Portrait
Foundation

Operations Guide

Edition 14.0

28 July 2014

Pitney Bowes
Software

Portrait Foundation
Operations Guide

Acknowledgement of trademarks

Other product names, company names, marks, logos and symbols referenced herein may be the trademarks or registered trademarks of their registered owners.

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

Purpose of document

This document is intended to provide Operations staff with guidelines on how to ensure their Portrait Foundation system remains operational and performing at acceptable levels.

Intended audience

Anyone interested in the current and future operation of a Portrait Foundation implementation. This is likely to include Operations staff responsible for maintaining a live Portrait Foundation system and those responsible for planning roll outs.

Related documents

Performance and Scalability White Paper

Management Console help

Live Updates User Guide

Problem Determination Overview

Database Administration Guide

Model and Node Counters Reference

Software release

Portrait Foundation 5.0 Update 1 or later.

Contents

1 Components

1.1 Overview

The Portrait Foundation runtime system is installed as a set of services on the servers that comprise your Portrait Foundation system. This section gives a brief description of the role performed by each Portrait Foundation service within your system.

For each Portrait Foundation service, Table 1 summarises:

- The processing performed by the service.
- The server within your Portrait Foundation system on which the service resides.

Table 1 – Portrait services

Service name	Functionality	Server Type
AMC ServiceHost	The main container for Process Server based Portrait code. It contains Portrait 'services' that conform to the Portrait Service Host architecture in a single service. See section 1.2, below for details on functionality provided by service hosting.	Process
AMC SysConfig Auxiliary	This service runs on devices where ServiceHost does not and is responsible for setting and updating registry settings that are configured in the Portrait Management Console.	
AMC Automatic Tasks Engine	Polls for the presence of automatic tasks in the operational database. Executes such tasks by running the relevant Process Model.	Process
AMC Web Channel Service	Passes requests from the Web channel to/from the Process server.	WEB
AMC Collector Service	Allows data existing within the Portrait infrastructure to be published to external components (e.g. Perfmon)	Process
AMC Hint Server	Helps the Object broker to route incoming requests to the Process server that last processed a request for the current session.	WEB
AMC Log Message Concentrator	Forwards log messages to clients such as the Portrait Model Diagnosis Tool	Process
AMC Log File Writer	Manages the writing of log messages for the 'File' type destination. This need not be started if 'File' log destinations are not enabled in the Portrait Management Console.	Any
AMC Message Queue Listener	Polls for e-mails/documents and passes them to: The Process server for inbound e-mails/documents The relevant external message processing system (e.g. MS Exchange)	Process for inbound e-mails/documents SIMPLEX/DUPLEX for outbound e-mails/documents
AMC QAS Service	Allows access to QAS Address lookup.	Process
AMC Telephony Server	Provides a middleware-independent call management channel to a PTS client.	TELEPHONY
AMC Exchange DMS Inbound Adaptor Service	Puts inbound e-mails from MS Exchange onto a queue, ready for reading by Portrait Foundation.	SIMPLEX/ DUPLEX

1.2 Service hosting

Service hosting allows flexible packaging of Portrait Foundation components that were previously packaged as individual Windows services. This enables:

- Consolidation. Many Portrait Foundation components bundled as a single service – providing both performance and administrative benefits.

- Multi-instance. More than one copy of a specific piece of Portrait Foundation functionality, for example, multiple Process Engines. This allows full exploitation of the processing power of large multi-processor machines like the Unisys ES7000.

Each 'package' of Portrait Foundation functionality resides in one or more ServiceHost services. An XML configuration file determines the content and number of each ServiceHost service. This file is referred to by the value `ServiceConfiguration` in the registry key

`HKEY_LOCAL_MACHINE\SOFTWARE\PST\Portrait\Systems\<MySystem>`

By default, Portrait Foundation is configured to use "*Install_directory*\Portrait Foundation\Systems*<MySystem>*\Data\ServiceConfigConsolidated.xml"

which defines a single ServiceHost service that consolidates all the 'hostable' components. The following sub-sections describe the functionality provided by this consolidated ServiceHost service.

Contact Portrait Support to obtain advice on how to re-configure service hosting if your system could benefit from the flexible service host architecture, for example, to fully exploit hardware with a high number of processors.

1.2.1 Parameters

This is the service that caches the Smart Lookup data.

1.2.2 Process Engine

This service supports execution of Portrait Foundation Models.

1.2.3 System Configuration

This service supports access to operational parameters. These are mostly set up by the Portrait Management Console.

1.2.4 Reference Data Global Cache

For performance reasons (to avoid database access) reference data groups and reference data items are cached by this service. This cache is accessed from components that are on the Web tier via DCOM.

1.2.5 Data Object Factory Global Cache

For performance reasons (to avoid database access) the structure of DataObjects (for example, Consumer) are cached by this service. This cache is accessed from components that are on the Web tier via DCOM.

1.2.6 Resume Engine

Polls for the presence of active Delay nodes. If the delay period has elapsed, this service resumes the model containing the delay node.

1.2.7 AutoTask Engine

Polls for the presence of active Create AutoTask nodes. If the activation delay period has elapsed, this service resumes the model containing the Create AutoTask node.

1.2.8 Edge Gateway

Enables communication with Edge through Portrait Foundation nodes.

1.2.9 State Management

Handles synchronising model state and session management state database tables.

1.2.10 Client Events manager

Manages external interactions for Client Events.

1.2.11 Local caching

Provides caches for many frequently access database tables (Model inputs, Data Access Class definitions, etc.)

1.2.12 Session Management Global Cache

This service provides access to the database for state management. It is access from the Process server and from the Web tier.

