Portrait
Foundation

Troubleshooting Guide - Initial Self Help

Edition 1.1

16 January 2013

Pitney Bowes
Software

Portrait Foundation
Troubleshooting Guide - Initial Self Help

About Portrait Software

Portrait Software is now part of Pitney Bowes Software Inc.

Portrait Software enables organizations to engage with each of their customers as individuals, resulting in improved customer profitability, increased retention, reduced risk, and outstanding customer experiences. This is achieved through a suite of innovative, insight-driven applications which empower organizations to create enduring one-to-one relationships with their customers.

Portrait Software was acquired in July 2010 by Pitney Bowes to build on the broad range of capabilities at Pitney Bowes Software for helping organizations acquire, serve and grow their customer relationships more effectively. The Portrait Customer Interaction Suite combines world leading customer analytics, powerful inbound and outbound campaign management, and best-in-class business process integration to deliver real-time customer interactions that communicate precisely the right message through the right channel, at the right time.

Our 300 + customers include industry-leading organizations in customer-intensive sectors. They include 3, AAA, Bank of Tokyo Mitsubishi, Dell, Fiserv Bank Solutions, Lloyds Banking Group, Merrill Lynch, Nationwide Building Society, RACQ, RAC WA, Telenor, Tesco Bank, T-Mobile, Tryg and US Bank.

Pitney Bowes Software Inc. is a division of Pitney Bowes Inc. (NYSE: PBI).

For more information please visit: http://www.pitneybowes.co.uk/software/

UK

Portrait Software
The Smith Centre
The Fairmile
Henley-on-Thames
Oxfordshire, RG9 6AB, UK

Email: support@portraitsoftware.com
Tel: +44 (0)1491 416778
Fax: +44 (0)1491 416601

America

Portrait Software
125 Summer Street
16th Floor
Boston, MA 02110
USA

Email: support@portraitsoftware.com
Tel: +1 617 457 5200
Fax: +1 617 457 5299

Norway

Portrait Software
Portrait Million Handshakes AS
Maridalsveien. 87
0461 Oslo
Norway

Email: support@portraitsoftware.com
Tel: +47 22 38 91 00
Fax: +47 23 40 94 99

About this document

This document is provided as a guide on how to approach the investigation of problems with a Portrait Foundation implementation.

Purpose of document

It is a self-help guide for investigating problems with and gathering information about Portrait Foundation implementations. It contains a collection of problem solving techniques and describes the tools required for investigating issues in Portrait Foundation and EDGE2020.

Intended audience

System administrators, Portrait Foundation implementation administrators, developers and support staff. This is not a Portrait Foundation Training Document. It is assumed that the reader already has some familiarity with Portrait Foundation.

Related documents

Technical Architecture (*Technical_Architecture.pdf*)

Problem Determination Overview (*Problem_Determination_Overview.pdf*)

Logging Usage and Standards Guide (*Logging_Usage_and_Standards_Guide.pdf*)

Operations Guide (*Operations_Guide.pdf*)

Model and Node Counters Reference (*Model_and_Node_Counters_Reference.pdf*)

Request Token Tracing Overview (*Request_Token_Tracing_Overview.pdf*)

Software release

Portrait Foundation 4.4 and later, but generally this document is applicable to all versions of Portrait Foundation.

Contents

1 About Portrait Foundation

Portrait Foundation is a function-rich integrated software platform which is used to build customized solutions which bring together data from disparate systems. Due to the nature of such a solution, all related systems must be functioning properly at all times. Problem determination in such environments can be very complicated, and require a blend of skills, experience and logical thinking.

Portrait Foundation can function as a web service which provides information to other systems, but often it's the agent facing front end to a complex solution involving, in some cases, many disparate systems. Portrait Foundation users may be the first to experience the symptoms of problems occurring anywhere in the architecture of the solution because it ultimately has an impact on the User Interface (UI). From the Agent Perspective, every problem will appear to be a problem with Portrait Foundation, but often this is not the case. Presupposing that the problem is being caused by a bug in Portrait Foundation can considerably delay the resolution of the problem.

Portrait Foundation can also serve as a service, or content provider for other systems via web services.

1.1 Typical Enterprise Infrastructure

The message here is that a Portrait Foundation solution can be very complicated. In all cases it will be highly dependent on your network infrastructure. While it's not possible to show all possible network architectures, the diagrams below show some typical examples.

