Windchill® Performance Tuning Guide

Windchill 7.0

December 2003

AutoDesk Inventor are registered trademarks of Autodesk, Inc. Baan is a registered trademark of Baan Company. CADAM and CATIA are registered trademarks of Dassault Systemes. COACH is a trademark of CADTRAIN, Inc. DOORS is a registered trademark of Telelogic AB. FLEXlm is a registered trademark of GLOBEtrotter Software, Inc. Geomagic is a registered trademark of Raindrop Geomagic, Inc. EVERSYNC, GROOVE, GROOVEFEST, GROOVE.NET, GROOVE NETWORKS, iGROOVE, PEERWARE, and the interlocking circles logo are trademarks of Groove Networks, Inc. Helix is a trademark of Microcadam, Inc. HOOPS is a trademark of Tech Soft America, Inc. HP-UX is a registered trademark and Tru64 is a trademark of the Hewlett-Packard Company. I-DEAS, Metaphase, Parasolid, SHERPA, Solid Edge, and Unigraphics are trademarks or registered trademarks of Electronic Data Systems Corporation (EDS). InstallShield is a registered trademark and service mark of InstallShield Software Corporation in the United States and/or other countries. Intel is a registered trademark of Intel Corporation. IRIX is a registered trademark of Silicon Graphics, Inc. MatrixOne is a trademark of MatrixOne, Inc. Mentor Graphics and Board Station are registered trademarks and 3D Design, AMPLE, and Design Manager are trademarks of Mentor Graphics Corporation. MEDUSA and STHENO are trademarks of CAD Schroer GmbH. Microsoft, Microsoft Project, Windows, the Windows logo, Windows NT, Visual Basic, and the Visual Basic logo are registered trademarks of Microsoft Corporation in the United States and/or other countries. Netscape and the Netscape N and Ship's Wheel logos are registered trademarks of Netscape Communications Corporation in the U.S. and other countries. Oracle is a registered trademark of Oracle Corporation. OrbixWeb is a registered trademark of IONA Technologies PLC. PDGS is a registered trademark of Ford Motor Company. RAND is a trademark of RAND Worldwide. Rational Rose is a registered trademark of Rational Software Corporation. RetrievalWare is a registered trademark of Convera Corporation. RosettaNet is a trademark and Partner Interface Process and PIP are registered trademarks of "RosettaNet," a nonprofit organization. SAP and R/3 are registered trademarks of SAP AG Germany. SolidWorks is a registered trademark of SolidWorks Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the United States and in other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. Sun, Sun Microsystems, the Sun logo, Solaris, UltraSPARC, Java and all Java based marks, and "The Network is the Computer" are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and in other countries. VisTools is a trademark of Visual Kinematics, Inc. (VKI). VisualCafé is a trademark of WebGain, Inc. WebEx is a trademark of WebEx Communications, Inc.

Licensed Third-Party Technology Information

Certain PTC software products contain licensed third-party technology: Rational Rose 2000E is copyrighted software of Rational Software Corporation. RetrievalWare is copyrighted software of Convera Corporation. VisualCafé is copyrighted software of WebGain, Inc. VisTools library is copyrighted software of Visual Kinematics, Inc. (VKI) containing confidential trade secret information belonging to VKI. HOOPS graphics system is a proprietary software product of, and is copyrighted by, Tech Soft America, Inc. G-POST is copyrighted software and a registered trademark of Intercim. VERICUT is copyrighted software and a registered trademark of CGTech. Pro/PLASTIC ADVISOR is powered by Moldflow technology. Moldflow is a registered trademark of Moldflow Corporation. The JPEG image output in the Pro/Web.Publish module is based in part on the work of the independent JPEG Group. DFORMD.DLL is copyrighted software from Compaq Computer Corporation and may not be distributed. METIS, developed by George Karypis and Vipin Kumar at the University of Minnesota, can be researched at http://www.cs.umn.edu/~karypis/metis. METIS is © 1997 Regents of the University of Minnesota. LightWork Libraries are copyrighted by LightWork Design 1990-2001. Visual Basic for Applications and Internet Explorer is copyrighted software of Microsoft Corporation. Adobe Acrobat Reader is copyrighted software of Adobe Systems. Parasolid © Electronic Data Systems (EDS). Windchill Info*Engine Server contains IBM XML Parser for Java Edition and the IBM Lotus XSL Edition. Pop-up calendar components Copyright © 1998 Netscape Communications Corporation. All Rights Reserved. TECHNOMATIX is copyrighted software and contains proprietary information of Technomatix Technologies Ltd. Apache Server, Tomcat, Xalan, and Xerces are technologies developed by, and are copyrighted software of, the Apache Software Foundation (http://www.apache.org/) - their use is subject to the terms and limitations at: http://www.apache.org/LICENSE.txt. UnZip (© 1990-2001 Info-ZIP, All Rights Reserved) is provided "AS IS" and WITHOUT WARRANTY OF ANY KIND. For the complete Info-ZIP license see

Parametric Technology Corporation, 140 Kendrick Street, Needham, MA 02494 USA

Contents

Change Record ix

About This Guide xi

Related Documents xi

Technical Support xi

Documentation for PTC Products xii

Comments xii

Documentation Conventions xii

Third-Party Products 1-xiii

Measuring Windchill Performance and Scalability 1-1

Performance Measurement 1-2

Collecting Measurements 1-2

Servlet Engine Metrics 1-7

Server Resource Utilization 1-7

Web Server Logs 1-8

Collecting Oracle Statistics 1-8

Heap Management (Garbage Collection) 1-8

Performance Testing Guidelines 2-1

Simulating Workload 2-2

Single User Testing 2-2

Multi-user Testing 2-3

Windchill Server Tuning Guidelines 3-1

Objectives 3-2

Managing System Memory 3-2

Balancing System Memory 3-2

HotSpot Server VM 3-3

Generational Garbage Collector 3-4

J2SE1.4.1 Garbage Collector 3-5

Priority Control (Server only) 3-6

Setting Cache Sizes 3-6

wt.pom.cachedStatementReuseLimit 3-7

Profiling Method Server Application Logic 3-8

Imposing a Result Set Size Limit 3-9
Wildcard Behavior 3-10
Reviewing the Persistent Object Manager Log 3-10
Using Anti-Virus Software 3-10
Debug Tracing 3-10
Throttling Concurrent HTTP Transactions 3-11
Windchill Configuration Options 4-1
Monolithic versus Multi-Tiered Hardware Configuration 4-2
Configuring a Single Method Server 4-2
Configuring a Background Server for Background Queues 4-3
Configuring multiple Background Method Servers for Background Queues 4-4
Multiple Method Servers 4-4
Load Balancing For Multiple Method Servers 4-5
Selecting a Server 4-5
Threshold Detection 4-6
Configuring Multiple Log Files 4-6
Windchill Adapter Performance Issues 4-7
Defining Multiple Instances of Windchill Adapter on a Server 4-7
Changing LDAP Properties for Load-Balancing Among Instances of Windchill Adapter 4-8
Server Cluster Configuration 4-12
Windchill Server Configuration (Cache Slaves) 4-13
Persistence Storage Server Configuration (Master Cache) 4-14
Troubleshooting Cluster Problems 4-18
Windchill Client Tuning Guidelines 5-1
Resource Contention 5-2
Browser Cache Settings 5-2
Using the Bootstrap Client 5-2
Netscape Browser on UNIX 5-3
RMI tunneling 5-3
wt.tools.javarmi.JavaRMIServlet 5-4
Known issue with Internet Explorer 5 with iPlanet and Jrun 3.0 5-5
Measuring Network latency 5-5
Measuring Network Traffic 5-6
Implementing a Customized User Interface 5-7
Setting HTML Page Expiration 5-8
Setting Page Expiration Time 5-8
Tuning Oracle 6-1
Improving Oracle Database Performance 6-2

Setting Instance-Based Parameters 6-3
Dynamic SGA Configuration 6-3
How to Approximate SGA Sizing 6-3
Init.ora Performance Improvement Features 6-6
Implementing Cost-Based Optimization 6-7
Tuning Oracle for Customized Applications 6-7
Identifying and Tuning SQL Statements 6-8
Identifying Internal Oracle Bottlenecks 6-15
Oracle Tools to Collect Database Statistics 6-17
Advantages of Statspack Over Bstat/Estat 6-18
Using Statspack Proactively 6-18
Oracle Version Compatibility Matrix report for the Preceding Tools 6-19