2 Tools

Portrait Software provide a number of tools that are used for the administration and maintenance of a Portrait Foundation System. These tools fall into a number of categories such as operational tools, diagnostic tools, and capacity planning tools.

The following is a brief listing of the most commonly used tools:

Tool	Usage	Type	Further info
Portrait Management Console	Primary tool for setting the runtime characteristics of a system	Operational/Setup	Application Help
Object Broker Management Console Snap-in	Used to configure Portrait Object broker	Operational/Setup	Application Help
ServiceCheck	Performs an orderly start and stop of Portrait services and IIS on a particular device	Operational	Problem Determination Guide
Log Viewer	Enables viewing of either live feeds of diagnostics from a running system or can open saved files. It can handle both Windows Event Viewer files (.evt, .xml) and Portrait Log (.plf) files.	Operational/Diagnostic	Problem Determination Guide
Model Diagnosis Tools	Provides the ability to execute a given model	Diagnostic/ Development	Problem Determination Guide
Model and Node Profiler	Analyses the data collected regarding the runtime execution of models and nodes within a system	Diagnostic/ Development	Model and Node Counters Reference
Version Display Tool	Reports on the versions of binaries installed. Portrait Support may require this information as part of problem determination.	Setup/Diagnostic	Problem Determination Guide
Portrait Performance Monitoring (AKA Cecil)	This utility can be used to capture and report on system performance.	Capacity planning/diagnostic	This document

3 Initial Setup

In order to be able to support an environment, it is necessary to set in place certain procedures and perform a number of preparatory tasks.

This section documents the tasks and processes that should be undertaken as part of initial system commissioning. A number of these tasks should be repeated either regularly (e.g. a health check) or whenever a change is made to the environment.

Portrait Support provide checklist for preparing a system for operational use. The document is called Environment Configuration Checklist.

3.1 Perfmon

Portrait Foundation processes produce Perfmon counters that can be used to provide diagnostic information and can provide input to system health assessment and capacity planning.

Portrait Software recommend capturing all of the Portrait Perfmon counters for all processes that create them. Portrait Software also recommend a list of Windows Perfmon counters that should be collected for servers with Portrait Foundation installed.

Some of the counters should be reviewed in a daily basis to determine whether there are issues with the system (e.g. 'Portrait Process Engine\Hardware exceptions' which indicates that a memory dump has been created because of an error) .

These counters should be captured and archived for problem resolution, historical comparison, and capacity planning purposes.

Portrait Support provide a document describing the recommended configuration for Perfmon in a Portrait Foundation environment. This document is called Windows Performance Monitor Configuration.

3.2 Model and Node counters

These counters can be used to profile the models and nodes executed within the system. This information should be captured by default (there is minimal overhead in capturing this data). Portrait Support may request this data as it provides a useful insight into the operation of a system.

These counters and how to use them is documented in Model and Node counters reference document.

3.3 Logging

Portrait Foundation produces diagnostic information that can be subsetted and delivered to many different 'destinations'. A given message can be routed to more than one destination and messages can be filtered out so that they are not delivered to any destination.

In the recommended settings section it is recommended that all Portrait Foundation servers use a particular filter setting that will cause serious messages to be written to the Windows Eventlog. The default install setting for the destination 'Eventlog' should enable this.

In addition to this, Logging to file should also be enablde. Incorrectly enabling other destinations can impact system performance.

The Windows Eventlog should be configured to capture information without overwriting old entries. There also needs to be a process in place to review and archive eventlogs.

Efforts are made to reduce the records written to the eventlog to only those items that should be actioned – i.e. every 'error' entry indicates something that needs corrective action. Sometimes 'error' entries are created for situations that are normal, known, or do not require corrective action. In this case Portrait Software can arrange for the these log messages to be suppressed. Suppression is done by the creation of a file referred to by the registry key `HKEY_LOCAL_MACHINE\SOFTWARE\PST\Portrait\Systems\<MySystem>\Logging` value "`Filter`". By default this is *`install_directory`*`\Portrait Foundation\Systems\<MySystem>\Data\LogFilter.plx`. Obviously care has to be exercised to ensure that vital alerts from the system are not suppressed; that is why the creation of .plx files is undertaken by Portrait Support rather than as a client action.

3.4 MiniDump

In circumstances where unusual events occur within a running system it is necessary to have the system set up to capture enough diagnostics to be able to resolve problems. Part of this is to install a mechanism for capturing memory dumps when 'exceptions' occur. Minidump is the Portrait Software developed utility that is used for this purpose.

Minidump should capture minimal diagnostic information when errors occur in Portrait Foundation services. The list of services to monitor is recorded by default under `HKEY_LOCAL_MACHINE\SOFTWARE\PST\Portrait\Systems\<MySystem>\MiniDump`. Memory dumps are produced by the services themselves invoking the utility and by setting the mortem debugger to intercept unhandled exceptions via the registry key `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AeDebug`.

It is also important that a procedure is put in place to capture these memory dumps as soon as reasonably practical after they occur and forward them to Portrait Support. By default they are written to the directory that Portrait Foundation is installed in as .dmp files. By default these files are not large and only contain minimal diagnostic information. If necessary Portrait Support may request changes to the default Minidump settings to gather more information than is the default. It is important to note that these memory dumps are not the massive full memory dumps that are often associated with the term 'memory dump'

Minidump can be used as a command line utility to produce memory dumps ranging from the minimal to full memory.

Minidump is described in the Debugging Tools document.

4 Housekeeping

4.1 Health

The daily tasks that need performing are referred to elsewhere in this document and are stated in the Portrait Support document Portrait System Health Checklist.