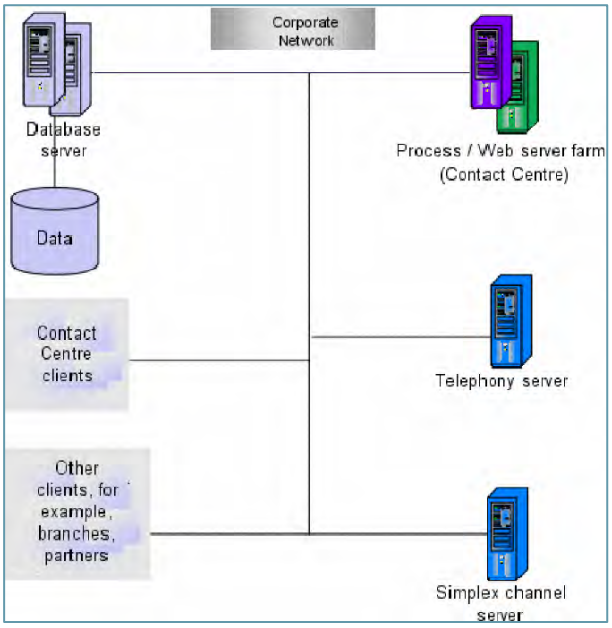


Figure 1 - Corporate network (LAN) Structure

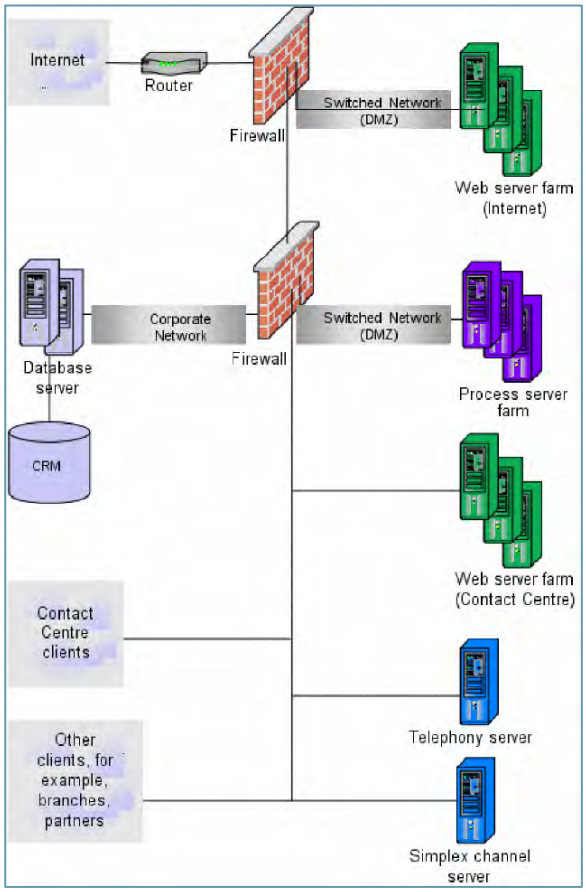
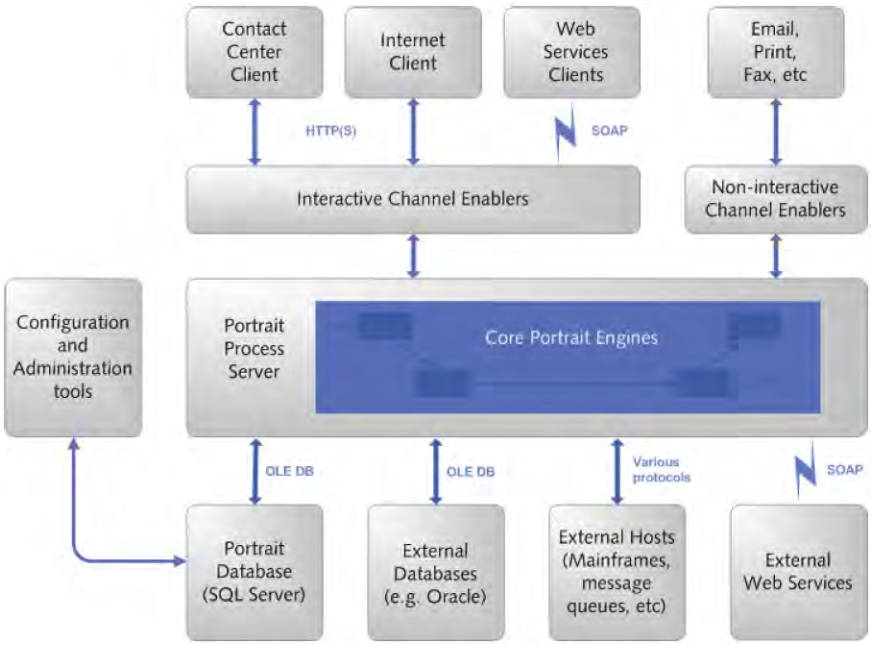


Figure 2 - WAN (Internet) Structure

1.2 External Systems

This diagram below shows inter-component communication that can exist in a typical Portrait Foundation solution.

Figure 3 - Integration



Monitoring the connectivity to external systems can provide useful information when a problem is reported. Being aware of existing issues outside of Portrait Foundation can save a lot of time.

- Connectivity to data sources, data access transactions, queries, etc.
    - Web services
    - Message Queue Transactions
    - Databases
    - Back-end systems
- External systems where processes are performed on behalf of Portrait Foundation
    - Telephony systems
    - Print Server

Finally, if all external systems check out OK, the Implementation specific configuration should be checked, including such things as Data mappings, Custom (or modified) models, Custom DAT's, Custom nodes, etc.

1.3 Once Everything Is Working

This document does not concern itself with installation issues, and makes the following assumptions:

- All of the guidelines in the Operations Guide will have been followed during installation and setup.
- A stable functioning Portrait Foundation installation has been achieved to begin with.
- Routine Housekeeping has been set-up to monitor the health of the system.

- Event Logs
- Portrait Logs
- Windows Performance Monitor (PerfMon)
- Performance by Request

1.4 Useful Tools to Know

The table lists tools and techniques available for problem determination and indicates their major uses in problem determination and the level of specialist knowledge required for their use.

Tools and techniques available for problem determination

PORTRAIT TOOLS	Major uses in problem determination
Portrait Management Console	This is the primary tool for managing all of the Foundation runtime settings and various characteristics of the system
Portrait Service Check	Check whether appropriate services are installed and running
Portrait Version Checker	Use this tool from the program menu to verify or confirm the versions of all of the Portrait binaries in the Portrait program install and the implementation directories
Portrait Logging	Obtain information about errors and normal function of Portrait components
Portrait Log Viewer	Obtain information about errors and normal function of Portrait components. Also capturing a log viewer log, capturing logs to a file.
Performance by Request Logging	Performance by Request logging provides response time information for all requests into the web server and process server, providing detail of the operation, model, web service that is being executed and any Data Access Nodes executed in the process server for that request.
Performance Auditing (Model and Node Viewer)	Performance Auditing provides data on the performance of models & nodes within the process server that can be used for both monitoring general performance and for diagnostic purposes when investigating critical issues.
Portrait Model Diagnosis Tool	Execute and record models and examine the path taken and the data passed through models

MICROSOFT TOOLS	Major uses in problem determination
Windows Event Log	Check for significant System and Application events
Windows Task Manager	Check basic health of a process (running, responding, CPU usage and so on) A dump file for a process can be created with Task Manager. It will be written to the C:\Users*UserName*\AppData\Local\Temp directory
Windows Performance Monitor	Check resource usage (CPU, Memory, Disk I/O and so on) Note: When using a 64-bit operating system, use the 32-bit version of the Performance Monitor, by running the command: "`mmc.exe /32 PerfMon.msc`"

OTHER RESOURCES	Major uses in problem determination
Portrait Knowledge Base	http://www.portraitsupport.com/Pages/knowledge/Default.aspx Compare the nature and circumstances of the problem with the descriptions of known issues
Portrait Support	Access to specialist expertise when other problem determination tools have been exhausted
MSDN	Microsoft Developer Network contains a wealth of information if you have access to it

OTHER RESOURCES	Major uses in problem determination
Microsoft Knowledge Base	Search the Microsoft website for white papers and KB articles relating to error codes, error messages or any other appropriate key words.

1.5 Tools To Be Most Familiar With

It is a good idea to be comfortable with the following tools. It is not the purpose of this document to

1.5.1 Event Logs / Event Viewer

The Event Viewer is a component of the Windows operating system, which is used to view event logs.

Windows Event Logs record events on the system into different logs including

- **Application** Log. Events in this log are classified as:
    - Error - a significant problem, such as loss of data
    - Warning - an event that is not necessarily significant, but might indicate a possible future problem
    - Information - describes the successful operation of a program, driver, or service
- **Security** Log contains security-related events, or audit events. These events are either successful or failed.
- **System** log system level events write to this log. The events are classified as:
    - Error - a significant problem, such as loss of data
    - Warning - an event that is not necessarily significant, but might indicate a possible future problem
    - Information - describes the successful operation of a program, driver, or service

1.5.2 Portrait Logs / Portrait Log Viewer

⇒ Note : The Portrait Log Viewer does not currently support *.evtx files.

- It is a tool installed by the Portrait Foundation installation which, besides displaying real-time logging, can open a saved Portrait (*.plf) log file and event log (*.evt or *.xml) files.
- The log viewer displays detailed information about the nodes being executed within models, the data being passed between nodes, and when data is returned back to web tier etc. It should not be used in a live environment under normal circumstances due to the performance hit associated with the volume of data that is generated. However, despite this large volume of data, since the logging captures information as it happens, it can be useful to be able to see the "live" sequence of events.
- It is quick and convenient to view live logging directly in the Portrait Log Viewer. Sometimes it is necessary to look at data over a long period of time or to study a complex sequence of events. In that case it is practical to have logging output to a file so you can open them in the Log Viewer at a later time.

