was generated from the web service in the middle tier:



*Web Service Event Details*

This shows that the bottleneck is not due to transport time between the tiers, because the client side call indicated a time of 16,638 ms to the web service, and 15,075 ms of that is actually being taken up by processing in the web service, which leaves roughly 1.5 seconds of overhead including transport time. So the problem must be in the web service itself.

Looking at the event details, 99.97% of the total execution time is taken up by OLEDB calls, and all but 1 ms is due to the three database INSERT statements. Not only that, but the details display the exact statements, including that data values, that were responsible for the slow execution time. This is precisely the information that FriendlyCo's DBA will require in order to troubleshoot the problem.

Keep in mind that although the FriendlyCo example has an ASP.NET application front-end talking to a Web Service on the middle tier talking to a SQL database on the back-end, the AVicode .Net Management Pack supports a wide variety of enterprise technologies.

For example, instead of an ASP.NET application on the front-end, it could have just as easily been a WinForms or COM+ application, and the Web Service in the middle tier could have been .NET Remoting or a Windows Service application.

Although the FriendlyCo application example uses a SQL back-end, the .NET Management Pack supports a complete range of databases including Microsoft SQL, Oracle, Sybase, DB2, etc. Because the pack comes pre-configured for the standard Microsoft OLE DB and ODBC drivers, it can support virtually any database. Additionally, the configuration files can be customized to provide support for other native drivers and their associated databases.

---

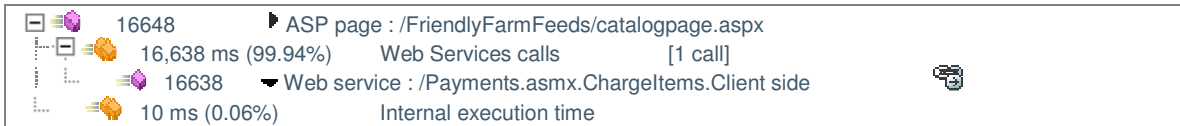
## SLA Monitoring for Connections to 3<sup>rd</sup> Party Web Services

In our previous example of end-to-end performance monitoring, we saw that the web service calls went to a monitored middle tier that is also the responsibility of the FriendlyCo developers. But what if the web service request goes to another FriendlyCo

## A Guide to Diagnosing .NET Application Health Problems with Ops Manager 2007 and the AVicode .Net Management Pack

division's data center over which they have no control? Or if the request goes to a 3<sup>rd</sup> party vendor who is breaching the SLA while the FriendlyCo developers got the blame for it?

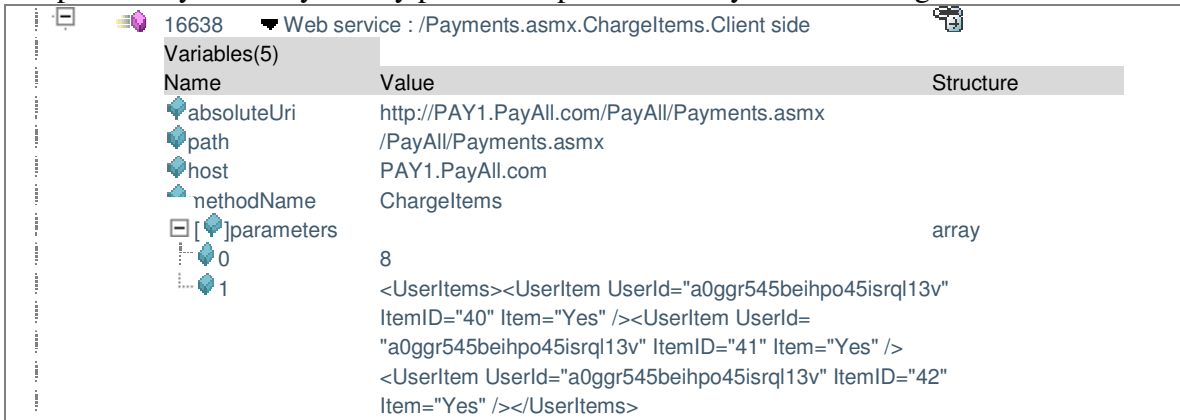
While they were able to monitor their own catalog web service to collect information about slow processing, it is unlikely that the remote 3<sup>rd</sup> party web services for collecting payments will be monitored. So let's look at what happens when the slowdown is in the outside party's web service:



16648	ASP page : /FriendlyFarmFeeds/catalogpage.aspx
16,638 ms (99.94%)	Web Services calls [1 call]
16638	Web service : /Payments.asmx.ChargeItems.Client side
10 ms (0.06%)	Internal execution time

*Call Times to 3<sup>rd</sup> Party Web Service*

We can easily see that only 0.06% of the time is taken up by our own code processing, and that 99.94% of the time is taken up by the vendor's web service - hard evidence that FriendlyCo's application is not responsible for the SLA breach. Of course assigning responsibility correctly is only part of the problem - they also need to get the issue fixed.