They can be summarised as:

- Review the Perfmon data for indicators of problems (e.g Failed Models, Hardware exceptions)
- Review the Eventlogs for errors that need actioning
- Ensure all memory dump (.dmp) files are harvested, passed to Portrait Support, and deleted.
- Archive the Perfmon, and Eventlogs. Ideally the archive should allow Portrait Support to request the Perfmon and Eventlogs that match any problems raised – application or system issues.

4.2 Database

The Database Administration Guide provides details of the housekeeping and maintenance tasks that are necessary a Portrait Systems database.

5 Problems

It is important that all of the checklists provided by Portrait Support and initial setup steps have been executed when a system is commissioned (i.e. prior to encountering problems). This will significantly quicken problem resolution.

Portrait Support provide a guide for people developing or supporting a Portrait Foundation implementation. This document is called Diagnosing Performance Issues in a Portrait Environment.

The document Debugging Tools describes some very low level debugging techniques that may be useful to those providing support for a Portrait Foundation system.

6 Capacity Planning

The primary input to capacity planning is the Perfmon counters – both the Windows and Portrait Foundation counters.

Model and Node profiling can also provide useful input to this process.

In order to be effective, the above information needs to be captured and archived as recommended in this document.

The following section describes the Portrait Performance Monitoring (AKA Cecil) utility which is valuable in capacity planning and some performance problem determination scenarios.

6.1 Portrait Performance Monitoring (AKA Cecil)

This tool can provide valuable diagnostic and capacity planning information that Portrait Support may request as part of problem determination. It is described below.

6.1.1 Portrait performance monitoring

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.

A record is written each time a Portrait Foundation action is requested. An action can be one of:

- Business Operation – Starting or resuming a business-level operation, such as amending an address.
- Model – Starting or resuming a Portrait Model, such as the framework model which controls the Contact Centre application.
- Custom Control (also known as UIE) – Retrieval of data to populate part of a page, such as the portfolio grid.

A number of fields are captured for each action. The subset of fields relevant to Basic Monitoring is as follows:

Field	Description
Timestamp	The timestamp represents the time the operation was started.
ASPScript	This represents the name of the ASP script that initiated the Portrait Foundation operation.
CRMServer	The name of the Process server that satisfied the Web Server request. If a cache on the Web Server was involved in satisfying the request then the cache is specified.
Duration	This is the time taken to execute the given operation, as observed at the Web Server. The time is shown in milliseconds.
OperationType	This is the type of operation (action) that has been requested of the Portrait Foundation Process server. - Operation, Model or Custom Control (UIE Data).

Field	Description
Resume	All Operations and Custom Controls are initiated exactly once. Operations may additionally be resumed one or more times. This flag indicates whether a user request was an initiation (zero) or a resumption (non-zero).
OperationName	This is the name of the Portrait Operation, Model or Custom Control that is being executed. In the case of custom controls each custom control being executed is listed, separated by ':'

The following is an example of a few lines from a Performance Monitoring log file.

Timestamp	CRMServer	Duration	OperationType	Res	OperationName
19/11/2002 13:50	PPCRM1	672	Message:ExecuteModel	0	ContactCentre
19/11/2002 13:50	PPCRM1/UIECache	454	Message:RetrieveUIEData	0	GenericSideBar1:PartyEngagementHistory
19/11/2002 13:50	PPCRM1	140	Message:ExecuteOperation	0	WrapUp
19/11/2002 13:50	PPCRM1	281	Message:ExecuteModel	1	ContactCentre
19/11/2002 13:50	PPCRM1	532	Message:RetrieveUIEData	0	ICRMBusinessViewNoTrace:ICRMPasswords
19/11/2002 13:50	UIECache	0	Message:RetrieveUIEData	0	GenericSideBar1

Capturing Portrait performance monitoring data

Portrait Performance Monitoring is configured through the Portrait Management Console. The Management Console help provides full details of the configuration of Portrait Performance Monitoring. The process is summarised here.

It is not possible to schedule Portrait Performance Monitoring in advance, so it is necessary to turn it on at the start of the test and turn it off at the end. (Note that in some implementations, Performance Monitoring will be switched on all the time. In this case it is not necessary to change the settings at the start and end of the test).

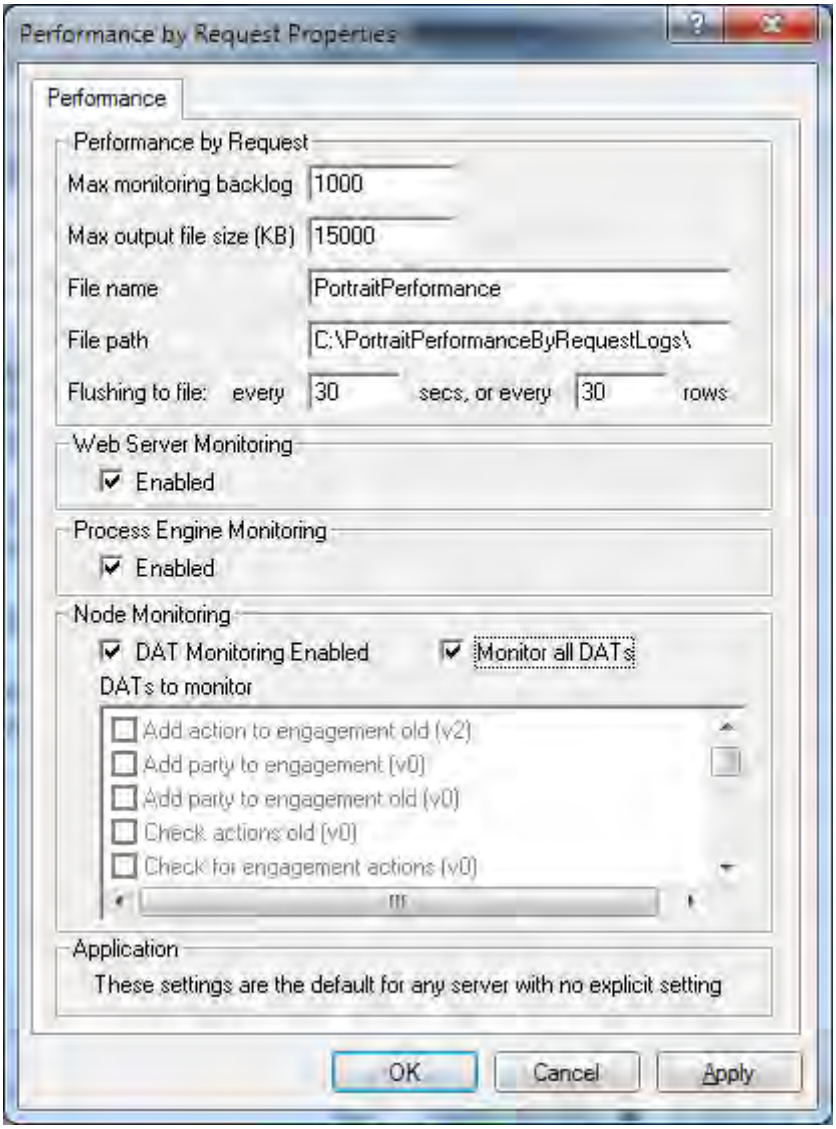
Performance by Request logging creates text based (.csv) files locally on each server. It is recommended that a specific directory is created for storing these files. The document assumes that you create a directory called