Tuning the Servlet Engine 7-1

Implementing a Throttle 7-2
Heap Size 7-4
iPlanet 6 7-4
Logging 7-4

Windchill Cache Mechanism 8-1

Caching 8-2
Implementing a Cache 8-2
Maintaining Cache Integrity 8-3
Using the Method Context Class 8-6
CacheManager 8-6
ProjectLink Caches 8-6

Tuning The Operating System 9-1

Tuning Hp/Ux 9-2
Configuring Oracle Datafiles with Raw Logical Volumes 9-3
The Solaris Kernel 9-3
Configuring Oracle datafiles with directio filesystem 9-4
Monitoring Operating System And System Resources 9-5
Tuning Windows NT 9-5

Troubleshooting 10-1

Performance Checklist 10-2
Disk 10-2
Network 10-2
CPU 10-4
Analyzing Full Thread Dumps 10-6
Monitoring Method Server and ServerManager Memory Consumption 10-7

Resolving TCP Name Problems 10-8

Workflow Performance Tuning 11-1

Queue Pooling 11-2

Dedicated Queues 11-3

Enabling Dedicated Queues 11-3

Combining Queue Pooling and Dedicated Queues 11-6

Checking the Status of Background Method Server Queues 11-10

wt.queue.execEntriesCount 11-10

Useful SQL Queries A-1

FILE Name: explain_display.sql A-2

FILE Name: explain_create.sql A-3

find_all_index_info.sql A-4

index_rebuild.sql A-5

Basic Oracle Administration B-1

Reference documents B-1

Understanding Space Use in Oracle B-1

General Oracle Tuning B-2

Example Commands B-3

Oracle Database Sizing for Windchill C-1

Information About PTC Setup C-2

Sizing the Database C-3

Example C-3

Windchill Network Sizing Guidelines D-1

Editing Windchill Properties Files E-1

About the windchill Command E-2

About the windchill shell E-4

About the xconfmanager Utility E-5

Formatting Property Value Guidelines E-6

Change Record

Table 1 Changes for Release 7.0

Change	Description
Chapter 4, Server Cluster Configuration	Updates have been made to server cluster configuration information.
Chapter 6, Tuning Oracle	Chapter has been updated with new information in many locations.
Appendix B, Basic Oracle Administration	Appendix has been updated with new information in many locations.
Appendix C, Oracle Database Sizing for Windchill	Appendix has been updated with new information in many locations.
Appendix E, Editing Windchill Properties Files	Appendix explains the method for editing Windchill property files.

About This Guide

The *Windchill Performance Tuning Guide* is designed to help improve the performance and scalability of the Windchill system.

- This guide assists you with performing the following types of work:
- Measuring performance and scalability
- Improving server performance
- Improving client performance
- Tuning the database and customized applications
- Troubleshooting

Related Documents

The following documents may be helpful as you use this guide:

- The *Windchill Installation and Configuration Guide*
- The *Windchill System Administrator's Guide*

Technical Support

Contact PTC Technical Support via the PTC Web site, phone, fax, or e-mail if you encounter problems using Windchill.

For complete details, refer to Contacting Technical Support in the *PTC Customer Service Guide* enclosed with your shipment. This guide can also be found under the Support Bulletins section of the PTC Web site at:

http://www.ptc.com/support/index.htm

The PTC Web site also provides a search facility that allows you to locate Technical Support technical documentation of particular interest. To access this page, use the following link:

http://www.ptc.com/support/support.htm

You must have a Service Contract Number (SCN) before you can receive technical support. If you do not have an SCN, contact PTC License Management using the instructions found in your *PTC Customer Service Guide* under Contacting License Management.

Documentation for PTC Products

PTC provides documentation in the following forms:

- Help topics
- PDF books

To view and print PDF books, you must have the Adobe Acrobat Reader installed.

All Windchill documentation is included on the CD for the application. In addition, books updated after release (for example, to support a hardware platform certification) are available from the Reference Documents section of the PTC Web site at the following URL:

http://www.ptc.com/cs/doc/reference/

Comments

PTC welcomes your suggestions and comments on its documentation—send comments to the following address:

documentation@ptc.com

Please include the name of the application and its release number with your comments. For online books, provide the book title.

Documentation Conventions

PTC documentation uses the following conventions:

Convention	Item	Example
Bold	Names of elements in the user interface such as buttons, menu paths, and dialog box titles. Required elements and keywords or characters in syntax formats.	Click **OK**. Select **File > Save**. **License File** dialog box **create_**<*tablename*>**.sql**
Italic	Variable and user-defined elements in syntax formats. Angle brackets (< and >) enclose individual elements.	**create_**<*tablename*>**.sql**
`Monospace`	Examples Messages	`JavaGen "wt.doc.*" F true` `Processing completed.`

Convention	Item	Example
"Quotation marks"	Strings	The string "UsrSCM" . . .
	The CAUTION symbol indicates potentially unsafe situations which may result in minor injury, machine damage or downtime, or corruption or loss of software or data.	When you add a value to an enumerated type (for example, by adding a role in the RolesRB.java resource file), removing that value can result in a serious runtime error. Do not remove a role unless you are certain there is no reference to it within the system.

Third-Party Products

Examples in this guide referencing third-party products are intended for demonstration purposes only. Some specific setting guidelines for various products are given resulting from our testing with Windchill. For additional information about third-party products, contact individual product vendors.

1

Measuring Windchill Performance and Scalability

There are two important aspects to measuring and tuning systems to maximize performance. The first is the perceived response time experienced by remote clients when using the system in their daily work. The second is the capacity of the system to process multiple simultaneous requests in a timely manner. In this document the term Scalability is used to describe the ability to extend the processing capacity of the system by adding resources.

Topic **Page**

Performance Measurement....1-2
Collecting Measurements....1-2
Heap Management (Garbage Collection)....1-8

Performance Measurement

Measuring performance changes in a complex system such as Windchill is difficult since there are many factors that influence it. It is common that tuning is performed in a series of cumulative steps, some of which may actually have negative effects on another part of the system. It is therefore very important that records be kept of configuration changes and workload levels. Due to fluctuations in workload and other factors, performance averages need to be generated over an extended period.

Performance can be measured from two viewpoints: end user and system administrator. The end user measures performance in terms of elapsed time (latency) required for an application to process his/her request. On the other hand the system manager is concerned with how many client requests can be satisfied over time, such as transactions per minute. The latter is known as throughput. Throughput and latency are affected by the computer system utilization. Utilization is a measure of the amount of time the computer resources are busy. It is normally expressed as a percentage.

Latency, throughput and utilization are related. It follows that as the number of simultaneous requests increases, there is more competition for resources (cpu, memory, disk, network, etc). For example, consider what happens when several users request a page at the same time. You might expect the utilization percentage to rise, the response times to be similar and the throughput to increase.

As utilization of one or more resources approaches 100%, runnable tasks are queued before being processed. As the ‘run queue’ grows, the amount of CPU time available to run user tasks decreases and the system becomes ‘saturated’. When saturated, the system will continue to process user requests, but response times rise and throughput drops. In extreme situations client browser requests will timeout after many seconds of waiting

Quantifying Windchill performance should therefore focus on client latency, throughput, and resource utilization.

Collecting Measurements

When measuring Windchill performance, it is necessary to configure and collect various metrics. Collecting the performance metrics allows you to measure the effects of changes made during the tuning process. There are several Windchill properties that control the verbosity and periodicity of metrics logging. Please refer to *`$WT_HOME/codebase/properties.html`* for details on available properties and the values that they could be set to. The following is a detailed description of some of these properties. See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

wt.method.serverMethodTiming

To enable a high level of verbosity in the Method Server set wt.method.serverMethodTiming true. The Method Server(s) must be restarted following the change.