Name	Value	Structure
absoluteUri	http://PAY1.PayAll.com/PayAll/Payments.aspx	
path	/PayAll/Payments.aspx	
host	PAY1.PayAll.com	
methodName	ChargeItems	
parameters	8	array
0	<UserItems><UserItem UserId="a0ggr545beihpo45isrq13v" ItemID="40" Item="Yes" /><UserItem UserId="a0ggr545beihpo45isrq13v" ItemID="41" Item="Yes" />	
1	<UserItem UserId="a0ggr545beihpo45isrq13v" ItemID="42" Item="Yes" /></UserItems>	

*Application Event Details for Call to 3<sup>rd</sup> Party Web Service*

The AVicode .NET MP allows the NOC team to drill down into the details of the call to the external web service. The NOC team can then see the exact values used in the web service request, and that data can easily be passed on to whoever is responsible for correcting the problem.

## Security Problems: Detection and Root-cause Determination

For security problems, the AVicode .Net Management Pack provides the FriendlyCo team with details about what credentials were being used, what resources they were trying to access, and what permissions they are missing - in other words, exactly the information needed to troubleshoot a security problem. This is critically important for applications that use impersonation and applications that use integrated security because it is not always clear what credentials will be used when accessing the database.

## A Guide to Diagnosing .NET Application Health Problems with Ops Manager 2007 and the AVIcode .Net Management Pack

Security failure in "System.Data.SqlClient.SqlConnection.OnError" has been detected: "EXECUTE permission denied on object 'usp\_ResumeContents', database 'FeedPay', schema 'dbo'".

Name	Value
Target	
SQLServer/Database	SQL.PayAll.com/FeedPay
Actions	
Query	usp_ResumeContents
Security context	
SQLServer account	Server=SQL.PayAll.com;database=FeedPay;uid=FeedPay;pwd=*****;
Thread identity	NT AUTHORITY\NETWORK SERVICE

### *Security Problem Description*

In this example, they can immediately see that the FeedPay uid does not have execute permissions on the usp\_ResumeContents query. If the problem was in permission for a table, then the name of the table in question would be displayed.

---

## Connectivity Problems: Detection and Root-cause Determination

For connectivity problems with internal or external web services, the AVIcode .NET Management Pack provides details for diagnosing the problem:

Connectivity failure in "System.Net.HttpWebRequest.GetResponse" has been detected: "The remote server returned an error: (404) Not Found."

Name	Value
Target	
AbsoluteUri	http://PAY1.PayAll.com/PayAll/Payments.asmx
Actions	
Get Response	System.Net.HttpWebRequest.GetResponse
Security context	
Thread identity	PAY1.PayAll.com\Administrator

### *Connectivity Problem Description*

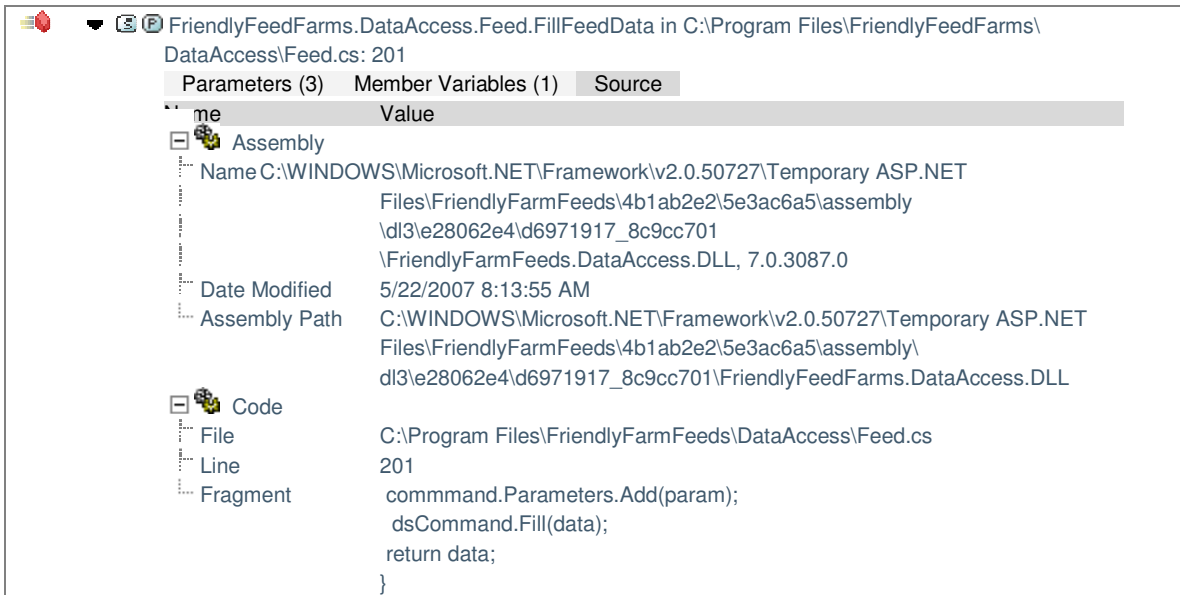
The details include to what the connection was targeted, what action was being performed that failed, and the security context of the connection. This example shows what address (<http://PAY1.PayAll.com/PayAll/Payments.asmx>) was being targeted when the server returned a file not found exception (404).