`C:\PortraitPerformanceByRequestLogs.`

Performance by Request logging is configured using the Portrait Management Console. Right click on the Performance item under "All Servers→Default settings", and select "Properties Configure the "Performance Properties" as per the dialog below.

Configure the "Performance by Request Properties" as per the dialog below.

Figure 1 – Configuration of Performance Monitoring settings through the Management Console



Update the settings to use the following values.

Max monitoring backlog should be **1000**

Max output file size (KB) should be set to **15000**

File name should be **PerformanceByRequest**

File path should be **C:\PortraitPerformanceByRequestLogs**
(note: including the trailing backslash)

Flushing to file every **30** secs, or every **30** rows

Web Server Monitoring should be **Enabled**

Process Engine Monitoring should be **Enabled**

DAT Monitoring should be **Enabled**

If Monitor all DATs is enabled, then the list of specific **DATs to monitor** is disabled and may be ignored.

Click the **Apply** button to start logging the information. At the end of the test, clear the **Enabled** check boxes and click **Apply** again.

As each request is processed by a server, an entry is written to the log file configured. For performance reasons, the log file is updated in batches (based on flush time and max monitoring backlog) rather than being physically updated each time one new line needs adding.

A new log file called "PortraitPerformanceByRequest.csv" will be created whenever the current one reaches 15 MB in size. Old log files are renamed to include the date & time that they reached 15 MB in size.

Processing Portrait performance monitoring data

Response times can be calculated by loading each Portrait Performance Log file into a spreadsheet or database. There will be at least one log for each Web Server in the system. Only those records with a timestamp within the range of the test should be included in the calculations.

The required information can be calculated by taking the average of the values in the Duration field. If required, this can be broken down by Operation Type and Operation Name and supplemented by the maximum time, although these are not within the scope of Basic Monitoring.

6.1.2 Summarising the data

Once the average response times and CPU usage have been calculated, an overall green, amber or red rating can be applied to each server. The definition of these ratings is:

Green	The server is operating well within the expected limits. No action is currently necessary.
Amber	The server is reaching its capacity and performance may start to degrade. The situation should be investigated and remedial action taken if appropriate.
Red	The server is seriously overworked. Immediate action should be taken to resolve the problem.

Note that the response time element of the rating can be affected by 'downstream' servers performing poorly. The response time observed at a Web server will depend primarily on the response time of the Process Server that it calls to process a request. This should be taken into account when analysing the ratings of each server.

The following table provides some guidance for assigning a rating to servers, based on the average response times and CPU usages. The guide numbers may be changed on a per-site basis to reflect local requirements and service level agreements.

	Green	Amber	Red
Average CPU	Under 60%	60% to 80%	Over 80%
Average Response[1]	Under 700ms	700ms to 1s	Over 1s

The overall rating for the Process and Database servers is simply the CPU usage as this is the only measurement captured. However, the rating for a Web Server will be the highest of the CPU and response ratings. For example, if a Web server has a Green rating for CPU usage and an Amber rating for Response times, it will have an Amber rating overall.

[1] Note: The average response time refers to the time taken for the Web Server and Process Server to process the request and does not include any network and client rendering time.

The overall rating for the system is determined by the worst-case server. If one of the Process Servers has a Red rating, then the whole system is rated as Red even if all the other servers are Green.

6.2 Usage profiling

Usage Profiling provides a measurement of the amount and type of work being carried out by the system. This can be compared with predicted usage patterns to help validate capacity planning assumptions. If predicted and actual usage patterns are significantly different, it may be necessary to undertake a new capacity planning exercise.

The primary outputs of Usage Profiling are:

1. Overall throughput figures for the system, expressed in terms of **User Actions per Second**. The average throughput per Process Server can also be derived.
2. A summary of all user actions being carried out, which helps determine the usage patterns.

6.2.1 User actions

A User Action represents a single request from a user to a Portrait Foundation server, causing one or more Models to be executed. Examples of User Actions include:

Initiating a new Operation, such as Change Address.

Submitting data into a Portrait Foundation page, such as a Generated Interaction or a Custom Interaction, which causes an Operation to be resumed.

Navigating to a page within the Consumer Desktop – for example Engagement History.