The following extract illustrates the messages appearing in the MethodServer log as a result of a remote client request.

```
INVOKE: start
wt.clients.folderexplorer.FolderBusinessObject$LiteObjectProvider.
getWTBusinessObjects
Registering context RMI TCP Connection(3092)-132.253.8.130, host =
132.253.8.130
reading client authentication.
reading arguments.
Invoking target method.
Sending feedback for Thread[RMI TCP Connection(3092)-
132.253.8.130,10,RMI Runtime]
sending result.
Unregistering context RMI TCP Connection(3092)132.253.8.130
INVOKE: end
wt.clients.folderexplorer.FolderBusinessObject$LiteObjectProvider.
getWTBusinessObjects, 1312 ms (15/1281/16)
```

The extract shows the fully qualified method being invoked, the IP address of the remote client and the elapsed time spent processing the call. The figures in parentheses break down the total elapsed time into three distinct phases:

1. Time to unmarshal arguments received from the client.
2. Time to execute the method.
3. Time to marshal the results to the client. Note sometimes the writeExternal field of the object being returned to the client causes additional methods to be called. In this case the marshal time may be significantly longer than the ‘time to execute’.

wt.method.clientMethodTiming

To aid in debug and performance measurement of Windchill applets you can set the property wt.method.clientMethodTiming to true in wt.properties. This property can be changed without restarting the Method Server since the file is served by the web server when the browser initiates its first RMI connection with Windchill (the first time an applet is started).

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

When set true, this property causes every remote method call invoked by the client to be logged in the browser’s Java console. For example, this sample message was produced when opening a folder in the Windchill Explorer:

```
INVOKE: start
wt.clients.folderexplorer.FolderBusinessObject$LiteObjectProvider.
getWTBusinessObjects
sending client authentication.
sending target object.
waiting for response...
received context id.
received server feedback.
received method result.
INVOKE: end
wt.clients.folderexplorer.FolderBusinessObject$LiteObjectProvider.
getWTBusinessObjects, 1344 ms
```

The elapsed time (1344ms) includes both the ‘server busy’ time and network latency.

wt.method.access.log.enabled

The next level of verbosity causes logging of every remote method invocation (call) to a file. Various data including the name of the target method and the duration of the call is logged. The default value is false but can be set true in both production and benchmarking tests – albeit with additional I/O overhead.

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

The access log is written in a comma separated value (CSV) format to allow easy import into a spreadsheet for further analysis. In the wt.properties the property wt.method.access.log.file=$(wt.logs.dir)\\access.csv specifies the path to the output file.

The following extract shows two entries from a sample access.csv file:

```
Wed 7/14/99 17:26:42: , 130.21.55.229, 340, 0, 1,
wt.session.SessionManagerFwd.getPrincipal,
wt.method.AuthenticationException,
```

```
Wed 7/14/99 17:26:45: , 130.21.55.229, 150, 0, 1,
wt.httpgw.HTTPServer.processRequest, wt.httpgw.HTTPResponse,
/wt.httpgw.HTTPAuthentication/login,
```

Each entry comprises eight fields which are described below

Date

The timestamp generated when the call completed.

Client host

The IP address of the originating host. Many of the IP addresses will likely resolve to the hostname where the HTTPGatewayServlet is invoked. RMI calls from applets and remote applications are logged with the client IP address (or firewall IP address).

Time in milliseconds

The elapsed time in milliseconds for the call to complete. In order to measure the CPU time spent processing the call you need to run the Method Server in a profiler.

Redirect count

Client requests may be redirected between Method Servers as part of load balancing. The redirect count field indicates the number of times each call was redirected prior to execution. This count is always 0 when using a single Method Server or when load balancing has not been configured correctly.

Active context count

The active context count shows how many other active calls were in progress in the Method Server when the call completed. The count only applies to the Method Server where the call was processed. Note that the active context count includes calls being redirected.

Target class

The target class name shows the class name in which the requested target method was invoked.

Target Method

Together with the Target Class field, provides the fully qualified identity of the method invoked.

Result class

Shows the classname of the result object returned to the caller.

Detail

If the call is received from the HTTPGatewayServlet the detail field contains the query pairs from the requested URL.

To collate the data from the csv file, import it to a spreadsheet and define a pivot table. The table can be sorted in a number of ways, such as by number of times a method was called, the average duration, or the total duration. This information can be very useful for finding transactions to tune. Look for frequently executed or long running transactions.

wt.method.methodSummaryInterval

A call summary line is periodically written to the method server log file. The default interval for activity reporting is ten minutes. The summary shows the number of calls, the average duration of the call, and the maximum and average number of active calls:

```
MethodSummaryWriter: SUMMARY: 10.0 min, calls = 6 (0.6/min), av
dur=492 ms, mx/av act = 1/1.0
```

Summary Interval

The first field in the summary line shows the summary interval, e.g. 10 minutes. The summary line is only written when the method server has completed one or more client requests in the interval.

Calls

The calls field shows how many client requests were completed in the interval. A call is defined as a remote method invocation typically from the HTTPGateway or remote applet/application. Included in the call count field is also a throughput figure in terms of calls over time.

Average Duration

Average duration provides an indication of elapsed times experienced by clients (not inclusive of network latency and rendering). The average duration is expressed in milliseconds and is derived from the accumulated elapsed time for all calls completed in the interval over the number of calls completed. Average durations are influenced by transaction complexity, result size and system utilization. Primarily, the number and size of database accesses made by a transaction affect its duration. Transactions which make use of data cached in the Method Server or thread will generally complete much sooner than those that require database accesses.

Maximum/Average Active Calls

The average number of active calls shows how many other transactions were active on when each call completed. It is a useful measure of how many client requests were concurrently active during the interval. Server saturation occurs when this average active count is greater than the capacity of the server (maximum sustainable concurrency). Refer to section titled Multi-user Testing on page 2-3 for details on how to determine the capacity of the server.

wt.pom.statementCache.summaryInterval

Controls the frequency of the JDBC statement cache metrics which may be written to the MethodServer.log file. The size of the JDBC cache greatly influences Windchill performance due to the overhead of preparing versus reusing JDBC statements. It is a fixed sized cache using a least recently used algorithm when replacing entries due to overflow. The statement cache summary shows the hit/miss statistics and is useful in determining its efficiency. Ideally the ratio of hits to misses should be 90% or more.

The summaryInterval value specifies the number of cache lookups between each summary. The default is 0 which disables the metric. A value of 2000 is recommended for tuning activities. Refer to page 3-7 for more information.

Be aware that there is a separate statementCache for each of the pooled database connections in a Method Server. The cache objects are identified by a unique string resembling ‘wt.pom.StatementCache@3f5841’

StatementCache: wt.pom.StatementCache@3f5841[size=25, count=25, hits=3452, misses=67854, aged=65854]

Size

Shows the capacity of the cache. The statementCache size is configured by property wt.pom.statementCacheSize (default 25) in db.properties

Count

Shows the current number of entries in the cache.

Hits

Shows the number of times a cached entry was found in the cache

Misses

Shows the number of times an entry was not found in the cache.

Aged

Shows the number of times an entry was removed from the cache because of overflow. Overflow occurs when there is insufficient space available in the cache to store a new statement.

wt.principal.cache.summaryInterval

The principal cache is used to keep frequently accessed user and group information in the Method Servers process memory. It is a fixed sized cache using a least recently used algorithm when replacing entries due to overflow. Ideally the principal cache is sized for the peak number of active users. If the principal cache is sized too small there will be additional database and distributed cache hits.

The summaryInterval value specifies the number of cache lookups between each summary. The default is 0 which disables the metric. A value of 2000 is recommended for tuning activities.

Servlet Engine Metrics

Depending on the servlet engine installed, you may be able to collect additional performance metrics from the HTTPGateway tier. Refer to 7-4 for more information.

Server Resource Utilization

When measuring Windchill performance CPU, memory, disk and network utilization should be recorded. If the Oracle server is running on a separate server, collect resource utilization on that server as well. On Solaris and HPUX this can be done with the sar command:

```
Sar –o sar.log 60 60
```

This command will generate a log file (sar.log) summarizing resource utilization once a minute for an hour. To examine the sar.log file after the test enter sar –f sar.log.

On NT use the perfmon tool to collect resource utilization. Refer to page 9-6 Using performance monitor for more information.

Refer to page 9-1 for OS specific tuning tips.

Web Server Logs

The web server log files are useful for measuring the number of HTTP requests received. Entries in the file also log the authentication name of the remote client, the URL requested, the HTTP status code (see below), and the length of the response in bytes.

Refer to the following URL for the definitive list of HTTP status codes: http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html

There are many commercial web server analysis tools which can report useful performance metrics from the web server log files.

Collecting Oracle Statistics

You may wish to have an Oracle DBA examine statistics collected during the test. Oracle provides various analysis tools including the utlbstat.sql and utlestat.sql scripts in $ORACLE_HOME/rdbms/admin. Run the utlbstat.sql script before starting each test and utlestat.sql at the end. Utlestat.sql produces a report file which contains useful statistics for trained DBAs.