---

## Coding Problems: Detection and Root-cause Determination

Sometimes the FriendlyCo developers are responsible for coding errors, and when that happens they need to track down the source of the problem as quickly as possible:

## A Guide to Diagnosing .NET Application Health Problems with Ops Manager 2007 and the AVicode .Net Management Pack



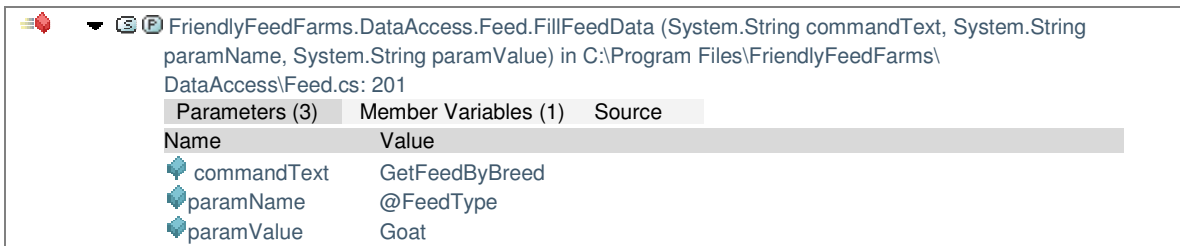
The screenshot shows a table with columns for Name and Value. It details the assembly path and the source code fragment for the error.

Name	Value
Assembly	Name C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\Temporary ASP.NET Files\FriendlyFarmFeeds\4b1ab2e2\5e3ac6a5\assembly\dl3\e28062e4\d6971917_8c9cc701\FriendlyFarmFeeds.DataAccess.DLL, 7.0.3087.0
Date Modified	5/22/2007 8:13:55 AM
Assembly Path	C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\Temporary ASP.NET Files\FriendlyFarmFeeds\4b1ab2e2\5e3ac6a5\assembly\dl3\e28062e4\d6971917_8c9cc701\FriendlyFeedFarms.DataAccess.DLL
Code	File C:\Program Files\FriendlyFarmFeeds\DataAccess\Feed.cs
Line	201
Fragment	command.Parameters.Add(param); dsCommand.Fill(data); return data; }

*Coding Error Details - Source Code*

The AVicode .Net Management Pack provides the developers with details about the error including: the name of the source file, the line number in the file, a fragment of the failed code, and build details about the assembly with the problem.

Additionally, they are able to see the exact values that were being passed as parameters into the failing function or stored as member variables (not shown):



The screenshot shows a table with columns for Name and Value. It lists the parameter values passed to the failing function.

Name	Value
commandText	GetFeedByBreed
paramName	@FeedType
paramValue	Goat

*Coding Error Details - Parameter Values*

---

## Performance Problems: Detection and Root-cause Determination

### Slow Database Issues

As shown in an earlier example, the FriendlyCo DBA can track down database issues by using the exact database query provided in the AVicode .NET Management Pack performance event details:

## A Guide to Diagnosing .NET Application Health Problems with Ops Manager 2007 and the AVIcode .Net Management Pack

1067 ms Web service : /FriendlyCatalog/FriendlyCatalog.asmxGetMatchingItems.Server side

1,000 ms (93.72%) OLEDB calls [4 calls]

1000 Oledb :SELECT TOP 20 ItemProperty.ITMPRO\_ITEM\_CODE AS ItemCode,  
COUNT(ItemProperty.ITMPRO\_ITEM\_CODE) As NrPropsThatMatch, 0 AS PercentMatch  
FROM ItemProperty INNER JOIN Property ON ItemProperty.ITMPRO\_PROPERTY\_CODE  
= Property.PRO\_CODE WHE

Variables(2)		
Name	Value	Structure
cmdText	SELECT TOP 20 ItemProperty.ITMPRO_ITEM_CODE AS ItemCode, COUNT(ItemProperty.ITMPRO_ITEM_CODE) As NrPropsThatMatch, 0 AS PercentMatch FROM ItemProperty INNER JOIN Property ON ItemProperty.ITMPRO_PROPERTY_CODE = Property.PRO_CODE WHERE ((ItemProperty.ITMPRO_VAL_VALUE= 'GoatFeed') AND (Property.PRO_NAME = 'City')) OR ((ItemProperty.ITMPRO_VAL_VALUE= 'Buy') AND (Property.PRO_NAME = 'Delivery')) OR ((ItemProperty.ITMPRO_VAL_VALUE= 'Yes') AND (Property.PRO_NAME = 'Aircoc')) OR ((ItemProperty.ITMPRO_VAL_VALUE= 'Yes') AND (Property.PRO_NAME = 'SweetGrain')) OR ((ItemProperty.ITMPRO_VAL_VALUE= 'Yes') AND (Property.PRO_NAME = 'LoadingDock')) GROUP BY ItemProperty.ITMPRO_ITEM_CODE ORDER BY COUNT(ItemProperty.ITMPRO_ITEM_CODE) DESC	
parameter	Empty Collection	