The number of User Actions theoretically supported by a server of a given size is specified in the Portrait Performance and Scalability white paper [ref 1]. However, it should be noted that this number is an estimate based on 'average' actions, which may vary from implementation to implementation.

The white paper also specifies a theoretical maximum throughput for the database server, based on support for a given number of Process Servers.

6.2.2 Capturing the data

The raw data required for Usage Profiling is the same as for Basic Monitoring – Portrait Performance log data. This should be captured as described in section 6.1.1, above.

6.2.3 Processing the data

Each of the Portrait Performance Log files should be loaded into a spreadsheet or database. There will be at least one log for each Web Server in the system.

All records representing Models and Operations relate directly to a User Action. Some (but not all) Custom Controls also equate to a User Action, such as selecting a tab page in the customer folder. However, some Custom Controls are invoked in order to populate a page that is part of another process, such as the identification of a customer.

To prevent double counting of User Actions, it is necessary to understand which Custom Controls are User Actions in their own right and which ones are invoked during the execution of another action. By manually inputting this data into a spreadsheet or database table as a one-off activity, it is possible to use it to filter

out non-Action Custom Controls from the log data before processing it[2]. A simpler solution is to exclude all Custom Controls from the log data before processing it. This will result in a lower (and less accurate) throughput figure, but may be satisfactory for tracking trends in data where absolute numbers may not be required.

Once the appropriate Custom Controls have been filtered out, the log records whose timestamp is within the range of the test should be selected for inclusion in the calculations.

The total User Actions during the test can be obtained by counting the number of records in the filtered log data. Dividing this by the length of the test (in seconds) gives a figure for the average User Actions per Second during the test. Dividing this by the number of Process Servers in the system gives the average throughput of each Process Server.

If required, the data can be further queried to produce a report of the number of times each Operation, Model and Custom Control is executed.

6.2.4 Interpreting the data

It is common practice to undertake a capacity planning exercise during the early stages of a project, before the system goes live. Amongst other things, this exercise should predict the overall server throughput (in Actions per second) during a 'busy period'. For example, capacity planning may have predicted an average throughput of 10 Actions per second across the whole system.

The throughput measured in section 6.2.3, above, should be compared with the capacity planning number to identify any anomalies. For example, if the measured throughput is 5, then the current usage appears to be only half of that predicted. The business may want to investigate why the system is not as heavily used as was predicted. Conversely, if usage is higher than predicted it may be necessary to update the capacity planning figures accordingly.

The User Actions per Second numbers should also be compared with the theoretical throughput given in the Performance and Scalability white paper. For example, if the theoretical throughput of a Process Server is 10 Actions per second and the measured throughput is 6 per second then the load is 60% of the theoretical maximum. Comparing this 'theoretical load' percentage with the actual CPU usage figure (the 'observed load' from the Basic Monitoring section) gives an indication of whether the servers are scaling as predicted.

[2] Note: Setting up the list of Action and non-Action Custom Controls requires knowledge of the individual Operations and Custom Controls in the application.

7 Operational Settings

This section describes some recommended settings for Portrait Foundation servers. Items not covered in this section will be covered in the Portrait Management Console Help.

These recommendations are based on a number of values such as maximum peak users. These settings should be reviewed in the light of significant changes in any assumptions or the environment. This document contains definitive calculation or formulae to derive the settings. Other documents, such as the Portrait Management Console Help, are reiterations of these recommendations.

7.1 Assumptions

Devices running Portrait Foundation are homogenous devices (within an environment) with an equal share of any workload.

Where there is a marginal choice between virtual memory and performance; err on the side of saving virtual memory.

The settings recommended will be common across all environments unless otherwise specified.

7.2 Management Console

The following headings are arranged in the order presented by the Portrait Management Console under the All Servers heading.

7.2.1 AutoTask\Automatic Task

The AutoTask service polls regularly for any active Create AutoTask nodes. Once the configured activation delay period has elapsed, the AutoTask service resumes the model containing the Create AutoTask node.

How you configure this depends on the nature of your Portrait Configuration: if you make extensive use of Create AutoTask and/or Delay nodes you might wish to have one server dedicated to the AutoTask and Resume services and not handling user requests at all. This will prevent impact on response times when processing AutoTasks and Delay nodes: the AutoTask and Resume services would be enabled on this server but not on any others. If resilience is important, you might wish to double this up and have two such dedicated servers.

Alternatively, you might enable the AutoTask service on one or more of your general use process servers. We recommend that you only enable it on one server (or two for resilience), to avoid having many servers in contention to process the same tasks - every server on which it is enabled will poll at the interval specified.

Enabled: As described above **Rationale**: Enable on just one or two servers.

Polling interval (sec.): 360 **Rationale**: The maximum time an AutoTask may have to wait beyond its configured activation delay is the interval between successive schedules of the SQL Server database job amc_te_upd_activation_*DatabaseName*, plus the polling interval configured here.

7.2.2 Caching\Cache Options

Smart Lookup

Enabled: Checked **Rationale**: Avoids the IO/build cost at little memory cost.

Cache Size: Calculated

Tuning: This value should not be less than the number of rows returned by the stored procedure `exec p_amc_prm_get_all_params`. A low value can adversely affect response times. There is little cost in setting a high value.

System Config

Enabled: Checked **Rationale**: Avoids the IO/build cost at little memory cost.

CBE Definition

Enabled: Checked **Rationale**: Avoids the IO/build cost at little memory cost.

Max properties: 2000

Tuning: If the Perfmon counter Portrait Cache Counters\CBE Definition Cache Misses is non-zero then this number needs to be increased. A low value can adversely affect response times. There is little cost in setting a high value.

Reference Data

Enabled: Checked **Rationale**: Avoids the IO/build cost at little memory cost.