Refer to page 6-1 for information on tuning Oracle.

Heap Management (Garbage Collection)

Pauses due to Garbage Collection (GC) may be frequent if insufficient memory has been allocated for heap storage. However, it is difficult to diagnose unless the VM is configured to log GC activity. It is therefore recommended that you set the command line argument –verbose:gc on the Method Server startup command (wt.manager.cmd.MethodServer) in wt.properties.

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

In J2SE1.4, the Method Server startup command can take the argument -Xloggc to specify a log file where the "verbose:gc" information is logged, and to stop the display in the standard output. You set the argument to a log file name. Details about the GC activity, such as size of the young and old generation before and after GC, size of total heap, time it takes for a GC to happen in young and old generation, and size of objects promoted at every GC, and other data, can be displayed in addition by using the XX:+PrintGCDetails argument.

The VM will then report its GC activity and log the elapsed time for each pass. The trick is to configure the heap to minimize GC activity.

Refer to your Java vendor's web site for more details on tuning Garbage Collection. Also, see page 3-4.

Sun & Intel: http://java.sun.com/docs/hotspot/gc/index.html

HP: http://www.hp.com/products1/unix/java/infolibrary/prog_guide/java1_3/hotspot.html

2

Performance Testing Guidelines

This chapter suggests a methodology for measuring latency, throughput and utilization in a pre-production system. It describes how to simulate typical transactions and measure response times. This helps to isolate transactions which could limit system performance when under high client demand.

Having measured and tuned to minimize single user response times, we describe how to drive the system into a saturated state. Having determined the system's maximum throughput you can either add/upgrade resources or set processing limits to avoid system saturation.

Topic **Page**

Simulating Workload 2-2
Single User Testing 2-2
Multi-user Testing 2-3

Simulating Workload

When tuning the system prior to production the best results will be obtained by an accurate simulation. This involves predicting the number of users who will simultaneously execute queries and updates. An accurate simulation needs to reproduce the types of transaction that will occur in the production environment.

We can simulate ‘real world’ user transactions with two types of workload: fixed and managed. Fixed workloads consist of invariant transactions, for example, a list of URLs that represent a set of user requests (HTTP GETs) – these may be gathered from the web server access log for example. A useful tool for generating fixed workloads is Apache Jmeter. It can be downloaded from http://java.apache.org. Jmeter provides a useful feature whereby URLs and their elapsed times may be recorded in a log file. Since Jmeter is a Java application it can be run on a variety of architectures. You may also wish to evaluate Sun’s JavaStar tool which is available from http://java.sun.com.

Fixed RMI transactions need to be manually initiated – either through an applet or some custom Java application.

Managed workloads more accurately simulate transactions and allow you to parameterize and randomize the input data each time it is run. Managed workloads require the use of proprietary tools like Mercury LoadRunner or Segue Performer. Such tools require experience in the scripting language. A wider range of transactions is possible with these tools, including multipart post, and the ability to parse HTML responses. As with fixed workloads, RMI transactions need to be manually initiated. Neither Mercury nor Segue have successfully demonstrated the ability to record, script and playback Windchill specific RMI transactions.

Single User Testing

To establish a baseline of single-user response times, define a set of queries representative of those a ‘real’ user would make. Analysis of existing web server logs should provide frequently accessed URLs. For example, find URLs that invoke a template processor comprising a set of query pairs. Steer clear of queries that return abnormally large result sets. Once you have identified the list of URLs which you wish to measure, configure the test tool (for example, Jmeter) on a server with access to the Windchill server.

Try to define acceptable single user response times for the requests being tested. Limit other interactive users for the duration of the single user tests since you will be trying to measure the ideal or optimal response times for a set of queries. This should also help limit variance in the response times being measured.

Decide how long the single user test should run. Fifteen minutes to thirty minutes is usually enough, but try to be consistent for all tests. If the tool is capable of simulating ‘Think Time’ (a fixed or random delay between URL hits), select the minimum value allowed.

1. Start the server monitoring tools as described in section Collecting Measurements on page 1-2. Configure the test tool to save its results to a log file and start the test.

2. When the test is complete, gather all log files into a common folder/directory. The list of files to gather should include (as a minimum), Jmeter log file, access.csv, MethodServer.log, sar/perfmon.log.

3. Import the Jmeter log file into a spreadsheet such as Excel and create a pivot table report to summarize the number of times each URL was called along with the average, min/max/stddev elapsed times. Sort the pivot table on the average duration to determine which URL takes the longest to process.

4. Repeat the test with SQL tracing enabled and analyze the trace file with TKPROF as described on page 6-8. Use the tuning suggestions in Chapters 3, 4 and 5 to obtain the lowest average single user response times.

During the single user tests you might find a specific page that takes an excessive amount of time to build. Various human behavior studies have suggested that users are generally prepared to wait up to 8 seconds for an HTML page to appear in their browser. Depending on the complexity and size of the page being built this might be a reasonable maximum to aim for. If you find pages whose average response times are significantly slower than your target(s) then you will need to investigate further. Refer to the sections on page 3-8 for hints on how to find application bottlenecks and Tuning Oracle for Customized Applications on page 6-7.

Multi-user Testing

With the system tuned to minimize single user response times, the next step is to measure the effect of increased user concurrency on transaction latencies and system throughput. Initially the multi-user testing should be performed using a single Method Server.

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

1. Jmeter can be configured to simulate up to 20 concurrent users. To simulate larger user communities, multiple instances of Jmeter may be configured either on the same host or on separate hosts. Alternatively, edit the Jmeter properties file which may be inflated from the Jmeter jar file (org/apache/jmeter/jmeter.properties) increasing the value for property threads.max.

2. Reuse the single user scenario with a gradually increasing number of threads (concurrent users) in each test. For each test collect client latencies and server resource utilization as described above. It is suggested that 3, 6, 9, 12, etc. users be simulated. Gather average user latencies and plot them against user counts. Each time you run the test, ensure that it is allowed to run for the same duration. This will help to determine transaction throughput.

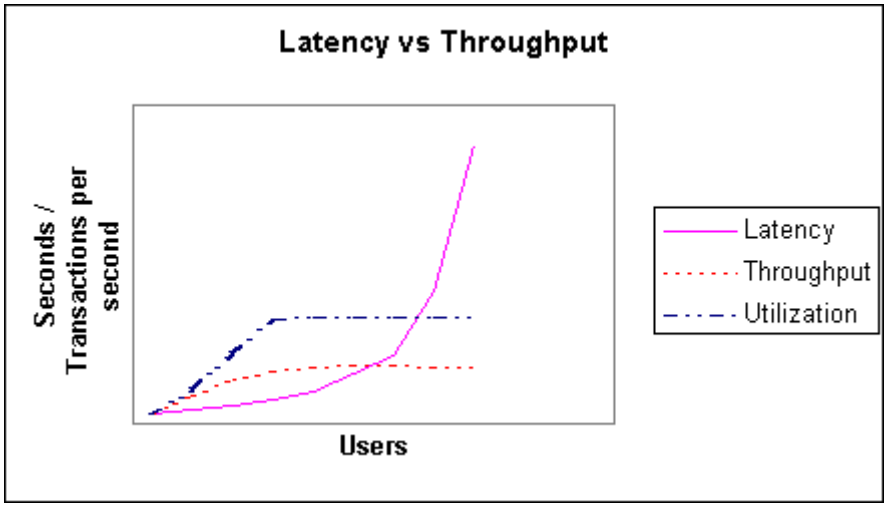
3. Examine the method server log file for each workload to check that concurrency (average active context count) is increasing. Some servlet engines limit concurrency if a license is missing or incorrectly installed. Refer to page 7-4 for information on logging in the servlet engine.

Using the access.csv file (see page 1-4), count the number of transactions completed in the test, taking care to analyze an identical length of time for each run. This can be made easier by importing the csv file into Excel and using the time stamps on each line to count the number of completed calls. Divide the total number of transactions completed by the number of seconds in the test to obtain a measure of throughput in terms of transactions per second.

One of the fields in the access.csv file notes the number of active calls. This indicates the level of transaction concurrency. Calculate the average number of active calls during the test.

Examine the server resource utilization percentages gathered from sar or perfmon. Find the most heavily utilized resource during the scenario, (e.g. CPU, disk, network, etc.) and calculate its average utilization. In the spreadsheet, plot the throughput value and utilization against active calls.