67 ms (6.28%) Internal execution time

*SQL Query*

### Slow Application Code Issues

Other types of performance problems also become evident when an inordinate amount of time spent on a single function is detected or a single function is being called too many times. In this example, the FriendlyCo developers can see that not only is the call to GetMatchingItems() slow, but that it is also being called twice with the same values, an obvious coding problem:

28838 ms ASP page : /FriendlyFarmFeeds/catalogpage.aspx

28789 ms FriendlyFarmFeeds.CatalogControl.Page\_Load()

28785 ms FriendlyFarmFeeds.CatalogControl.FillResultSet()

28757 ms **FriendlyFarmFeeds.CatalogControl.GetMatchingResult()**

127 ms FriendlyFarmFeeds.CatalogControl.ProductDef()

61 ms FriendlyFarmFeeds.CatalogControl.LoadCriteria()

3512 ms FriendlyFarmFeeds.CatalogControl.Matching.GetMatchingItems()

3512 ms **FriendlyFarmFeeds.CatalogControl.Matching.GetMatchingItems**

Variables(2)		
Name	Value	Structure
feedType	2	
payMethod	3	
volume	50	

3512 ms FriendlyFarmFeeds.CatalogControl.Matching.GetMatchingItems()

3512 ms **FriendlyFarmFeeds.CatalogControl.Matching.GetMatchingItems**

Variables(2)		
Name	Value	Structure
feedType	2	
payMethod	3	
volume	50	

*Function Calls and Function Values*

## Server Resource Utilization Issues

While some performance problems are due to coding issues, others result from server or application load. With the AVIcode .NET Management Pack, FriendlyCo can analyze complete information about resource utilization on each monitored server at the very moment a problem occurs:

Computer Name	CPU (% CPU Time)		Memory (MB)			I/O (kB/sec)		Application Load	
	Total	Applications	Available	Total	Applications	Total	Applications	Requests	Events
FRIENDLY CO.COM \SVR1	98 ▲	78 ▲	394 ▲	1,051 ▲	112 ▲	265 ▲	4 ▲	9 ▲	37 ▲
FRIENDLY CO.COM \WS_SVR1	20 ▼	14 ▲	500 ▲	1,022 ▲	100 ▲	125 ▼	1 ▲	2 ▲	28 ▼

*Computer Resource Utilization for All Servers*

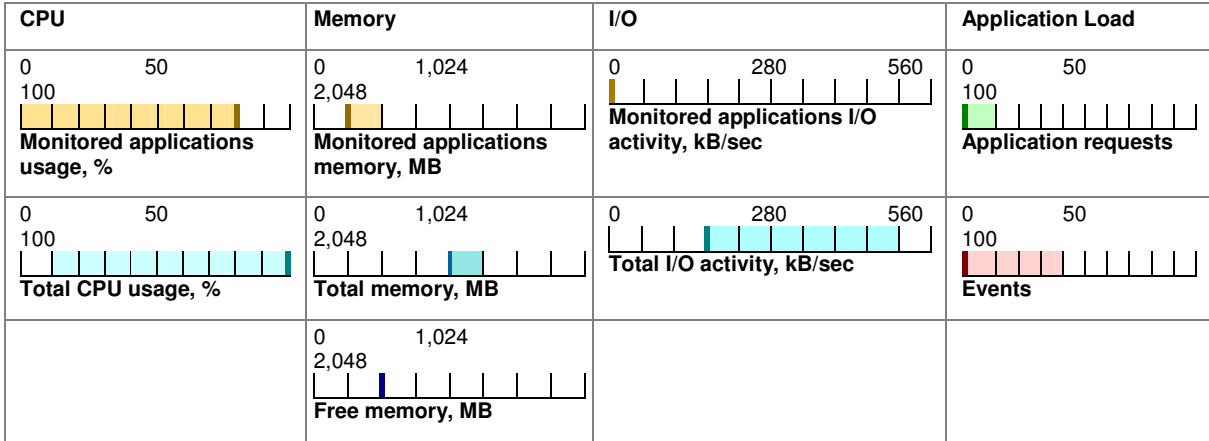
This table displays the exact load on each server for the previous hour during the slowdown, and shows server utilization from the applications vs. the total server utilization. Utilization is broken down into four aspects: CPU, Memory, I/O and application load. Within each, values are given for both the total server load and the amount that is being used by the monitored applications. In this example, these values indicate that there may be a CPU problem, since SVR1's CPU was running at 98% when the performance problems began, and that 78% of the CPU is being used up by the monitored applications. Additionally, the up arrows (▲) indicate that the percentage has gone up since the previous data reading, perhaps indicating that the problem is getting worse. It does not appear that they are having any problems with memory, I/O or application load.

Both FriendlyCo's NOC staff and their developers will be interested in analyzing the metrics supplied by the AVIcode .NET Management Pack, since it can point to either system-related issues or to application-related issues.