Max items: 5000

Tuning: The max items value should not be less than the number of rows in `amc_rd_ref_data_item`. A low value can adversely affect response times. There is little cost in setting a high value.

Max group stale time in minutes: 0

This determines how often the system checks if external reference data groups have changed. If this value is zero they are re-loaded every time they are accessed. If your external reference data groups are not changed frequently, or if changes do not need to be seen at once, set a higher value here. This has no effect on reference data groups that are not external.

Script Definition

Enabled: Checked **Rationale**: Avoids the IO/build cost at little memory cost.

Max properties: 500

Tuning: This value should not be less than the number of scripts in your Portrait Configuration. A low value can adversely affect response times. There is little cost in setting a high value.

7.2.3 Client Events\Client Events

Generally, the default values provided will be suitable here.

7.2.4 Data Object Factory\Caching

Enabled: Checked **Rationale**: Avoids the IO/build cost at little memory cost.

Set to Maximum: Enabled

Tuning: A low value can adversely affect response times. There is little cost in setting a high value.

7.2.5 Encryption\Encryption

Encryption of Web server session data

Enabled: Unchecked **Rationale**: Encryption of session data has a performance cost, so best performance will be achieve with this switched off.

Encryption of Process server session data

Enabled: Unchecked **Rationale**: Encryption of session data has a performance cost, so best performance will be achieve with this switched off.

7.2.6 Generated Interactions\Caching

Enabled: Checked **Rationale**: Avoids the IO/build cost at little memory cost

Prefetch domains: Checked

Set to Maximum: Checked

Tuning: A low value can adversely affect response times. There is little cost in setting a high value.

Prefetch: All Domains Latest version

7.2.7 Messaging\UIE Handler

Thread pool size: This value should be the same value as dispatcher threads **Rationale**: Setting it to this value reduces the likelihood of a UIE request being artificially delayed waiting for a thread.

Wait Timeout: 60 **Rationale:** Long enough to avoid 'false' timeouts. It is an implementation decision as to when a UIE response results in a timeout.

Tuning: If the thread pool can be reduced without impacting throughput the savings in virtual bytes are 1MB per thread.

7.2.8 Process Engine Caching Settings\Definition Caching

Enabled: Checked **Rationale:** Model definitions are time-consuming to load, so some degree of caching is highly recommended.

Set to maximum: Unchecked **Rationale:** Model definitions are the largest single entities in the system – too large a value here can result in your process running out of virtual memory.

Cache size: 800 **Rationale**: As a rule of thumb, 1 entry equates to about 1 MB of virtual bytes, though this depends on the size of your models. The down side is that if more than this number of top-level models are used, performance will degrade when models that are not in the cache have to be built.

Tuning: In a well-tuned system, the Perfmon counter Portrait Cache Counters\Model definition cache misses/sec should be at or near zero. For example a value of 0.1 would mean that a model is being loaded from the database once every ten seconds, on average. If it is higher than this, consider increasing the size of this cache but be sure to monitor the perfmon process counter "virtual bytes" to check that the process is not consuming too much memory. Before using either of these perfmon counters for tuning it is important for the system to achieve steady state – probably after a day or two of operation. NB: It is only possible to tune this using the values from a Live system, or a system that is experiencing exactly the same load profile as Live.

Prefetch: Unchecked **Rationale**: As the number of entries is below the number of model definitions sensible prefetching becomes difficult to achieve. It can only be usefully turned on if the prefetch list is less than or equal to the size of the cache. Analysis of the Model and Node audit counters may provide some help in determining which models, if any, should be prefetched.

7.2.9 Process Engine Caching Settings\State Caching

Enabled: Checked **Rationale**: Avoids the IO/build cost at little memory cost.

Shared: Checked **Rationale**: Clearing this checkbox activates a performance optimisation in the case of a system with a single process server. If there is more than one process server they all share model state and it is essential to check this box. It is always safest to have this setting checked. In circumstances where you are certain there is only one process server this may be unchecked.

Cache size: Calculated **Rationale**: This is 10% above the peak concurrent users of this device.

Tuning: This value should be kept high enough to keep the Perfmon counter Portrait Process Engine\Model State Cache Misses at or near 0 for a fixed community of users.

7.2.10 Process Engine General Settings\DAT settings

Use SQL sub-SELECT clause in CBE search queries: Checked **Rationale**: This generally gives better performance, but is provided as a setting to allow reversion to older behaviour for backward compatibility.

7.2.11 Resuming\Resuming

The Resume service polls regularly for any active Delay nodes. If the configured delay period has elapsed, the Resume service resumes the model containing the Delay node.

How you configure this depends on the nature of your Portrait Configuration: if you make extensive use of Create AutoTask and/or Delay nodes you might wish to have one server dedicated to the AutoTask and Resume services and not handling user requests at all. This will prevent impact on response times when processing AutoTasks and Delay nodes: the AutoTask and Resume services would be enabled on this server but not on any others. If resilience is important, you might wish to double this up and have two such dedicated servers.

Alternatively, you might enable the Resume serviceon one or more of your general use process servers. We recommend that you only enable it on one server (or two for resilience), to avoid having many servers in contention to process the same models - every server on which it is enabled will poll at the interval specified.

Enabled: As described above **Rationale**: Enable on just one or two servers.

Resume interval (sec.): 1800 **Rationale**: This is a polling interval and hence determines the maximum time a Delay node may have to wait beyond its configured delay before being the Workflow Process Model gets resumed and processed further.

7.2.12 Session Management\Caching

Local Properties

Entries: Calculated **Rationale**: 5*peak concurrent users

Tuning: This setting is high enough to largely avoid cache misses for a constant set of users. The Perfmon counter Portrait Session Manager\Session Management Local Cache Misses should be should rarely be above 0.