1.5.3 Windows Performance Monitor (PerfMon)

- PerfMon is an administrative tool built into all Windows versions. It monitors performance information from the operating system and Portrait. The Data Sets (or Counter Logs) are output to a binary file that can be reopened in PerfMon.
- The data will show the utilization of resources as well as the count of specific types of faults. It takes a snapshot of these figures at the set intervals.
- This information can be used for general health checking of the system to check for performance of the operating system components, server hardware, and Portrait, and can also be used for diagnostic purposes when investigating system problems.

1.5.4 Performance by Request

- This is setup via the Portrait Foundation Management Console, it should always be enabled. The output will be text based files that can be reviewed in an excel spread sheet and details the operation or Custom Control that was requested, the activity token, and the time to deliver it.
- Check the logs to identify the long running requests, such as certain operations taking longer than others, requests for a particular activity token always taking a long time, or all requests taking longer than expected at certain periods during the day.
- Similar information can be obtained from the Windows IIS log. IIS logging is enabled by default on installation of Windows, but requires a further configuration change to add the time-taken field to the logs that indicates how long the page took to be served. Please refer to Microsoft's IIS documentation if you need to do this.

1.5.5 Performance Auditing

- This provides data on the performance of models & nodes within the process server. It will record statistics around the time taken to execute individual models and nodes within those models. The data for each model and node includes the maximum, minimum and average durations and is configurable so that data can be averaged over a small window if necessary.
- This data is viewed using the Portrait Foundation Model and Node Profiler.

2 Initial Problem Investigation

Regardless of the cause of the problem, it is necessary to gather as much information as possible about it. It helps to focus on the answers to who, what, where, when and how? This investigation will aid in determining which approach has to be taken to resolve the problem.

2.1 Who?

- Are all users affected?
- Are a majority of users affected?
- Can the problem be limited to certain users?
- Does the problem only affect a limited number of users?
- Is the problem User or Machine specific?
- Is the problem limited to certain customer records?

2.2 What?

- Has this issue occurred previously? If so, what was done then to resolve it?
- Is there a specific function involved when the problem occurs? (i.e. accessing a database; accessing data on a back-end system; accessing back end transactions; etc.)
- Is there a new feature being used for the first time?
- Did this feature work correctly, before it failed? If so, what changed in between?
- Can the problem be isolated down to the smallest affected area, component, model or node?
- Is it possible to list the areas or functions of the system that work as expected, or are confirmed to be working correctly?
- Is it possible to list which areas or functions of the system that are suffering?
- Are a majority of product functions affected or only some?
- What features are common to the problem case(s) that are not present in the successful case(s)?

2.3 Where?

- Are a majority of machines affected or only some? Could the problem be isolated to a segment of the network?
- Are there any tiers or components that you can exclude from the problem? Can you identify where it works and where it doesn't work?
- Does the same problem occur in other Portrait Foundation environments, or does it only occur on one machine?

2.4 When?

- When, exactly, did the problem start?

- Does the problem happen at a specific time of the day, or day of the week? If so, does it coincide with other scheduled processes? (i.e. batch file runs, antivirus scans, data import/export, peak busy times, etc.)
- Did the initial occurrence of the problem coincide with any changes implemented in other systems, changes to architecture, upgrades of any type, maintenance, etc.?
- Is the problem intermittent or is it consistent and easily reproducible?
- When does the problem manifest itself (at logon or logoff, during high or low load conditions, during start-up, during shut-down or during normal usage, on certain days, at a certain time of day, etc.)?
- Is the feature being used for the first time? If not, what has changed since it last worked?

2.5 How?

- Does the problem have a pattern to it?
- Is the problem reproducible? If so, how is it reproduced?
- What is the minimum set of user actions required to reproduce the problem?

3 Defining the Problem

When trying to solve a problem it is important to be able to fully describe it. There are a number of ways to define problems, but those which occur with enterprise solutions such as Portrait Foundation will usually fall into one of the following categories:

3.1 Crash

PROBLEM DEFINITION:

The application, service or User Interface (UI) will terminate unexpectedly.

If the crash happens in UI there may an application or system error message displayed. If there is a "More Information" button it should be pressed to gather more detailed description of the problem.

With a web service there might be a timeout message in the log file of the application communicating with the web service. In any case there should be error messages in log files and event logs. There should also be a UserDump memory dump file with the date and time of the crash.

WHAT TO DO ABOUT IT:

For initial investigation the Portrait Foundation log files, or event logs, need to be opened in the Portrait Log Viewer. In the Log Viewer filter on or search for any displayed error numbers, error text and/or key words, such as **critical**, **severe**, **fatal**, **exception**, **timeout**, **error**, etc. These searches may inform you about problems inside or outside of Portrait Foundation.

Deeper investigation may require an analysis of memory dump (.dmp) files which should be sent to Portrait Support along with all other collected logs.

3.2 Hang

PROBLEM DEFINITION:

The application or service stops responding.

In a UI, it will stop functioning, but remain on the screen. It will not respond to the mouse or keyboard input. The Windows Task Manager may report the application as "Not Responding". With a web service there will be timeouts reported by the calling application.

WHAT TO DO ABOUT IT:

For initial investigation the Portrait Foundation log files, or event logs, need to be opened in the Portrait Log Viewer. In the Log Viewer filter on or search for any displayed error numbers, error text and/or key words, such as **critical**, **severe**, **fatal**, **exception**, **timeout**, **error**, etc. These searches may inform you about problems inside or outside of Portrait Foundation.

Deeper investigation may require an analysis of memory dump (.dmp) files which should be sent to Portrait Support along with all other collected logs.

3.3 Unexpected or Wrong Data

PROBLEM DEFINITION:

Everything appears to be functioning correctly, but the displayed or produced data is missing or incorrect.

If, incorrect data is displayed in a UI, it is significant to identify the object displaying the incorrect data (custom interaction, combo box, grid control, edit control, etc.) so it can be traced back to the source.

WHAT TO DO ABOUT IT:

Since this is typically caused by errors in customization (custom controls, model definition, model mapping, nodes, data access transaction, generated interaction, custom interaction, database stored procedure, XSLT transform or coding error) it must be dealt with by the developers of the implementation.

Scanning log files for key words, such as **critical**, **severe**, **fatal**, **exception**, **timeout**, **error**, etc. may help indicate where the problem lies.

3.4 Unexpected or Wrong Events

PROBLEM DEFINITION:

Everything appears to be working correctly, but there is an unexpected UI event (i.e. an unexpected message box appears, the application navigates incorrectly, a hyperlink navigates to the wrong URL, an error message or other content).

WHAT TO DO ABOUT IT:

Since this is typically caused by errors in customization (custom controls, model definition, model mapping, nodes, data access transaction, generated interaction, custom interaction, database stored procedure, XSLT transform or coding error) it must be dealt with by the developers of the implementation.