The graph below illustrates an example where latency, throughput and utilization are plotted against an increasing transaction concurrency with a fixed workload:



Actual values will vary from system to system. Factors such as number and speed of CPUs, JVM performance, and transaction complexity will yield different results.

In this example it is clear that throughput and utilization rise as the number of concurrent user transactions increase. Note how throughput levels off when utilization approaches 100%. Note also how response times rapidly deteriorate as a result when the system is saturated.

Determine and record the average active calls when at peak throughput since it indicates the maximum sustainable transaction concurrency. Any more workload will cause throughput to drop. Extended periods where workload is above this maximum may result in client timeouts.

Once you have recorded the capacity of the server to process multiple transactions concurrently you should use the guidelines in Chapter 2 and 4 to tune the single Method Server configuration. Making one change at time to the configuration or properties files, repeat the multi-user test and compare and document the results. If you measure an improvement in performance leave the corresponding change in place. Conversely, remove changes which cause performance regressions.

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

After having experimented with all the properties you should have an optimally tuned single Method Server configuration. If the server has multiple CPUs you should repeat the tests using multiple Method Servers as described in Chapter 3.

If you are configuring a clustered Windchill system, you should repeat the multi-user tests after following the procedure described in Chapter 3. If the server configuration for each node in the cluster is identical, the Windchill configuration files can be copied to each host after the initial host has been tuned.

In a production environment the Windchill administrator should monitor average active contexts during various times of the day. On systems where the user community is growing over time, it should be possible to extrapolate when a server will reach saturation using the upward trend in the average active contexts.

3

Windchill Server Tuning Guidelines

The preceding chapters suggested techniques for measuring performance and collecting metrics. This chapter focuses on tuning the Windchill server processes to maximize performance. These guidelines describe properties shared by all Method Servers and so apply equally to both single and multi Method Server configurations.

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

Chapter 4 describes how to configure multiple Method Servers and load balancing. Also provided in Chapter 4 are detailed instructions for configuring clustered Windchill servers. Chapter 5 contains some suggestions for improving client performance.

Topic **Page**

Objectives 3-2
Managing System Memory 3-2
Balancing System Memory 3-2
HotSpot Server VM 3-3
Generational Garbage Collector 3-4
Priority Control (Server only) 3-6
Setting Cache Sizes 3-6
Profiling Method Server Application Logic 3-8
Imposing a Result Set Size Limit 3-9
Wildcard Behavior 3-10
Reviewing the Persistent Object Manager Log 3-10
Using Anti-Virus Software 3-10
Debug Tracing 3-10
Throttling Concurrent HTTP Transactions 3-11

Objectives

Before starting to tune the system decide what objectives need to be met. Without setting clear objectives it will be hard to measure the success or failure of the effort.

If you are tuning to maximize transaction throughput you need first to optimize the entire system configuration. This includes activities such as adjusting the number of Method Servers, heap sizes, database connections, cache sizes etc. Follow the guidelines in Chapter 2 to test throughput with a high transaction concurrency.

If your goal is to improve response time for a particular transaction this chapter has some tips which apply to server tuning and some advice on profiling the application.

Alternatively, if you are troubleshooting a general performance problem then refer to the Performance Checklist on page 10-2.

Managing System Memory

An integral part of optimizing Windchill performance is proper memory management. The frequency of Java garbage collection (heap management) is greatly influenced by transaction load and available memory. Both the Windchill method server and the Oracle database server can efficiently cache persistent data in memory to minimize disk reads and thus increase transaction throughput. In addition, the Web server, Servlet engine and Operating system require significant amounts of memory. The trick is to balance all memory requirements with physical memory and avoid excessive virtual memory paging.

Balancing System Memory

If you combine applications on one physical system, it is important to balance the available memory across the system to avoid unnecessary disk paging. A sample memory allocation follows, demonstrating a system with 512 megabytes of RAM running Windchill, Oracle, Search Engine, Web Server, and the operating system.

Table 1 Sample Memory Allocation

Server	Maximum Memory setting	Percent of Available Memory
Windchill	160 Megabytes	31
Oracle	160 megabytes	31
Search Engine	32 megabytes	6.25
Operating system	64 megabytes	12.5

Server	Maximum Memory setting	Percent of Available Memory
Servlet Engine	32 megabytes	6.25
Web Server	64 megabytes	12.5

Similar percentages could be used on systems with higher available memory.

Where Oracle and Windchill are running on separate servers, the respective percentages should increase to around 60% of the available memory.

The Oracle sga size can be obtained with:

```
SVRMGR>connect internal
Connected.
SVRMGR> show sga
Total System Global Area      4438144 bytes
Fixed Size                    48260 bytes
Variable Size                 3973092 bytes
Database Buffers              409600 bytes
Redo Buffers                  8192 bytes
```

Setting Java Heap Size

It is important to configure the Java heap sizes for all Virtual Machines based on the Servers available memory. The heap sizes are specified as an argument on the java command.

Maximum heap size regulates the total amount of memory that will be used by a VM. The maximum heap size should be increased to balance available memory across the system. Maximum heap size is set with the –Xmx argument. The default initial heap size of 32 megabytes should be increased to 50 to 100 percent of the maximum heap size. The initial heap size is set with –Xms argument.

Table 2 Sample command to start a method server:

```
wt.manager.cmd.MethodServer=cmd.exe /C start "MethodServer" /MIN
"$(wt.java.cmd)" –server -classpath "$(wt.java.classpath)"
–Xms128m –Xmx256m -Xnoclassgc  wt.method.MethodServerMain
```

In the above example, the initial heap size is 128 megabytes, 50 percent of the maximum heap size.

Note: Where multiple method servers are configured, the total amount of memory consumed will be n*mx (where n is the number of method servers and mx is the maximum heap size).

HotSpot Server VM

The HotSpot Server compiler is tuned for the performance profile of typical server applications. It is a fully optimizing compiler. The optimizer performs all the classic optimizations, including dead code elimination, loop invariant hoisting,

common subexpression elimination, and constant propagation. It also features optimizations more specific to Java technology, such as null-check and range-check elimination. The server compiler performs full inlining and full deoptimization.

The default HotSpot compiler in most VMs is set up for client side applets/applications so some of the optimizations specific to long running applications are disabled. To take advantage of the Server compiler ensure that the –server flag is added as the first argument of the java command to launch the Method Server in wt.properties.

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

Generational Garbage Collector

Since the introduction of Java HotSpot, virtual machines have adopted a generational approach to garbage collection which allows short and long lived objects to be managed separately. The vast majority of Windchill objects are short lived, (they are garbage collected once the transaction completes) so it is possible to tune the VM accordingly.

If you are tuning to minimize single user response times you might try adding the –Xincgc flag (if supported) to the method server invocation. This can help ensure that adequate heap is always available. However, be aware that the incremental garbage collection can seriously affect throughput.

To tune garbage collection for optimal transaction throughput it may be possible to configure the size of the ‘eden’ or young generation space. A larger eden has been shown to increase throughput in benchmarks. Tuning the memory management of the HotSpot VM is a complex task. HotSpot learns over time, and adjusts its behavior to get better performance for your specific application. This is an excellent feature, but it also makes it more difficult to evaluate the output from the verbosegc flag. To gain a real understanding of HotSpot's interactions with Windchill, you need to run tests that simulate the production system, and run them for long periods of time.

Unfortunately each Java vendor has implemented different techniques for configuring the eden heap size. Refer to your Java vendor’s web site for more information:

Sun & Intel: http://java.sun.com/docs/hotspot/gc/index.html

HP:
http://www.hp.com/products1/unix/java/infolibrary/prog_guide/java1_3/hotspot.html

The next two properties specify how often the distributed garbage collection is performed. The example below changes the default value for both properties (60000 ms) to one hour. More information is available at http://java.sun.com/j2se/1.3/docs/guide/rmi/sunrmiproperties.html/

```
Sun.rmi.dgc.server.gcinterval=3600000
Sun.rmi.dgc.client.gcinterval=3600000
```

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

J2SE1.4.1 Garbage Collector

Two new garbage collectors, the Parallel Collector and the Concurrent mark-sweep (CMS) collector were introduced in J2SE1.4.1.

Parallel Collector

The parallel collector is implemented in the young generation. It enables garbage collection to occur on multiple threads for better performance on multiprocessor machines. Even though it suspends all mutators (application threads), it is able to complete the given amount of garbage collection work much more quickly by leveraging all available CPUs on the system. This reduces the GC pauses in the young generation significantly.