Global Properties

Entries: Calculated **Rationale**: 5*peak concurrent users

Shared: Checked **Rationale**: If there is more than one process server they all share session management data. It is always safest to have this setting checked. In circumstances where you are certain there is only one process server this may be unchecked.

Tuning: This setting is high enough to largely avoid cache misses for a constant set of users. The Perfmon counter Portrait Global Cache\Session Management Global Cache Misses should be should be rarely above 0.

7.2.13 Session Management\Security

Anti-replay validation

Expire requests after: 360 seconds **Rationale**: This determines the maximum length of time after which two requests with the same request counter will be rejected. Duplicates can happen during normal running of Portrait, but if they are not closely spaced in time they may represent a replay attack on the system.

Sequence tolerance: 0 **Rationale**: This value determines the maximum sequence tolerance for the internal request token sequencing. Under normal circumstances, the resuest token sequence number used in transaction requests to the web controller is allowed to be either *equal* to the previous number or *one more* than it. This is the behaviour that will be seen when this value is set to its default of zero. Under certain circumstances, however, it is possible to contrive a scenario whereby the sequence can get out of order but not as a result of an actual replay attack. If you find that you are periodically seeing these *false positives* resulting in the suggestion of replay attacks of type 'out of sequence', then you should raise a ticket with Portrait Support and supply them with the usual logs. In the interim, so that your Live operations are not interrupted, you can change this setting to a value of 1. This will allow a 'softening' of the sequence rules so that the current sequence is allowed to be either *one less* than the previous sequence number or *two more* than it.

We do not recommend setting this value to anything greater than 1 without approval from Portrait Support. The standard setting should be 0 (zero).

Include client IP address: Checked **Rationale**: Provides a higher degree of security against possible attacks on the system. When selected, the system will check that all transactions within a specific activity context originate from the same client IP address. This should only be turned off if you receive *false positives* of type 'Invalid IP' during your normal transaction processing. This can sometime happen if your system is using a mixture of IPv4 and IPv6, or if you are using an incorrectly configured load-balancer.

7.2.14 Threading\Dispatcher threads

Dispatcher threads: Calculated **Rationale**: This value is very much dependent on the hardware and characteristics of the work the system is doing. It needs to be determined by experimentation. A starting value of 2*number of physical processors is likely to work for a processor intensive system. If this value is too high throughput will decrease as contention in the form of context switching appears. If it is too low then the system will artificially be given less work to do than it is capable of performing. The right value is the value that provides the best throughput for a typical load. This value can be changed on a system in flight and the effects observed after a few minutes to stabilise.

Tuning: Dispatcher threads should be decreased if there is excessive context switching (10,000/second is regarded as high on our test equipment) and decreasing it improves throughput. It should be increased if the Perfmon counter Process Engine\Average wait for dispatcher thread is non zero and increasing the number improves throughput and decreases this counter. It is unlikely that a value less that the number of real (as opposed hyper threaded) processors will suffice.

7.2.15 Web Caching\Cache Options

XSL Caching

Enabled: Checked **Rationale**: Avoids the IO/build cost at little memory cost

Max Size: Calculated

Tuning: This should be based on the number of XSL transforms used by Custom Controls. A low value can adversely affect response times. There is little cost in setting a high value.

Custom Controls

Enabled: Checked **Rationale**: Avoids the IO/build cost at little memory cost

Max Size: Calculated

Tuning: This should be based on the number of Custom Controls configured. A low value can adversely affect response times.

7.3 Processor Affinity/Multi-instance

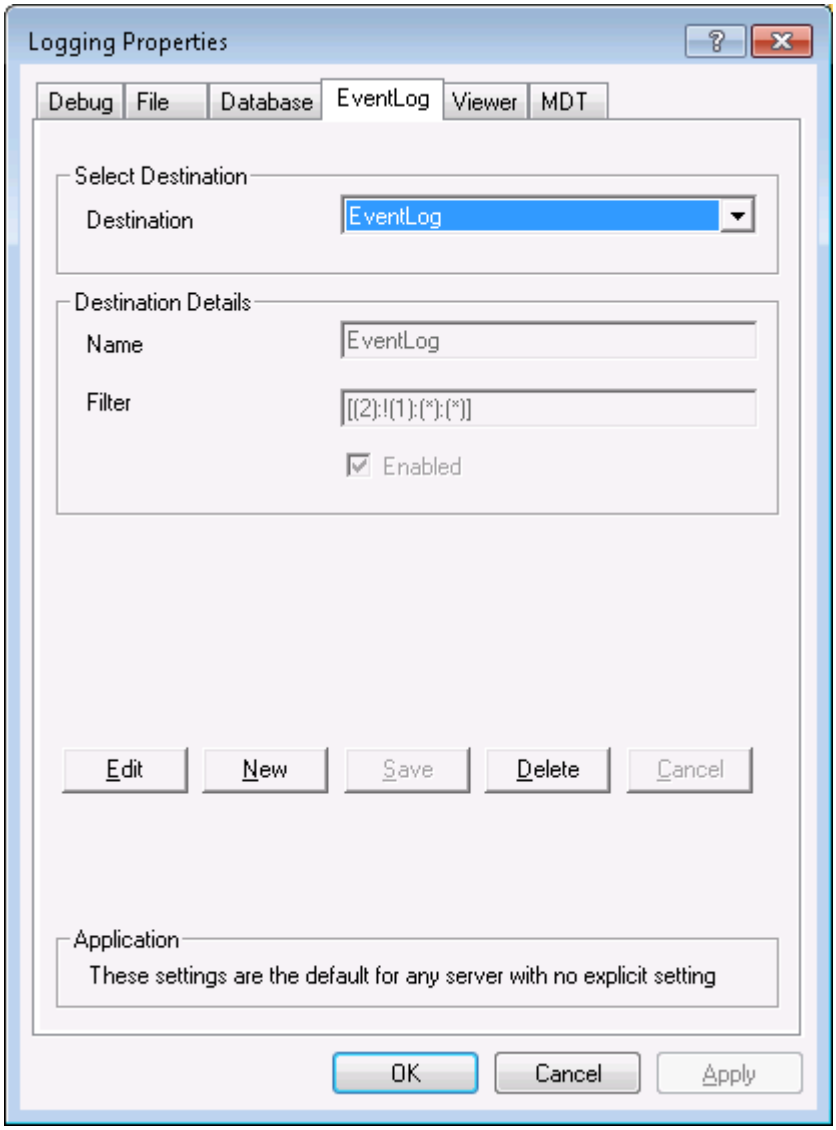
On devices with a large number of processors (typically above 4, e.g. ES7000) options are available to achieve a greater throughput than would ordinarily be achieved by a standard installation and setup. Information on these options is available on request.