Scanning log files for key words, such as **critical**, **severe**, **fatal**, **exception**, **timeout**, **error**, etc. may help indicate where the problem lies.

3.5 Slow Response

PROBLEM DEFINITION:

Everything is working as expected, but it takes a long time to complete, longer than expected to display a page, or the data on a page.

WHAT TO DO ABOUT IT:

Reviewing the log files, PerfMon counters, Performance by Request and Performance Auditing logs could help in identifying which components are taking a long time to complete. This could have a number of possible causes, all of which are complicated issues which must be dealt with by the developers of the implementation. The cause could be by any of the following:

Resources such as Memory, CPU and Disk I/O could be depleted. There could be a coding error in an XSLT transform, node, data access transaction or database stored procedure causing excessive volumes of data to be processed. A model could have incorrect data mapping within it. There could be a logical error in the definition of a model or a custom interaction. A database query may be missing an index or running inefficiently for another reason.

Another thing to check is which logging is turned on in the Management console. Normally the only logging that should be turned on in a production system is logging to the event log. MDT logging should never be used in a production system. You can also check to see that the logging that is enabled does not use a filter that is too open.

3.6 High Memory Use

PROBLEM DEFINITION:

Everything is working fine up to a point at which it suddenly gets slow.

Alternately it may be discovered that a process is using much more memory that it is expected to use. This can cause memory paging which will dramatically slow down a system.

WHAT TO DO ABOUT IT:

Since many Portrait Foundation components are designed to use large amounts of memory in the form of caches in order to improve performance by cutting usage of other resources such as Disk I/O it is advisable to review the cache settings first.

Portrait Logs and Event Logs can be searched, scanned or filtered on appropriate key words, such as **critical**, **severe**, **fatal**, **exception**, **timeout**, **error**, etc. to see if any problems are being reported. PerfMon logs should to be analyzed to determine which system resources are being consumed and Portrait Performance logs to determine if certain Portrait objects are consuming excessive resources.

It is possible that memory resources are being over consumed or consumed and never released. The cause could be inappropriate cache settings but it could also be a memory leak in a custom node or Data Access Transaction. These types of issues would have to be dealt with by the developers of the implementation because it may require a review of all customized project code (especially resource handling code).

3.7 Resource Exhaustion/Depletion

PROBLEM DEFINITION:

Everything is working fine up to a point at which it suddenly gets slow or starts experiencing faults or failures. It may be noticed that the system is running out of resources, or a particular resource over time such as memory, handles, semaphores, disk space, etc.

This is similar to memory issues above, but there are other finite resources that can be consumed, or leaked, over time. Leaks/depletion of Handles, semaphores or system Page Table Entries (PTE's) can show up as failures or progressive instability or poor response.

WHAT TO DO ABOUT IT:

To troubleshoot this type of problem it is necessary to look at PerfMon logs to find the resources which are being depleted. The Portrait logs may give some clues about which components may be depleting the resources. As with high memory use, the cache settings might need to be revised.

3.8 Instability

PROBLEM DEFINITION:

The system is not reliable. Sometimes it works and sometimes it appears to be unstable with random errors and inconsistent behavior. Instability is an umbrella term describing a system that appears to behave inconsistently and/or unreliably under certain circumstances.

WHAT TO DO ABOUT IT:

In this case the problem itself is unclear and the problems encountered are inconsistent. This will require a systematic approach, starting with going through Portrait logs scanning or filtering on appropriate key words, such as **critical**, **severe**, **fatal**, **exception**, **timeout**, **error**, etc. All performance logging should be reviewed to try to identify any issues or irregularities. Possible things to be looking for include exceptions, model failures, event log entries, and any other indicators which indicate failures.

4 Progressing the Problem

4.1 Installation Issues

A faulty install of the product will cause issues in the implementation. Install logs will be written to a **PortraitInstallLog** folder. It is important to make sure the installation of Portrait Foundation has completed without errors, warnings, failures or alerts before beginning to investigate any implementation specific issues. If there have been any unresolved issues in the install log they must be addressed, or Portrait Foundation needs to be reinstalled prior to investigating any implementation related issues.

As indicated earlier, this document makes the assumption that the server was originally configured correctly, a successful installation has taken place and that a problem occurred after the system was successfully installed.

One cause of issues in an otherwise clean installation of Portrait Foundation is Windows Security settings. Default security settings in the most recent versions of Windows are very tight and have to be modified in order for Foundation to function.

It is likely that there is an environment issue or software installation issue if a problem affects:

- The majority of product functions, or some functions but not all.
- If affects some functions so badly that other functions cannot be reached.
- One function fails causing cascading failures in other functions.
- It affects one system but not another where both are theoretically identical.

4.2 Environmental Issues

Problem solving can be an iterative process, or trial-and-error, but until the discovery of something specific that is malfunctioning or functioning badly it is just guesswork. Following these guidelines can make it educated guesswork and thereby expedite the eventual solution.

To find out where the problem really is, it is important to do some preliminary investigation work. To advance the problem it is always a good idea to confirm that all external systems are functioning correctly, including, but not limited to:

- Network Infrastructure (everything including network card configuration, protocol, sockets, routers, domain controller, etc.)
- Any Back-end database (stored procedures, queries, functions, triggers, etc.)
- Transactions on Back-End systems (including network subsystems, message protocols, message syntax, etc.)
- Web services (transaction selections, message construction, data parsing, etc.) This can often be tested (in a non-production environment) with a third party tool such as *soapUI*.

Knowing and understanding the tools at your disposal is important, but it is more important to approach the overall problem in a logical way in order to find the cause of the problem as quickly as possible.

It is best to start off at a high level, looking at the system and the problem as a whole, and then drill down to the detail bit by bit. Use the following sequence as a guide:

4.2.1 Determine what has changed, "ANYWHERE"

When a system works, and then suddenly it stops working correctly, it is almost certain that **"something has changed"**. It is often easier to ignore changes that have been implemented in other systems that Portrait Foundation communicates with, or you may not have been informed about it. Trying to rule out Portrait Foundation as a first can cause days, if not weeks of time wasted looking in the wrong direction. The Portrait Foundation Log files may point to a failing back-end system, but it is a lot faster to verify those systems as a first step. Similarly, when problems appear after implementing changes in Portrait Foundation, those changes need to be revisited if a problem suddenly appears. It is always unwise to assume that the implemented changes are unrelated to the problem without a thorough investigation. The most important question to ask is: **Has anything else in a related area or system changed recently? If so, what was it? Start your investigation there.**

4.2.2 Confirming Where the Problem Is

Whoever is responsible for the health of Portrait Foundation should endeavor to stay informed of changes, upgrades, etc. or at least the overall status of all systems that Portrait Foundation communicates with, for the following reasons:

- If a back end system times out, Portrait Foundation will not get the response.
- If back end systems are slow, it will cause Portrait Foundation to be slow.
- If a SQL query takes a long time, Portrait Foundation will have to wait for the response.
- If an agent cannot log in to their telephone directly then they will not be able to log in to telephony with Portrait Foundation either.