The flag -XX:+UseParNewGC turns on parallel garbage collection in the young generation. It can be enabled together with the CMS collector in the old generation. This setting could be tried with the ServerManager invocation. The flag -XX:+UseParallelGC also turns on parallel garbage collection in the young generation. However, it does not work with the CMS collector in the old generation. This is more suitable for JVM using larger heap and could be tried with the MethodServer invocations. The flag -XX:ParallelGCThreads=n sets the number of parallel GC threads that the JVM must run for performing garbage collection in the young generation. The default value of n is equal to the number of CPUs on the system. The number of threads should be set to a minimum of two, if the Parallel Garbage Collectors are used on a single CPU machine.

Concurrent Mark-Sweep (CMS) Collector

On the other hand, CMS is implemented in the old generation. It trades the utilization of processing power that would otherwise be available to the application for shorter garbage collection pause times. This collector is particularly very useful when the Full GCs are occurring very frequently, and the amount of memory cleaned is comparatively smaller.

The -XX:+UseConcMarkSweepGC flag turns on concurrent garbage collection in the old generation. This setting could be useful during invocation of the ServerManager. The -XX:CMSInitiatingOccupancyFraction=x flag sets the threshold percentage of the used heap in the old generation at which the CMS collection takes place. By default, this threshold is calculated at run time, and the CMS collector might be triggered only when the old generation heap is about 80 - 90% full.

Priority Control (Server only)

Depending on the shell and operating system in use when a Method Server is launched, the scheduling priority of the resultant process may be lowered. This is due to the process ‘nice’ value being automatically set to give preference to interactive processes. Since the servlet engine and Method Servers are typically started in sub-shells, the operating system may treat them as ‘batch’ jobs and thus lower their realtime scheduling priority. As a result other, unrelated, interactive applications may be given preference over Windchill by the scheduler. Refer to the man page for the ‘ps’ and ‘renice’ commands to determine if the ‘nice’ value of the Java processes (servlet engine, Method Servers) should be adjusted down to match other interactive processes.

On NT ensure the Application Performance Boost for foreground applications is set to 0 (System Properties->Performance->Application Performance)

Setting Cache Sizes

The various Windchill data caches should be sized large enough to avoid repeated RDBMS queries. Some of the caches support periodic summaries which show the cumulative number of hits and misses on cache lookup. A high ratio of misses to hits generally means the cache is sized too small. Refer to section on page for more information.

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

The relevant properties in the wt.properties file include the following:

- wt.cache.size.AclCache – The number of access control lists that can be cached in the servers. The default is 200. Make sure the ACL cache size is greater than the number of administrative domains. The number of administrative domains can be found with the following SQL statement:

  ```
  sql> select count(*) from administrativedomain;
  ```

- wt.cache.size.IndexListCache – The number of index lists that can be cached in the servers. The default is 200. The number of index lists in the database can be found with the following SQL statement:

  ```
  sql> select count(*) from indexpolicylist;
  ```

- wt.cache.size.NotificationListCache – The number of notification lists that can be cached in the servers. The default is 200. The number of notification lists in the database can be found with the following SQL statement:

  ```
  sql> select count(*) from notificationlist;
  ```

- wt.cache.size.SessionCache – The number of client sessions for which authentication information is cached in the servers. The default is 500. The SessionCache should be sized for the expected peak active user count.

- wt.cache.size.WTPrincipalCache – The number of principals (users and groups) that can be cached in the servers. The default is 200. The principal cache should be sized for the expected peak active user count. The principal cache supports periodic logging.
- wt.cache.size.ProjectCache – The number of Project objects that can be cached in the servers. The default is 200. The number of projects in the database can be found with the following SQL statement:

  ```
  sql> select count(*) from project;
  ```

- wt.cache.size.ReferenceCache – The number of cached object references. This cache is shared by various data types. To monitor the cache efficiency use the following utility:

  ```
  java wt.fc.cache.showCacheSte
  ```

  Inform:StdObjRefSvc: showCacheStats (wt.util.Cache@fdb73[size=200, count=200, hits=3245, misses=200, aged=0])

- wt.cache.size.PagingSessionCache – The number of open ‘paged’ database queries. The size should be based on the expected peak active users (as per wt.cache.size.SessionCache)

JDBC Statement Cache PropertiesThe JDBC Statement cache is a fixed size collection of reusable prepared JDBC statements. Before executing most SQL statements, Windchill checks to see if the statement already exists in the JDBC cache. Since significant overhead is required to create JDBC statements, the efficiency of this cache has a major influence on system performance. The JDBC statement cache may be monitored with negligible overhead — see page 1-6.

wt.pom.statementCacheSize

wt.pom.statementCacheSize specifies how many cached statements may be held by each database connection. The default is 25. Try to size the cache to minimize the number of misses reported by the periodic summary. Measure the effect of increased statementCache size on CPU utilization and heap usage.

wt.pom.cachedStatementReuseLimit

In addition to the StatementCache size there is another parameter which controls the number of times each cached statement may be executed before being removed from the cache. This property was added to work around potential memory leaks in the OCI driver. However, since the OCI driver is no longer used, it is safe to increase this reuse limit to a much higher value.

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

For example, in db.properties specify:

```
wt.pom.cachedStatementReuseLimit=32000
```

If you want to trace individual accesses to the JDBC statement cache you can set property wt.pom.log.statementCaching=true in db.properties. However, this will cause a large number of messages to be written the MethodServer.log file.

wt.pom.maxDbConnections

This property controls the number of database connections each Method Server holds open. The default value is 5. Having multiple connections open to the database increases throughput since there may be multiple concurrent JDBC statements executing in parallel (in separate threads).

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

When transaction concurrency is high (i.e. above maxDbConnections), threads will block waiting for database connections to be freed. Having multiple threads in this state can be a drain on both CPU and heap space.

If the number of active transactions is consistently above the maxDbConnection parameter, you should consider one of the following remedies:

1. Increase maxDbConnections
2. Increase the number of Method Servers
3. Impose more restrictive throttle in the servlet engine.

Note that after increasing either the number of database connections or Method Servers, you may need to re-evaluate the load balancing parameters (if you have multiple Method Servers).

Profiling Method Server Application Logic

Various 3rd party profiling tools are available to help highlight costly Java bytecode and isolate areas of the application logic which might benefit from software tuning. Examples of profiling tools are shown below:

NT	OptimizeIt!, Jprobe
Solaris	OptimizeIt!, Java Workshop
HPUX	HPJMeter

Profiling should not be performed on a production system. Profile customized services and applets during the development phase of a project only. You will need access to the source code in order to compile and test any changes.

When profiling a particular transaction make sure that the VM has pre-loaded all classes, modeled class information and cached data by repeating the transaction a number of times first. Capture the profile a couple of times to be sure it is consistent.

Things to look out for while profiling:

- Duplicate or unnecessary database queries. Perhaps the data being queried is already cached or has already been read. If possible use the MethodContext Hashtable to store the results of queries. Look for additional candidates for caching. Detailed information on implementing a Windchill data cache starts on page.
- Unnecessary property lookup. When reading from properties files, initialize values in static initialization rather than during runtime. This way the property value is assigned during class loading and can help the optimizing compiler produce more efficient byte code.
- Heavyweight client-side logic. Ideally, the client should make a single remote method call to perform a transaction. Every remote method call incurs a severe serialization overhead. Implement a server-side API or service that does the 'heavy-lifting' with a single call.
- Unnecessary serialization. If you don't need all the attributes returned from a query use a ClassAttributeQuery.
- Use the appropriate API. You might find overloaded methods that accept or return collections of objects as well as single objects. When dealing with multiple objects it is usually more efficient to use the corresponding API. Read the Windchill JavaDoc.
- Use threads wisely. When analyzing the logic flow consider if the task could be performed by more than one thread. However, extra work may be required to ensure transactional integrity between threads. When checking out a large number of objects use the threadedCheckout API in the StandardMultiWorkInProgressService.
- Measure elapsed CPU time spent in method calls. It is possible the bottleneck is caused by a blocking system call.