### *Monitored Server Key Metrics*

By looking at the Computer Resource Utilization information in the previous section, FriendlyCo is able to determine the precise aspect (CPU, Memory, I/O or Application Load) of the problem. As we saw in the last example, the problem seems to be CPU utilization. The developers then use the AVIcode .NET Management Pack to look at the Key Metrics data, which provides a graphical view of the same data, but only for the server that they are interested in (SVR1):

## A Guide to Diagnosing .NET Application Health Problems with Ops Manager 2007 and the AVicode .Net Management Pack



Name	Instance	Average	Today's peak
\\.NET Apps\\Avg. Request Time	_All monitored applications	0	22,436
\\.NET Apps\\Exception Events	_All monitored applications	0	29
\\.NET Apps\\Monitored Requests	_All monitored applications	0	9
\\.NET Apps\\Performance Events	_All monitored applications	0	9
\\ASP.NET Apps\\Requests Queued	_Total	0	0
\\Memory\\Available MBytes	_Total	318	467
\\Process\\IO Data Bytes/sec	_Total	212 kB/sec	550 kB/sec
\\Process\\IO Data Bytes/sec	_All Monitored Applications	3 kB/sec	10 kB/sec
\\Process\\Private Bytes	_Total	1,126 MB	1,132 MB
\\Process\\Private Bytes	_All Monitored Applications	119 MB	120 MB
\\Process\\Virtual Bytes	_All Monitored Applications	337 MB	339 MB

*Key Metrics by Server*

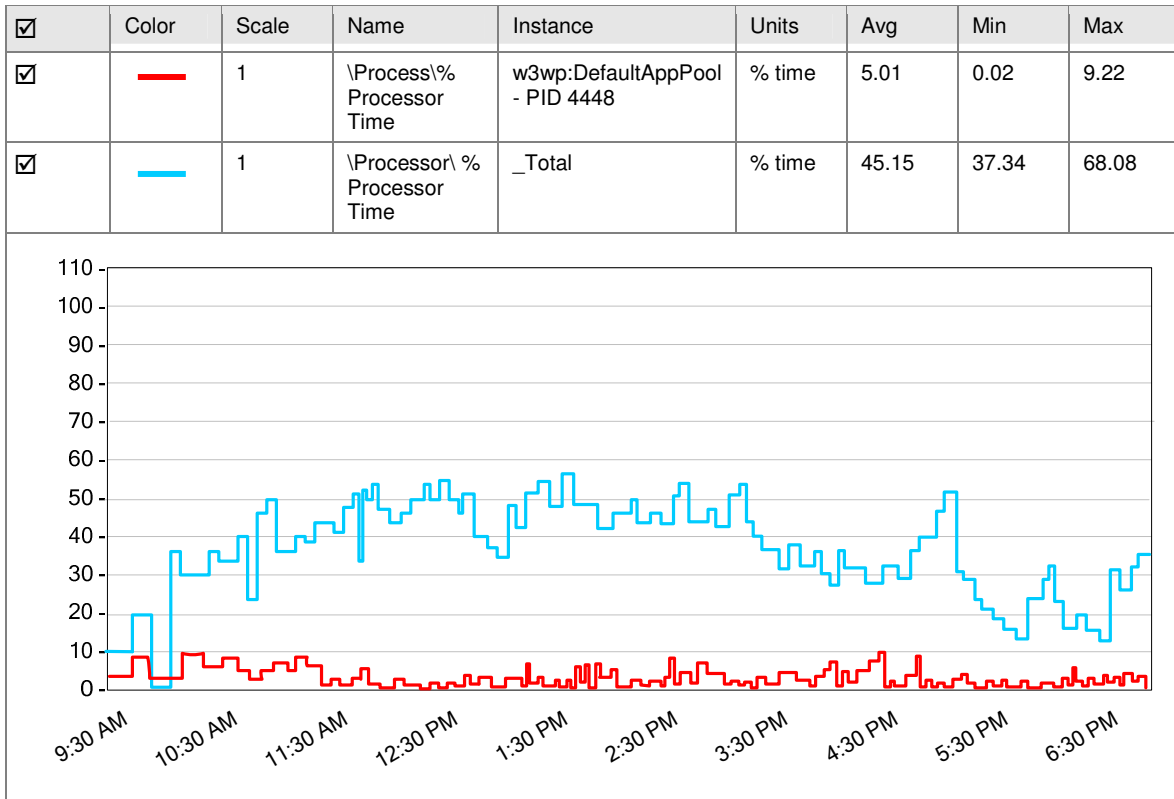
Whereas the Resource Utilization table provided data for the previous hour, the Key Metrics table provides a colored horizontal bar that displays the minimum and maximum range of values from the previous 24 hours. The graphical display makes it easy for the FriendlyCo developers to see that the monitored applications' CPU usage fluctuated between 0 and 80%, and does not seem to have been affected by memory, I/O or application load, all of which remained relatively low and constant over the same time period.

From there, the developers want to find out if the resources are being consumed by their application, or if they are being impacted by some other monitored component. To do this, they will look at the application-specific data collected by the AVicode .NET Management Pack.

### ***Monitored Server Trends***

The FriendlyCo developers look at the Trend reports for SVR1. Trend reports allow you to track various performance counters across the last 10 sampling periods. In this case they look at 2 counters - Process\\% Processor Time and Processor\\% Processor Time, for SVR1.