If all external systems appear to be functioning correctly, then the problem could very likely be in Portrait Foundation.

4.2.3 Confirming That There Is a Problem with Portrait Foundation

One means of determining if a problem exists is to establish response times on the web server. The web server is where Portrait Foundation is exposed to the world. All of the processing in Portrait Foundation has been done at this point, data has been obtained from back end systems and databases and it has all been parsed, processed and prepared for delivery to the end users. Any performance problem within the Portrait Foundation implementation or dependent systems should show up as slow delivery of web pages (or web services) or long times for operations. IIS or Performance by Request logs are both useful in investigating this.

Another approach is to analyze Performance Monitor data to see if there are any processes behaving irregularly. If there is evidence of recent memory dumps with the same dates and times that problems occur, or if the performance monitors show unexpected behavior then further investigation will be required.

5 Overview of the Logs

This section contains advice for tracing a problem to its source from within Portrait Foundation using various tools provided with the Portrait Foundation software or the Operating System.

If there is a problem there are actions which could expose the cause without involving Portrait Support.

5.1 Types of Logging

Type	Notes
Performance Monitor (PerfMon)	Data about system state and performance that are exposed by applications and recorded by the system tool "PerfMon" at predefined frequencies.
PortraitPerformance.csv	Otherwise known as "Token Tracing" or "Cecil Logs".
Portrait Logging	• Debug – shows up when debug is enabled in the event viewer • File – logs all entries to a file (*.plf files) • Database – logs all entries to a database • EventLog – logs entries to the Windows Event Log • Viewer – logs entries to the Portrait Log Viewer • MDT – Logs entries to database when MDT is running
Portrait log files (*.plf) or event logs (*.evt, *.xml)	Open in Portrait Log Viewer, even if it's an event log – we can show more information and better searches. We cannot currently load *.evtx files – it must be saved as *.evt or *.xml.
Model and Node counters	Data on time spent in each node of each model, averaged over time. Recorded to the operational database or transient database if there is one. Has minimal performance impact in Live, and incredibly useful.
SQL Profiler Trace	Run SQL Profiler on the database server, to investigate slow-running queries. Not generally used in a Live system.
Dump files	There are various methods available to create a dump of a running process. These depend upon the OS being used. Don't do this during Live use, but you can do this immediately before stopping and re-starting the service. Also, Portrait applications will generate a dump (with a configurable amount of detail) on hardware exceptions (e.g. access violations). You can also use the .plx (Portrait log filter) file to trigger a dump on a given log message.

There is also an extended matrix in Appendix B that may help to explain the differences between the various Portrait logging types.

5.2 Gotchas in the Logging

- The following logs record in local time:
    - PerfMon
    - PortraitPerformance.csv

The following record in UTC:

- Portrait logs (*.plf)
- Event logs (*.evt, *.xml) if viewed with Portrait Log Viewer, but will show in local time if viewed with Event Viewer!

- In general, the *Date Modified* on the files will be the local time of the last event recorded.
- UTC is essentially the same as GMT, so for a UK client in summer time the UTC time will be 1 hour before the local time. For an East-coast US client UTC will be 5 hours ahead of local time, or 4 hours in summer.
- PortraitPerformance.csv logs are not precisely in chronological order (they are in the order of the request ending, not beginning), and there can be overlap between successive files. Use the following syntax in a DOS box to concatenate two such files in order to eliminate overlap problems:

```
copy <First_PortraitPerformance>.csv + <Second_PortraitPerformance>.csv  Combined.csv
```

- The *Activity Token* in the PortraitPerformance.csv file is actually the concatenation of the *Root Token* plus *Activity Token* as recorded in the Portrait Log file.
- In a production environment only the Event logging should be turned on, unless otherwise recommended by Portrait Software. MDT Logging should NEVER be turned on in a production environment.

5.3 PerfMon Counters

A review of the PerfMon counters is a good place to start.

In all the groups listed below look at all the counters for the **HostU** processes, looking for anomalies or surprising values. Those in **bold** are the main ones to look at when doing a quick review of a system, the others for a more detailed look or if you have reason to suspect them.

Generally, HostU#1 is the *Primary ServiceHost* process, so for process-based counters that's the one we want to look at. However, the Portrait-specific counters list the *ServiceHosts* by their full names, so select the Primary instance for these.

Group	Counter	Things to look for
.NET CLR Loading	**Current Assemblies**	**This includes dynamically-generated assemblies - should not increase over time or it may indicate an "assembly leak", eg bad use of XmlSerializer - if so, go to Diagnosis section**
.NET CLR Memory	**All, especially: Gen 0/1/2 heap size Large Object Heap size #Bytes in all Heaps**	**Should level out, unless you have a leak - if so, go to Diagnosis section**
Network interface	All	Is the network card a limiting factor? Look at bytes sent and received and compare with the Current Bandwidth counter.
Physical Disk	All	Are the disks too busy?
Portrait *	All	These counters list the *Portrait ServiceHost* instances by name, rather than as "HostU#1". You are looking for the primary service host instance. Look for any anomalies, in particular…
Portrait *	Any cache misses counters	If any cache is showing a lot of misses relative to the number of hits, it's probably sized too small.
Portrait Cache Counters	**Model definition cache object count**	**This is the cache with much the largest objects, and hence that consumes the most memory - check this is not unlimited. If the steady-state memory usage is too large, consider reducing this.**
Portrait Fixed Allocation Counters	**FA allocated bytes**	**Should level off (though fluctuating with load) or you have a leak - if so, go to Diagnosis section**