7.4 Error Logging

The following settings are the only logging destinations that should be enabled. There should be a positive check that all other destinations are not enabled.

All logging destinations are configured using the Portrait Management Console. Right click on the **Logging** item under "All Servers→Default settings" and select "Properties".

Logging to Application Event Log



Select the "EventLog" tab of the "Logging Properties" dialog. The Filter value should be set to **[(2):!(1):(*):(*)]**. This setting is usually enabled by default.

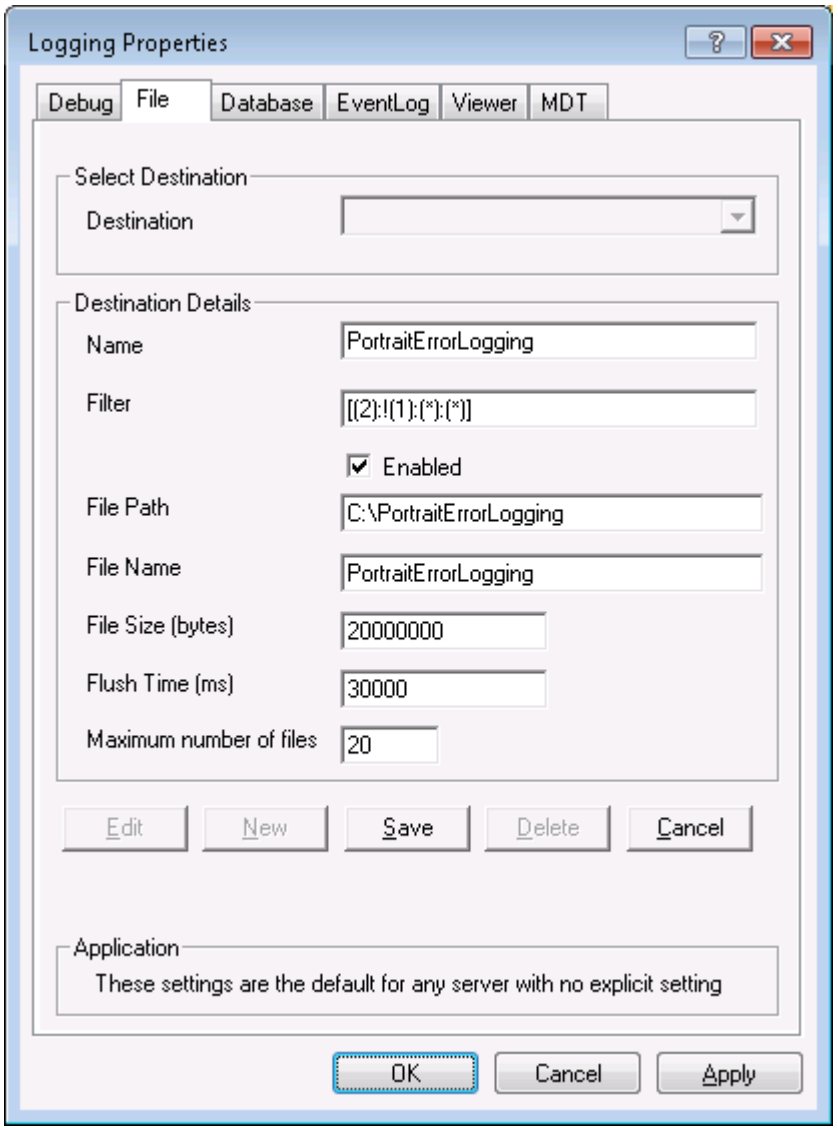
The PLX file suppression of spurious log messages should be used to avoid an unmanageable event log.

Logging to file

Error logging to file creates binary formatted Portrait log files (.plf) locally on each server. It is recommended that a specific directory is created for storing these files, and this document assumes that you create a directory called

```
C:\PortraitErrorLogging
```

It is important that the directory is created on all servers in the environment before the MMC changes are applied because logging will start immediately once the MMC changes are OK'd but will fail if the target directory does not exist.



Configure the "File" tab of the "Logging Properties" dialog as below by clicking on "New" to create a new destination, entering the values, then clicking "Save" followed by "OK".

Name should be **PortraitErrorLogging**

Filter should be **[(2):!(1):(*):(*)]**

Enabled should be checked

File Path should be **C:\PortraitErrorLogging**

File Name should be **PortraitErrorLogging**

File Size (bytes) should be **20000000** *(20 megabytes)*

Flush Time (ms) should be **30000** *(30 seconds)*

Maximum number of files should be **20**

A new log file called "PortraitErrorLogging.plf" will be created whenever the current one reaches 20 MB in size. Old log files are renamed to include the date & time that they reached 20 MB in size.