Imposing a Result Set Size Limit

Windchill performs minor validation of user queries prior to execution. As a consequence, queries may be constructed which return many thousands of rows from the RDBMS. Such queries can cause the executing thread to consume huge amounts of memory and CPU time. One symptom of this is when the system the system appears to be very sluggish The Method Server may also restart due to an OutOfMemoryException.

wt.pom.queryLimit

This property can be added to db.properties. It limits the number of rows in every result set to work around 'run away' queries. Set the value to some arbitrarily high value that is large enough to satisfy all realistic result set sizes (e.g. 20,000). By default its value is 0 or unlimited. Once the limit is exceeded Windchill returns a truncated result set to the caller (wrapped in an Exception).

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

Wildcard Behavior

Certain characters are treated as wildcards by Windchill and SQL. Characters '*' and '%' are used as multi-character wild cards. The single character wildcard is '_' (underscore). Customer data names or numbers using any of the wildcard characters may cause the Local Search processor to return more data than desired. To disable the single wild card character altogether set wt.query.singleWildCard=e

Reviewing the Persistent Object Manager Log

You can record the SQL statements issued by each transaction by setting the property wt.pom.log.SQLStatements=true in the db.properties file. For ease of use, set wt.pom.log.enabled=false and wt.method.serverMethodTiming=true in wt.properties. Consequently, the SQL statements will be written to the Method Server log file along with the invoked method that emitted them. These commands can be analyzed to determine which Oracle tables are being accessed. These properties should not be enabled in production environments, because they can significantly degrade performance.

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

Using Anti-Virus Software

Anti-virus software can be the cause of an unexpected performance drain. Many anti-virus products support scanning on both reads and writes of files. Server performance will suffer from scanning reads of Windchill codebase files. Client browser performance will suffer from scanning of browser cache files.

Most virus scanning products allow you to configure files or directories to be excluded from read or write scanning. On the server, the Windchill codebase should be excluded from read scanning, and the Windchill log file directory should be excluded from write scanning. On the client, the Web browser cache directory should be excluded from both read and write scanning.

When Norton AntiVirus is enabled, the performance of Netscape browser class loading is impacted, even when the class path directories are excluded. Netscape generates many file open attempts in the directories in its Java class path, and the number of file system accesses appears to quadruple when Norton AntiVirus is enabled. The same effect is not observed with Symantec VirusScan.

Debug Tracing

Debug tracing is a good troubleshooting tool, but should not be left on in a production environment. Debug tracing has a negative impact on both CPU usage

and I/O, because each line is flushed to the trace file to allow multiple server output to the same file. By default, debug tracing is turned off. The only output in the log files should be startup and shutdown messages and exception tracebacks. Refer to the properties.html file in the codebase directory for more information on which properties affect debug tracing.

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

Throttling Concurrent HTTP Transactions

One way to prevent server saturation is to limit the number of simultaneous HTTP requests passed to the method server(s). Servlet engines, such as JRun and Tomcat generally provide a mechanism for limiting the number of simultaneous requests by throttling the number of active threads. Every time an HTTP request is received by the HTTPGateway, a separate thread is assigned to process it via an RMI call into the Windchill Method Server. Therefore, every HTTP GET/POST request from client browsers causes a separate thread to be processed in the servlet engine.

We can effectively limit the number of simultaneous calls into the Method Server by throttling the number of active threads in the servlet engine. The servlet engine maintains a FIFO queue of requests (threads) to ensure that all transactions are handled in the order they are received. The length of the queue may also be tunable and can be set to limit the number of queued requests. Once the queue overflows, additional requests are rejected (until the queue returns below its ‘high water mark’) and the user is informed that the server is too busy.

For configuration details on throttling HTTP requests and other servlet engine tuning tips, refer to Implementing a Throttle on page 7-2.

Note: simultaneous RMI requests, e.g. from applets or other client applications cannot be throttled at present.

4

Windchill Configuration Options

This chapter contains information and recommendations about different configuration options for Windchill.

Topic **Page**

Monolithic versus Multi-Tiered Hardware Configuration 4-2
Configuring a Single Method Server 4-2
Configuring a Background Server for Background Queues 4-3
Load Balancing For Multiple Method Servers 4-5
Windchill Adapter Performance Issues 4-7
Defining Multiple Instances of Windchill Adapter on a Server 4-7
Changing LDAP Properties for Load-Balancing Among Instances of Windchill Adapter 4-8
Server Cluster Configuration 4-12

Monolithic versus Multi-Tiered Hardware Configuration

You have a choice of two configuration schemes when implementing the Windchill three-tier architecture.

The first approach is to run all of the Windchill and related processes (web server, servlet engine and Oracle server) on the same host. This is sometimes called a 'monolithic' configuration. In this case, the client browsers are typically run from one or more remote UNIX or NT workstations. This approach simplifies server administration and decreases the cost of system hardware.

The second approach is to run the Oracle server on a second system. This distributed hardware approach may provide the user with increased flexibility and reliability. Depending on the hardware resources available to you and the typical user workload, there may be performance advantages as well.

Note: Sample multi-user workloads have indicated up to a 30% increase in performance when the Oracle database is moved onto a second server. The relative improvement is most noticeable when there are insufficient processing resources on a single server to host both Windchill and Oracle. For single CPU Windchill servers, PTC recommends that the Oracle database be moved to a second server.

Server hardware is available in Uniprocessor or Multiprocessor configurations. The decision on whether to deploy Windchill and Oracle on a single versus multiple servers should be based on the expected transaction load and server capacity.

Using a second system for the database also allows for greater flexibility in tuning. The database host can be tuned for faster Oracle performance while the Windchill host can be tuned for network application performance. When using a separate database server, it is important to have high performance communication between the Windchill server and the database host to avoid a slowdown. If you use this setup, PTC recommends using a dedicated network interface between the two hosts.

Configuring a Single Method Server

Method server startup and monitoring is provided by the `StandardServerMonitor` class. The following, default *wt.properties* configuration will start up a single method server:

```
#Classes to be instantiated in the ServerManager at startup

wt.manager.loadObjects=wt.manager.StandardServerMonitor

#Services to be monitored by the StandardServerMonitor

wt.manager.monitor.services=MethodServer

#Number of servers to start

wt.manager.monitor.start.MethodServer=1
```

On single CPU servers, only one Method Server is recommended. With a single Method Server, all client requests and background queue processing will be performed by the same Java Virtual Machine. Use the default value (true) for property wt.queue.executeQueue.

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

Configuring a Background Server for Background Queues

The Windchill property settings described below allow you to configure a separate method server dedicated to running background queues. (See the chapter Configuring and Administering Background Queues in the *System Administrator's Guide for Windchill technology* for additional information about these queues and about background processing.)

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

For best performance, a separate Background Method Server should be run on a multi-CPU (SMP) architecture. Additional Method Servers may also be configured to share processing the interactive client requests.

Background Method Server performance is greatly affected by the complexity and number of Workflow processes. Refer to Chapter 11, for specific recommendations for improving Workflow performance.

The following wt.properties configuration will start up two method servers, dedicating one to running background queues:

```
#Background MethodServer command

wt.manager.cmd.BackgroundMethodServer=\
$(wt.manager.cmd.MethodServer)\

wt.method.serviceName=BackgroundMethodServer\

wt.queue.executeQueues=true

#Classes to be instantiated in the ServerManager at startup

wt.manager.loadObjects=wt.manager.StandardServerMonitor

#Services to be monitored by the StandardServerMonitor

wt.manager.monitor.services=MethodServer \ BackgroundMethodServer

#Number of Servers to start

wt.manager.monitor.start.MethodServer=1

wt.manager.monitor.start.BackgroundMethodServer=1

#Queue default execute setting

wt.queue.executeQueues=false
```

Configuring multiple Background Method Servers for Background Queues

Background queues can also be configured to execute on multiple background method servers. This is beneficial when the queue load is heavy and spare system resources (CPU and memory) are available. In such situations, multiple background method servers usually provide a performance boost, if the background queues load is shared between them.

If the queue load is heavy and spare system resources are available, you should consider starting an additional background method server and configuring half the process queues to execute on each of the two background method servers.

Each background method server instance is identified by a unique queuegroup whose value is set using wt.queue.queueGroup in the server's startup command. The queuegroup value is 'default' for the first background method server instance.

To configure a queue to execute on a second background method server, simply change the value of the queue's group to the queuegroup value of the second background method server, using Windchill QueueManager.