Group	Counter	Things to look for
Portrait Fixed Allocation Counters	FA allocation count	Should level off (though fluctuating with load) or you have a leak – if so, go to Diagnosis section
Portrait Fixed Allocation Counters	FA real allocated bytes	Goes up in steps, but should level off or you have a leak – if so, go to Diagnosis section
Portrait Fixed Allocation Counters	FA real allocation count	Should level off or you have a leak – if so, go to Diagnosis section
Portrait Global Cache	Global Session Management Cache Object Count	Should stop at the preconfigured level. Must not be unlimited or you'll definitely run out of memory.
Portrait Global Cache	Global Session Management Compressed/Uncompressed IO bytes/sec	A measure of the data traffic to and from the transient database. Is this overloading the network card? If so, consider switching on compression if it's not on.
Portrait Misc Object Counters	**All**	**All these should level off, though fluctuating with load. Check they return to the same level each night, or they indicate a leak – go to Diagnosis section**
Portrait Process Engine	**Actual Dispatcher Threads**	**Should equal Configured Dispatcher Threads. If this rises you are "losing" dispatcher threads which means some requests are taking a very long time to complete. If this doesn't fall back again some requests are not completing at all. This is a very bad thing. Note the times and go to Diagnosis section**
Portrait Process Engine	**Dispatcher Threads Busy**	**If this doesn't return to 0 periodically, you may have one or more requests that are either taking a very long time or not completing at all. Note the times and go to Diagnosis section**
Portrait Process Engine	**Dispatcher Threads Waiting**	**If this rises above zero it's a good sign that the system is overloaded – work is being held up while other things complete. Note the times and go to Diagnosis section**
Portrait Process Engine	**Failed models**	**We don't want any of these. Note the times when this counter rises and go to Diagnosis section**
Portrait Process Engine	**Hardware exceptions**	**Should have none of these. If we do, note the times and go to Diagnosis section**
Portrait Process Engine	**Models Total, Nodes Total**	**As for the Portrait Misc Object Counters, these should level off, though fluctuating with load. Should return to the same level each night. If not you may have a leak – go to Diagnosis section**
Portrait Session Manager	**Data Contexts**	**Should stop at the configured level – must not be unlimited or you'll run out of memory.**
Portrait Session Manager	Local Session Management Cache Object Count	Should stop at the configured level – must not be unlimited or you'll run out of memory.
Portrait UIE Thread Pool Counters	**UIE pool average execution time/UIE pool average group execution time**	**Look for peaks in this that indicate the system is running slowly – note times and go to Diagnosis section**
Portrait UIE Thread Pool Counters	**UIE pool average wait time**	**Should be zero most of the time – peaks indicate the system is slow – note times and go to Diagnosis section**
Portrait UIE Thread Pool Counters	**UIE pool requests waiting**	**Should be zero most of the time – peaks indicate the system is slow – note times and go to Diagnosis section**
Portrait UIE Thread Pool Counters	**UIE pool timed out groups**	**Should be zero. If this rises it indicates either a very slow running system or UIE requests that never terminate. Note the times and go to Diagnosis section**
Portrait Web Service	**xxx requests/sec**	**This is the best measure of the level of load being applied to the system.**

Group	Counter	Things to look for
Portrait Web Service	Average xxx response time	These give the average response times of different types of request. Look for peaks indicating times of slow running. Note times and go to Diagnosis section
Process	**% Processor Time**	**As in Processor Information above**
Process	All	For all the *HostU* processes, especially HostU#1, look for anomalies. In particular check the following…
Process	**Handle Count**	**Portrait uses a lot of handles (for locks) – of the order of 100,000 is fine.**
Process	**Private Bytes**	**Broadly speaking, the total amount of allocs done by the process. Should level out after a day or two. Will fluctuate with load but should return to the same level each night or you may have a leak – if so, go to Diagnosis section**
Process	**Thread Count**	**Expect between 50 and 150.**
Process	**Virtual Bytes**	**This is the number that kills the process when it hits 2GB (or 3GB if running on a 32-bit machine with the /3GB switch set, or a little over 3GB on a 64-bit machine). Should level out after a day or two or you have a leak – if so, go to Diagnosis section**
Process	**Virtual Bytes Peak**	**Same as Virtual Bytes, but useful as it shows brief peaks that may have happened between samples of Virtual Bytes.**
Processor Information	% Processor Time	How busy is the CPU? If it's consistently high (e.g. over 70%), the system is running close to capacity.
Processor Information	All	General health of the PC

5.4 Performance by Request Logs

Performance by request logs show the length of time a process takes to run. It is useful to view this data in a spreadsheet and sort the data by duration. This will show you which processes are taking the most amount of time to finish.

5.5 Portrait Logs

Portrait Logs can be searched, scanned or filtered on appropriate key words, such as **critical**, **severe**, **fatal**, **exception**, **timeout**, **error**, etc.

5.6 Performance Auditing

A quick first step is to sort the node data by average node execution time and look at the nodes that are taking the longest time to execute. For example, a data access node that is accessing an external system that is taking a long time should show up near or at the top of the list. Also check the instance counter for nodes within a model. A node or number of nodes within a particular model that have executed a large number of times may suggest that a model is looping, perhaps incorrectly.

Do the same exercise for model execution times, and see which models are taking the longest to execute. It may be that individual nodes within models are all running quickly, but a complex path through a model, or a loop, means that the entire model takes longer to execute.

If the performance problem is one that only happens at certain periods of the day, plot the data over time and compare model & node execution times during the slow period with those during a period where the system is performing correctly.

It can be a useful exercise to group model and/or nodes by their type and look at the performance of different groups. That is, nodes that are simple basic Portrait

nodes (or, and, conditional branch) can be grouped together, and should all perform quickly. Data access nodes that access external systems can be grouped together. Nodes that access the Portrait database, such as "*SaveParty*" or "*RetrieveEngagementDetails*" can be grouped as *Database Transactions*.

5.7 Memory Dumps

The Portrait Foundation installation configures MiniDump for you. The dump files will be written on the Portrait Foundation server to the *C:\Program Files\PST\Portrait* directory (or the *C:\Program Files(x86)\PST\Portrait* directory on a 64bit server). The name of this file will be in the following format: *processName_Date_Time.dmp*. You should check to see if there is a .dmp file with the same date and time that the problem occurred.

6 Examples: Log File Diagnosis

Having spotted problems in the PerfMon counters, this section describes how you can use other types of logging for investigating further. In general you should have noted the times of the problems above.

6.1 Diagnosis – Things Are Slow or Overloaded

Having noted the time when the slowdown occurred, load the PortraitPerformance.csv for that time into Excel. Change the formatting of the time column to Custom - dd/mm/yyyy hh:mm:ss.000. Select Cell B2 and choose View – Freeze Panes to keep the top row and left-hand column in view.

It is assumed in the following that you have request logging at the Web level, the Dispatcher level and at the Node level at least for all DATs and DACs that go "off-box", but ideally for them all.

Firstly, bear in mind that the Portrait Performance logs only contain rows for requests that end. If you had a call that ran a model that went into a tight loop, never calling DATs and never returning, then you won't see any rows in the log for this, at any Level. If you suspect this is the case, go to the Diagnosis 6 section instead.

By default, sort on Timestamp, as the rows are save in the order the requests end, not arrive.

Now try sorting on Duration to see the slowest requests. Are they at the time you expected?

For each slow request in turn, note the Request Token and filter the Request Token column so you only show the rows related to that request. If it helps, re-order by Timestamp so you can see the correct sequence. Expect to see either:

a. One Node-level request that takes up most of the time. For example, the Web-level Duration might be 51234 (51 secs), the Dispatcher-level might be 51012 (51 secs) and at the the Node-level you might have a single DAT request that takes 49442 (49 secs). This indicates that the cause of the problem is that Data Access Transaction. If it goes off-box (e.g. calls one of the customer's own web services) then they need to review performance of that, or network connectivity. If it is one of our DATs (e.g. SearchForTask2) then we need to look in more detail at the SQL being performed. It's either a very inefficient query, or it's returning a large amount of data. If the latter, you'd expect to see a spike in the PerfMon Data Object count at that time.

b. Alternatively, you might see 500 Node-level requests, all associated with the same Request Token. This tends to indicate a badly-designed application: it shouldn't generally be necessary to make this never of data access requests.

In both cases, now go to the section General information to gather about problems.