## A Guide to Diagnosing .NET Application Health Problems with Ops Manager 2007 and the AVicode .Net Management Pack



*Trend Reports by Server*

The Process % Processor Time counter (in blue) gives the percentage of the time that is processor time consumed by ASP.NET. The developers know how much is being used by their application by checking the instance column to see which application pool the counter is reporting for. When you compare performance under standard load against a previously captured baseline, decreases in this counter reveal lower processor requirements, and therefore greater scalability. Values for this counter greater than 70% for extended periods of time indicate a need to purchase hardware or optimize your application.

The Processor\% Processor Time (in red) is the percentage of elapsed time that the processor spends to execute a non-idle thread for all processed threads. This counter is the primary indicator of processor activity, and displays the average percentage of busy time observed during the sample interval. High values may not necessarily be bad. However, if the other processor-related counters are increasing linearly such as % Privileged Time or Processor Queue Length, high CPU utilization may be worth investigating.

By comparing the amount of processor time that their application is using in the default application pool against the total processor utilization, they are able to determine that their application is not the one burning up all of the CPU cycles.

### ***Monitored Server Worker Processes***

At this point, the FriendlyCo developers have been able to determine that the excessive CPU load was coming from somewhere, but not from their application. Therefore it is

## A Guide to Diagnosing .NET Application Health Problems with Ops Manager 2007 and the AVicode .Net Management Pack

important for them to understand what processes are running on the box, and how they may be affecting the machine load:

Name	Process Up Time	Statistics (Previous Hour)				Applications	Additional Information
		% CPU Time	.NET Exceptions thrown/sec	Private bytes	.NET bytes		
w3wp#1	1:59:38	1	0.1	119 MB	22 MB	DefaultDomain	PID: 5996 Monitor: operation Framework: 2.0.50727.42 APPPool: DefaultAppPool
w3wp#2	2:23:08	3	0.5	85 MB	10 MB	FriendlyFarmFeeds	PID: 5985 Monitor: operation Framework: 2.0.50727.42 APPPool: Feeds
w3wp#3	0:12:15	70	0.1	185 MB	12 MB	FriendlyCatalog	PID: 5999 Monitor: operation Framework: 2.0.50727.42 APPPool: Catalog

*Monitored Processes by Server*

Assuming that each LOB application runs under its own application pool, and that each application pool runs under its own w3wp process, we immediately get a clear picture of what is consuming resources on the machine. This information shows exactly which pool and PID the process runs under (eliminating the difficult task of manually correlating the process, pool and PID), and how much of the resources it is consuming. For each process, the view displays the CPU load, the rate of .NET exceptions thrown per second, the full memory utilization, and what portion of that memory is being used by .NET. The FriendlyCo developers had already determined that their application running under the DefaultAppPool was not the problem, and now they can see that the problem is with the Catalog application pool, which is using 70% of the CPU resources.

### *Application Statistics*

The FriendlyCo developers started with the big picture of the Key Metrics for each of the 4 aspects (CPU, Memory, I/O and Application Load), and then drilled down into the trends for each aspect. They first determined that there was a CPU utilization problem on SVR1, and then that the problem was in a monitored application. They eliminated their application as the problem, and then determined which application pool was running the application that was using too many CPU cycles.

The next step then is to look at the application statistics:

Name	Statistics (Previous Hour)				Current Load	
	Monitored Requests	Avg Time (ms)	Exception Events	Performance Events	Sessions	Requests/sec
FriendlyFarmFeeds	612	612	0	2	5	8
FriendlyCatalog	200	16300	3	17	1	3

*Application Resource Utilization for Applications by Server*

Now, the FriendlyCo developers are able to gain insight into the exact load on each of the applications on SVR1 for the previous hour, and can see that while the FriendlyCatalog

application does not have a particularly heavy load, that it does seem to be unusually slow, and it has generated 17 performance events.

---

## Application Load Balancing and Performance Issues

So far we have covered performance problems from the standpoint of: the database, an external web server, resource utilization and coding performance deficiencies. When analyzing performance problems, it is important to understand whether that problem is unique to the server with the problem or if it applies to multiple servers. The question becomes "What is my distribution of the load between multiple servers, and how does my load balancer work?" For any selected application, the AVIcode .NET Management Pack provides a picture of your total application load across each server running the application:

Name	Statistics (Previous Hour)				Current Load	
	Monitored Requests	Avg Time (ms)	Exception Events	Performance Events	Sessions	Requests/sec
FRIENDLYCO.COM\SVR1	212	22	6	5	10	8
FRIENDLYCO.COM\SVR2	2210	622	15	22	35	78

*Load Balancing Statistics by Computer for an Application*

This allows you to see how balanced the application load is, and to see if the monitored server with the application issues is experiencing any unusual load in terms of the number of sessions or requests.