6.2 Diagnosis – Failed Models

Open the Portrait Log files (*.plf) or event log files (*.evt) for the relevant time with the Portrait Log Viewer.

Search for the text RESULT_ERROR: every failed model will log an error containing this text.

Once you've found the error, note the request token and activity token. Try applying a filter to the Log Viewer so you only see those lines with the same Request Token, i.e. just those lines associated with the problematic request.

Look for the first error, then go to the code and try and deduce what might have caused the problem. For example, was it a Custom Interaction returning with an invalid outcome, or an invalid data item?

Gather as much information about the error from the log file, and then go to the section **General information** to gather more context about the problem.

6.3 Diagnosis – Exceptions

Open the Portrait Log files (*.plf) or event log files (*.evt) for the relevant time with the Portrait Log Viewer.

Search for words like "Exception", "Access violation", "ACCESS_VIOLATION".

Once you've found the error or errors, follow the steps under Failed Models above to get more information about what request was in progress.

6.4 Diagnosis – Requests That Don't End

These can be more problematic, as they may not leave any record in the PortraitPerformance.csv file.

If it's a UIE timing out that you're looking into then you'll get some help from the Portrait log file or event log file, and should be able to find the Request Token of the timed out request. Search for the text "*A timeout occurred before all UIEs could be processed*". This line will give you the Request Token for the problematic request, so you can then see if any other lines were logged from the same request. This may give you some clues.

Also, go to the PortraitPerformance.csv file for the same time and search for that Request Token – there *may* be records saved for this request that will tell you more. Failing that, try searching for the Activity Token instead – at least that will tell you what that user did *just before* the request that didn't end.

On the other hand, if it's a Dispatcher thread timing out (and you don't have a UIE timeout as well), it may be hard to find out the Request Token of the problematic request. Scan the event log for suspicious errors at that time. You could also try looking in the PortraitPerformance.csv file to see if you can find any Node-level records for which no Dispatcher-level or Web-level rows exist. In other words, if the request never ended you won't see rows for the Web or Dispatcher levels. However, if the request called any DATs successfully before whatever happened that never terminated, you should still see rows for these requests. That may give you a call-stack in the *Call Detail* column, which may tell you what was going on.

6.5 General Information to Gather About Problems

Once you've got the Request Token for a problem, the best information generally comes from the PortraitPerformance.csv file. The Call Name will give you information about the call at each level, and the Call Detail at the Node level will give you a call-stack of the models that resulted in this call. This is tremendously useful in saying how we came to call this Node.

Gather up all this information to give the best picture of what was being executed.

Also, look in the Portrait Log file or event log file to see if any further information is available there for that Request Token or Activity Token. Remember, Request

Token identifies that request – generally a single click on a menu or button, while Activity Token identifies a user's session, i.e. *everything that person did*. Also remember the "gotcha" about Activity Tokens not matching between PortraitPerformance.csv files and Portrait Log files (see section 5.2 – Gotchas in the logging).

Having got all the information about an incident (by that I mean a failed model, slow request, etc.), use the Portrait Performance logs to answer the following questions as well:

- Is this type of request always slow (and/or does it always call a lot of DATs)? On all servers?
- Is it slow at certain times of day only?
- Are there other types of error going on at the same time, e.g. network or database errors?
- Are other things slow at the same time? i.e. is it a symptom of other problems, rather than necessarily the problem itself?

7 Appendix A – Contacting Portrait Support

When contacting Portrait Support always provide the following information:

Site details:
- <Customer name> / <Contact name>
- <Contact company – if not same as customer>
- <System type: live, development, test, training >

Environment details: <whatever applies>
- Operating System version details:
  - Windows Server <i.e. 64Bit 2008 R2 SP1 Enterprise>
  - Windows client <i.e. 64Bit Win 7 Enterprise SP1>
  - UNIX version <output from "uname –a">
- Database <i.e. SQL Server 2008 R2 Standard>
- Windows SharePoint Services <i.e. 3.0 SP2 x64>
- IIS version <i.e. Internet Information Services 7>
- Silverlight version <i.e. 5.1.10411.0 (64-bit)>

Portrait product version details: <whatever applies>
- Foundation <i.e. 43.0.100.78>
  - <Single-Box or Split-Box>
- Interaction Optimizer <i.e. 5.3 GA>
- Dialogue <i.e. 5.1.3.114>
- Miner <i.e. 6.1>
- EDGE Developer <i.e. 7.7.5 {732070705001}>
- EDGE Client <i.e. 7.7.2 {732070702001}>
- EDGE Server <i.e. 7.7.2 {225070702001}>

Problem: <Describe the problem as clear as possible>
Call type: <Support issue, Change request, Question, Suggestion, Task>

Details:
- <Customer Request>
- <Detailed description of problem>
- <Steps to reproduce>
- <Additional information/details>
- <Impact on customer>

Actions taken so far:
- <Work-around>
- <attached files (PerfMon logs, Event logs, IIS logs, Performance By Request logs, memory dump /user dump / core dump files), Foundation logs, Dialogue logs, EDGE logs.

8 Appendix B – Portrait Log Types

The following matrix shows the various different Portrait Log types and answers a number of common questions about each of them.

	Portrait Foundation Logging				Performance by Request (aka Portrait Performance)	Performance Auditing (aka Model & Node)	Performance counters (aka PerfMon)	Notes
	Logging to file	**Logging to Event Log**	**Logging to Portrait viewer**	**MDT logging**				
What is it?	Sends Portrait output logs to a file location on the server.	Sends Portrait output logs to the Windows Event Log.	Streams live Portrait logging information to the Portrait Log Viewer.	Sets up Portrait Foundation session recording and isolated model testing used in the MDT.	Records individual timings for configured Data Access Transactions.	Records statistical info about Models and Nodes over pre-set intervals.	Extends Microsoft's Performance Monitoring to include snapshot values about Foundation data.	Portrait Foundation Logging 'Debug' and 'Database' logging have not been included here.
What's the impact to the clients?	None (see note)	None (see note)	None (see note)	Not applicable	None (note)	None (see note)	None (see note)	Clients are only impacted to the extent that the server performance is affected.
Is there any downtime required to set or unset the log type?	Portrait Services restart **not** required for setting or unsetting.	Portrait Services restart **not** required for setting or unsetting.	Portrait Services restart **not** required for setting or unsetting.	Portrait Services restart **not** required for setting or unsetting.	Portrait services restart **not** required for setting or unsetting.	Portrait services restart **not** required for setting or unsetting.	No, but should have been selected during the initial Portrait server installation.	These answers assume that the individual servers have all been listed within the MMC.
Is this something that should just run, or activate as needed?	Needs to be activated in the MMC, and can be used as *an alternative* to Event Log logging, but normally only turned on upon request.	By default it is activated in the MMC. It is useful to see Portrait logs alongside other system event logs, but it can cause Event Log rollover if the logging is too verbose.	Needs to be activated in the MMC, and can be used as an alternative to Event Log logging, but normally only turned on upon request.	MDT logging should not need to be switched on manually. The Model Diagnosis Tool (MDT) automatically sets the logging availability. Never run the MDT against a Live system.	Needs to be activated and configured via MMC, and recommended to always be turned on in Live - logging all DATs and web and dispatcher levels. **See document for details**.	Needs to be activated and configured via MMC, and recommended to always be on in Live. **See document for details**.	Needs to be activated (we have template PerfMon log files to make this simple) and then should always run	We recommend that Perf by Request, Perf Auditing and PerfMon counters are always enabled on a Live system along with either Portrait Event Log logging *or* Portrait Log to file.
Is there any impact to the servers when implemented?	Some overhead depending on the configured filter levels. We recommend that the same filter as the Event Log is used in a Live environment.	Small overhead depending on the configured filter levels. We recommend the default filter in a Live environment.	Some overhead depending on the configured filter levels. We do not recommend running the default settings in a Live environment.	MDT Interactive or Recorded logging has a significant impact on the server's performance. Only use it in Dev or Test environments.	Typically less than 2% performance impact when used with all DATs turned on.	No significant overhead	no significant overhead	If any Portrait server is running at 100% capacity, then a small impact can affect performance, but if the server has some extra capacity, then any minor performance reduction should be fine as the benefits far outweigh the impact.
Does it fill up drives (disk); create logs or increases memory usage?	The file path, name, max size, flush time and max # files can all be set in MMC. Limited memory footprint increase when used.	If too much is logged, then the Event Log will fill very quickly causing it to overwrite older entries. Limited memory footprint increase when used.	The file path, name, max size, flush time and max # files can all be set in MMC. Limited memory footprint increase.	The MDT logs its data to the database. Individual session traces can be removed from the database using the MDT.	It will create new log files of the size specified in the MMC. These will need to be periodically cleared down from the disk. No significant memory footprint increase.	Data is stored in database, and an SP is included to purge it as required. No significant memory footprint increase.	The PerfMon counter logs are saved to disk and combined will slowly grow at about 8.5MB per day. These can be manually purged after a relevant period of time.	As part of any administration jobs, the availability of server disk space and database table sizes should be monitored and controlled. We normally supply SPs for database record purging, but expect our customers to use their own admin tools for file system housekeeping.
Notes about this logging type	The filter settings in the MMC make the difference between no significant impact and a huge performance impact. If you were to log just errors, then the impact would be negligible, but if you were to log everything (which might result in an output of many hundreds of rows per second) then the server's performance would be significantly impacted. Please contact Portrait Support if you feel that one of these Portrait Foundation Logging destinations is not logging enough or is too verbose.				The number of rows created in the logged file will depend upon the number of users, the number of DATs selected, and the number of models and nodes in the configuration. There is an XSL template that helps with the interpretation of this log's csv output.	The number of rows created in the logged file will depend upon the number of nodes selected, and the number of models and nodes in the configuration, and the frequency of the logging interval. Once every 15mins will have about 100 times as many rows created as if the interval is once per day.	There are two types of logging advocated; Summary logging and Full logging. The summary logging logs less information, but at more frequent intervals.	

9 References

9.1 Glossary

Term	Description
business component	A Portrait Foundation component that provides business-functionality and is visible to an analyst who is configuring the system. Examples include the process engine and code nodes supplying customer functionality.
Business object	A soft-configured object representing a party, product or campaign. Business objects are defined through the Configuration Suite and are represented at run time by data objects.
Business transaction	A transaction representing a business operation, such as amending a customer's details or applying for a new product.
Channel Enabler	A set of components that make the generic CRM server functionality available through a particular interaction channel.
Code node	A Process Engine component that can call a piece of bespoke functionality implemented in any language that supports COM interfaces.
Configuration Suite	A consistent set of tools within a single framework. The Configuration Suite is a Portrait Foundation deliverable.
Configuration time	This refers to the implementation and maintenance stages of a project when the behavior of Portrait Foundation components are defined by the business community. See also **run time**.
CRM server	The term collectively applied to the set of components that are deployed on an application server in the middle tier.
Data object	A configurable object that represents a particular business entity at run time.
Deployment	The process of making configuration information available to run time components.
Party	A person or organization known to the Portrait Foundation system. Examples of parties include customers, Contact Centre agents and third-party suppliers.
Plug-in	A component that can be integrated into the Configuration Suite.
Run time	This refers to the dynamic behavior of the Portrait Foundation system, as opposed to configuration time. A distinction is drawn because there are many components which are only applicable to one or other of these environments.
System component	A Portrait Foundation component that provides system-level functionality and is not generally visible to an analyst who is configuring the system. Examples include inter-tier communication mechanisms and the logging component.
MDT	Model Diagnosis Tool. The Portrait tool used for both recording and playback of sessions or models.
MMC	Microsoft Management Console. A Microsoft technology used as a standardized container for various configuration and reporting utilities. The Portrait Management Console is an example used for the configuration of various Foundation settings.
UIE	User Interface Element. The old name for a Portrait Custom Control which is a configuration item that is developed as a user interface control called from a web page. It usually receives data using a Data Access Model (DAM).
Split Box	A server environment where the application server processes are installed on one (or more) servers and the web server processes are installed on one (or more) other servers.
Activity Token	A GUID used to identify a single user session. Whenever a request is made to Portrait, an Activity Token is usually passed to ensure that the state is maintained.

9.2 Other Recommendations

Name	Description
Logging	Logging should be turned on at all times. The minimum logging should be to the Event Log using the default filter [(2):!(1):(*):(*)]
Log Filters	The logging filters determine which information gets logged. This can be modified to show more or less information. Using the filter tool in the Log Viewer is the best way to understand the filters. More information can be found in the Operations_Guide and the Logging_Usage_and_Standards_Guide.
Log Filter : Overly detailed	The logging filter "[(*):(*):(*):(*)]" logs everything and this can cause a big drain on the server. Therefore this is not recommended in a production environment. This may be very useful in a development or Test environment.
Cache settings	There are several cache settings that can be set in the Management Console. It is not possible to suggest one size fits all recommendations because each implementation is unique. Your system should be reviewed to determine the optimum cache settings which are appropriate to your system.
PerfMon	PerfMon should always be turned on, with settings provided by Portrait Software. We have template files which turn on the appropriate settings. If this has not been done by your implementation team, you should request this from Portrait Software

9.3 Logging Default Filters

Filter	Default filter
Debug	[(2):!(1):(*):(*)]
File	
Database	
Event Log	[(2):!(1):(*):(*)]
Viewer	[(*): (*):(*):!(1,2,3,4,5,12)]
MDT	[(*):!(2):(*):(9,10,11,13,14,16)][!(1):!(2):(*):(0)][(*):(*):(*):(12)][(2):!(1):(*):(*)]

9.4 Special Logging Filters

Recommended Special Purpose Filters	Portrait Support can provide you with specific log filters Use the spaces below to note any special filter settings you are given or that you use
[(2):!(1):(*):(*)]	Standard default logging
[(*):(*):(*):(*)]	Overly detailed logging (logs everything)