---

## Reporting

The AVIcode .NET Management Pack provides the ability to use SQL Server Reporting Services to create, manage and deliver both traditional, paper-oriented reports and interactive, Web-based reports based on stored event data. It comes with sample .rdl files that are briefly described below:

## A Guide to Diagnosing .NET Application Health Problems with Ops Manager 2007 and the AVicode .Net Management Pack

Report Type	Report Description
Application Daily State Monitoring Report	For any selected application performance counter and any selected date, this report displays today's average value for that hour and compares it to historical average values for that same hour by day of the week.
Application Hourly State	For any selected application performance counter, this report displays today's average values by hour and compares it to historical average values for each hour of the day.
Com Plus Hotspots	For the slowest COM+ calls from each .NET application, this report provides details about the COM+ object, actual method called, average call time and maximum call time.
Computer Daily State	For any selected computer performance counter and any selected date, this report displays today's average value for that hour and compares it to historical average values for that same hour by day of the week.
Computer Hourly State	For any selected computer performance counter, this report displays today's average values by hour and compares it to historical average values for each hour of the day.
Exception Events	This report shows the most frequently occurring exception events for selected application.
Performance Events	This report shows the most frequently occurring performance events for selected application.
Remoting Hotspots	For the slowest .NET Remoting calls from each .NET application, this report provides details about the remote object, actual method called, average call time and maximum call time.
SQL Hotspot	For the slowest SQL query from each .NET application, this report provides details about the calling page, SQL command executed, average call time and maximum call time.
WCF Hotspots	For the slowest Windows Communication Foundation (MS.NET 3.0) calls from each .NET application, this report provides details about the actual method called, average call time and maximum call time.
Web Service Hotspots	For the slowest web services calls from each .NET application, this report provides details about the web service, actual method called, average call time and maximum call time.

There are three basic types of reports. The first two types are daily and hourly statistics, both of which are based on performance counter data. These reports are useful for determining the best day and time to do server maintenance and upgrades when server utilization and application load are at a minimum. The third type of report is based on events from resource calls (SQL, .NET Remoting, etc), that help you to determine what functionality is failing in your applications.

Application	Operation	Command	Avg time, s	Max duration, s
FriendlyCatalog	/FriendlyCatalog/FriendlyCatalog.asmx.Saveltems	OleDb : INSERT INTO UserAnswer (UserId, QuestionId, AnswerId, Answer) VALUES('kbeulf55sqzudnbnijxf4vfq', 32, 80, 'GoatFeed')	5.22	5.4
FriendlyCatalog	/FriendlyCatalog/FriendlyCatalog.asmx.Saveltems	OleDb : INSERT INTO UserAnswer (UserId, QuestionId, AnswerId, Answer) VALUES('3z5ts5ibxedmyi55temntlzu', 40, null, 'Yes')	5.17	5.17
FriendlyCatalog	/FriendlyCatalog/FriendlyCatalog.asmx.Saveltems	OleDb : INSERT INTO UserAnswer (UserId, QuestionId, AnswerId, Answer) 5.14VALUES('jptdcm45zpz2svajnxboei55', 32, 80, 'GoatFeed')	5.14	5.14

*SQL Hotspot report*

## Data Protection

The AVIcode .NET Management pack provides multiple security features to protect sensitive information. These features include:

### *Sensitive Field Recognition*

When the AVIcode .NET Management Pack receives data, it automatically recognizes sensitive fields (like passwords), and removes that information before it ever gets stored in the system. The information is replaced with asterisks in the event details:

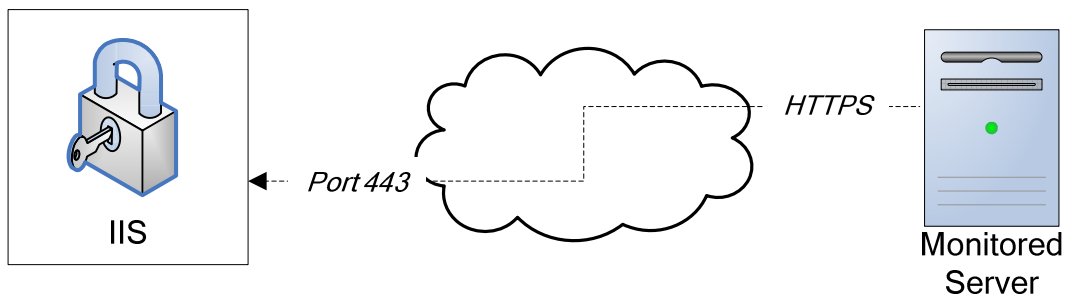
```
Security failure in "System.Data.SqlClient.SqlInternalConnection.OnError" has been detected: "Login failed for user 'FeedPay'. Reason: The password*****".
```

me	Value
Target	
SQLServer/Database	SQL.PayAll.com/FeedPay
Actions	
Method	Open
Security context	
ConnectionString	Server=SQL.PayAll.com;database=FeedPay;uid=FeedPay;pwd=*****;
Thread identity	NT AUTHORITY\NETWORK SERVICE

*Example of Sensitive Data Protection - Protected Password in a Connection String*

### *HTTPS*

Data being sent from the monitoring agents can be protected during the delivery phase by using the Secure Socket Layer (HTTPS):

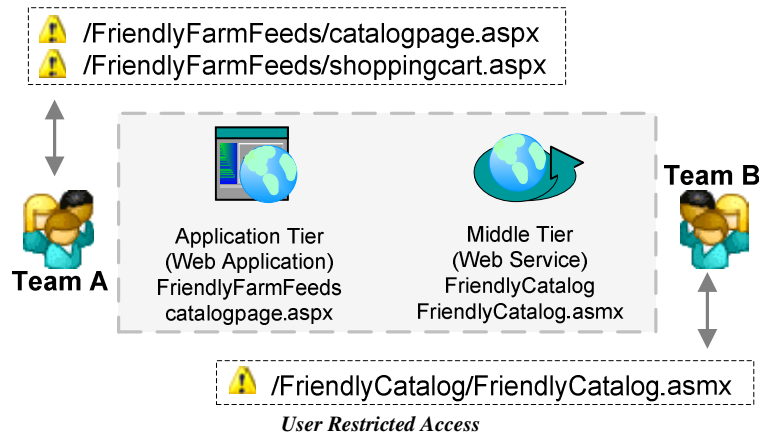


*HTTPS Data Protection*

### *Application Groups*

Another way that the AVIcode .NET Management pack protects sensitive data is by restricting which users have access to which data. For example, FriendlyCo can restrict Team A (which developed the application tier) to only see events from FriendlyFarmFeeds, while Team B (which developed the middle tier) can only see events generated by FriendlyCatalog:

## A Guide to Diagnosing .NET Application Health Problems with Ops Manager 2007 and the AVIcode .Net Management Pack



---

### Conclusion

The AVIcode .NET Management Pack for Operations Manager 2007 extends System Center Operations Manager, enabling comprehensive application monitoring and real-time problem diagnosis. In this technical document, we show how the AVIcode .NET Management Pack can detect an SLA threshold violation for an ASP.NET application. Once detected, the AVIcode .NET Management Pack also provides a mechanism for tracking the problem through the web service on the application server all the way to the database tier, providing the precise information needed for a DBA to successfully troubleshoot the problem. The AVIcode .NET Management Pack provides complete root cause details and insight into overall system behavior that enables rapid issue resolution, even when the problem is related to an un-monitored 3<sup>rd</sup> party component.

Similarly, the AVIcode .NET Management Pack detects and diagnosis security, connectivity and coding problems. It provides definitive root cause information, such as execute permissions for queries, the security context for connections, source file names, and fragments of failed code. This detailed information enables problem resolution.

In the case of performance problems, the AVIcode .NET Management Pack enables you to obtain the exact database queries and pinpoint functions that are either running slowly or are being called too frequently. This data can be then correlated with resource utilization data (CPU, Memory, I/O, or Application Load) to enable a deeper understanding of the error. Further insight is obtained by drilling into key metrics and examining trend reports for pertinent performance counters. This level of detail and understanding, along with the ability to view values for worker processes and individual applications, empowers operations personnel to definitively determine whether or not performance issues are related to server resources. This saves time in identifying and triaging performance problems, and can alleviate the need to drill into source code to determine the cause of the performance degradation.

In addition to helping to detect, diagnose, and resolve application problems, the AVIcode .NET Management Pack also allows organizations to analyze historical data by leveraging SQL Server Reporting Services. And, throughout the troubleshooting and review process, the AVIcode .NET Management Pack provides multiple mechanisms for ensuring data confidentiality and security of sensitive information.

## **Other System Center 2007 Solutions from AVIcode**

### ***AVIcode BizTalk Server 2006 Management Pack***

The AVIcode BizTalk Server 2006 Management Pack extends the native monitoring capabilities of System Center Operations Manager 2007 by enabling organizations to detect and diagnose problems related to their BizTalk applications. Functionality includes detecting failures due to erroneous .NET code activities within BizTalk Orchestrations and Pipelines, pinpointing data processing errors inside XSLT transformations, security and connectivity problems from BizTalk Adapter configuration errors, and much more.

### ***AVIcode Office SharePoint Server 2007 Management Pack***

Together with the Microsoft Office SharePoint Server 2007 Management Pack, the AVIcode Office SharePoint Server Management Pack enables a complete SharePoint monitoring solution. The AVIcode Management Pack monitors all Microsoft Office SharePoint Server 2007 components, including Enterprise Search, Excel Services, and Office Forms Server 2007. Some of the features are: detecting security and connectivity problems from component (e.g. Business Data Catalog, Office Forms Server 2007, Excel Services, Enterprise Search Calls, etc) configuration errors, coding errors within custom Workflow classes, performance monitoring of BI Dashboards based on KPI, and more.

### ***AVIcode Reporting Services Management Pack***

The AVIcode Reporting Services Management Pack ensures that Reporting Services solutions are always available and reliable. It extends the monitoring capabilities of System Operations Manager 2007 by detecting and diagnosing problems affecting custom reports rendered by Reporting Services, including problems with SQL queries and both connectivity and security issues. Other features include: extensible reporting on reporting web service usage, monitoring for security and connectivity problems related to configuration errors (for report data sources, networks, SQL Server, etc), real-time identification of performance bottlenecks related to long-running SQL queries, and more.

Contact [sales@avicode.com](mailto:sales@avicode.com) for more information about these management packs.