The following wt.properties configuration must be additionally set to start a second background method server and assign a queuegroup value of group1.

```
#Second Background MethodServer command

wt.manager.cmd.BackgroundMethodServer1=\
$(wt.manager.cmd.MethodServer)\

wt.method.serviceName=BackgroundMethodServer1\

wt.queue.executeQueues=true\

#queueGroup value must be unique

wt.queue.queueGroup=group1

#Services to be monitored by the StandardServerMonitor

wt.manager.monitor.services=MethodServer \ BackgroundMethodServer \
BackgroundMethodServer1

#Number of Servers to start

wt.manager.monitor.start.BackgroundMethodServer1=1
```

Multiple Method Servers

The optimal number of method servers is affected by the number of CPUs, location of the RDBMS, search engine and scalability of the architecture’s Java Virtual Machine. Generally, it is suggested that one Method Server for every 2 or 3 CPUs be configured. However, if Oracle and all Windchill applications are running on the same server, it is best to leave at least one CPU free to execute other applications.

The optimal number of Method Servers has to be determined experimentally. One of the objectives of this guide is to help gather enough data to make that judgement. Additional Method Servers may be started with the property wt.manager.monitor.start.MethodServer

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

Load Balancing For Multiple Method Servers

Multiple method servers may be started on a single host to distribute and balance load across multiple operating system processes. This may be beneficial if you are running a multiprocessor system and the transaction concurrency cannot be handled by a single Method Server.

There are numerous reasons to configure multiple Method Servers. These include the scalability of the Virtual Machine's use of native threads and the application design. As VMs improve in scalability, thread contention within the application becomes more of a bottleneck and so limiting the number of running threads in each VM can improve throughput.

In a multiple method server environment, the default setup performs round robin balancing on initial HTTP and RMI client connections. Subsequently an attempt is made to associate HTTP based clients with the same method server.

Load balancing takes the balancing idea one step further to the actual method call. This gives the method server the opportunity to switch servers on clients at the level of individual method calls if server load is excessive. Code changes are not required to use the load-balancing capabilities within Windchill. Load-balancing behavior is controlled by properties in the wt.properties file.

Selecting a Server

Selection of the next available server is performed by classes implementing the wt.manager.ServerSelector interface. Windchill provides two server selectors, StandardServerSelector and BalancedServerSelector. Setting wt.manager.serverSelector.server name specifies the server selector to be used (for example, wt.manager.serverSelector.MethodServer = wt.manager.BalancedServerSelector). The getServer(String service) and getNextServer(String service, Remote server) methods in the server selector interface deal with load balancing. getServer( ) returns the server which the client will interact with upon initial connection. getNextServer( ) returns the failover method server which the client will switch to when the current server surpasses a threshold.

Server Selector Description

wt.manager.StandardServerSelector On a getServer() returns a server in a round-robin fashion. On a getNextServer() returns the next server and wraps to the first server if called from the last server. This is the default selector.

wt.manager.BalancedServerSelector On a getServer() returns the first server. On a getNextServer, returns the least recently used server.

Threshold Detection

When a request is made on a method server, the current server checks if any thresholds have been surpassed. If so, a wt.method.ServerLoadException is thrown from this server and is caught by the remote method server (e.g. client or HTTPGW). Within the exception is a reference to the next server; the remote method server then redirects the request to that server. The wt.method.loadbalance.maxRedirects property specifies the maximum number of times a single method call will be redirected. The default setting is 1. A setting of 0 causes method calls to be redirected until a server that falls below the threshold has been identified. The threshold descriptions that follow are checked when load balancing is used.

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

wt.method.loadbalance.activeContext

This property defines the maximum number of currently active contexts allowed within a Method Server before a ServerLoadException is thrown.A separate context is normally created for each request received from the HTTPGW.

Configuring Multiple Log Files

To measure the distribution of transactions across all Method Servers it is useful to configure separate log files for each. A utility class is available which can parse the launch command and substitute numeric identifiers for tokens. To configure the launch utility set property:

```
wt.manager.cmd.MethodServerLauncher =
wt.method.MethodServerLauncher
```

The server launcher will delegate Method Server launching to the utility class. To configure each method server to write to separate log files, edit the `wt.manager.cmd.MethodServer` property as follows:

```
wt.manager.cmd.MethodServer=\
cmd.exe /C start "MethodServer {0}" \
"$(wt.java.cmd)" -classpath "$(wt.java.classpath)" \
-Xms32m Xmx64m -Xnoclassgc wt.method.MethodServerMain \
wt.method.access.log.file=$(wt.logs.dir)\\access{0}.csv\
wt.method.log.file=$(wt.logs.dir)\\MethodServer{0}.log
```

Each method server will run in a separate console window with a numeric identifier appearing in the title bar. The same numeric identifier will be appended to both the method server's log and access.csv files. The Unix equivalent can be achieved by launching each Method Server from a script, for example, launch_ms.ksh:

```
wt.manager.cmd.MethodServer=/opt/ptc/Windchill/launch_ms.ksh {0}
```

Where the script launch_ms.ksh is simply:

```
#!/bin/ksh

/usr/java/bin/java -Xms32m -Xmx64m -Xnoclassgc -noverify
wt.method.MethodServerMain \
wt.method.access.log.file=/opt/ptc/Windchill/logs/access$1.log \
wt.method.log.file=/opt/ptc/Windchill/logs/MethodServer$1.log \ >
/opt/ptc/Windchill/logs/heap$1.log 2>&1
```

Windchill Adapter Performance Issues

Windchill adapter is a tightly integrated component of the Windchill server. It provides the API through which Info*Engine accesses data in the Windchill system. Please refer to the *Windchill Adapter Guide* for more details about Windchill adapter.

When a method server is started, an instance of Windchill adapter is started automatically as a part of the method server startup sequence (this is true only for foreground method servers; background method servers don't startup Windchill adapter by default). However, there is only once instance of Windchill adapter defined in LDAP by default. This causes the following problem if more than one method servers are defined:

When the first method server starts up, it creates an instance of Windchill adapter which binds to a default address. However, when any method server that starts next tries to create a Windchill adapter, this adapter tries to bind to the same default port on which the Windchill adapter of the first method server was bound. Since this port is not available, a "java.net.BindException: Address already in use" exception is thrown.

So all the Windchill adapter requests will be served by the one method server with the Windchill adapter running inprocess even though several method servers might be up and running. In heavy load scenarios, this could lead to performance degradation as a single Windchill adapter might not be able to keep up with the load.

In order to circumvent this problem, there should be one instance of Windchill adapter defined for each method server (by method server here, we mean foreground method server only; since background method servers don't startup a Windchill adapter they don't need an instance of Windchill adapter in LDAP). Below, we describe how this can be done.

Defining Multiple Instances of Windchill Adapter on a Server

Below we describe the methodology for defining one Windchill adapter. For each method server, you should similarly define a Windchill adapter.

1. Go to Windchill homepage and click on site map

2. Click on Info*Engine Administrator under System Administration
3. Log in with your ldap id and password.
4. In the pull down menu Create Entry, select Windchill Adapter. This will open up a property editor for windchill adapter instance you are trying to define.
5. Enter the following parameters in the form:
   - Service Name--ex. windchillAdapter2
   - Runtime Service Name-- ex. com.ptc.mn.nansen.Windchill
   - Host--ex. nansen.mn.ptc.com
   - Port--ex. 18002
   - Logging Directory--ex. ~/windchill60/logs

Note: To ensure load distribution, Runtime Service Name attributes for all Windchill adapters should be same and should match the value of wt.federation.ie.VMName property defined in wt.properties. For example, if you have three method servers defined (i.e. wt.manager.monitor.start.MethodServer=3), you should define 3 instances of Windchill adapter as described above. Each of the 3 instances will have distinct Service Name (s) s.a. WindchillAdapter1, WindchillAdapter2, WindchillAdapter3 and distinct Port (s) s.a. 18001, 18002, 18003 etc. However, each of the three instances will have the same Runtime Service Name and this name is the same as the value of wt.federation.ie.VMName as defined in wt.properties. In our case, this value is: com.ptc.mn.nansen.Windchill. See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

There are other parameters for debugging and other functions which you can chose according to your needs.

6. Click **OK.** This creates an instance of Windchill adapter.

After creating an instance of Windchill adapter for each method server, you should stop and restart the Windchill server. When the method servers startup, they will look up Windchill adapter instances in the LDAP and try to bind with them in order. If they get a "java.net.BindException: Address already in use" exception, they go to the next Windchill adapter instance and try to bind with its port. This process is repeated for each method server until it finds an unoccupied port to bind to or until all Windchill adapter instances in LDAP are exhausted.

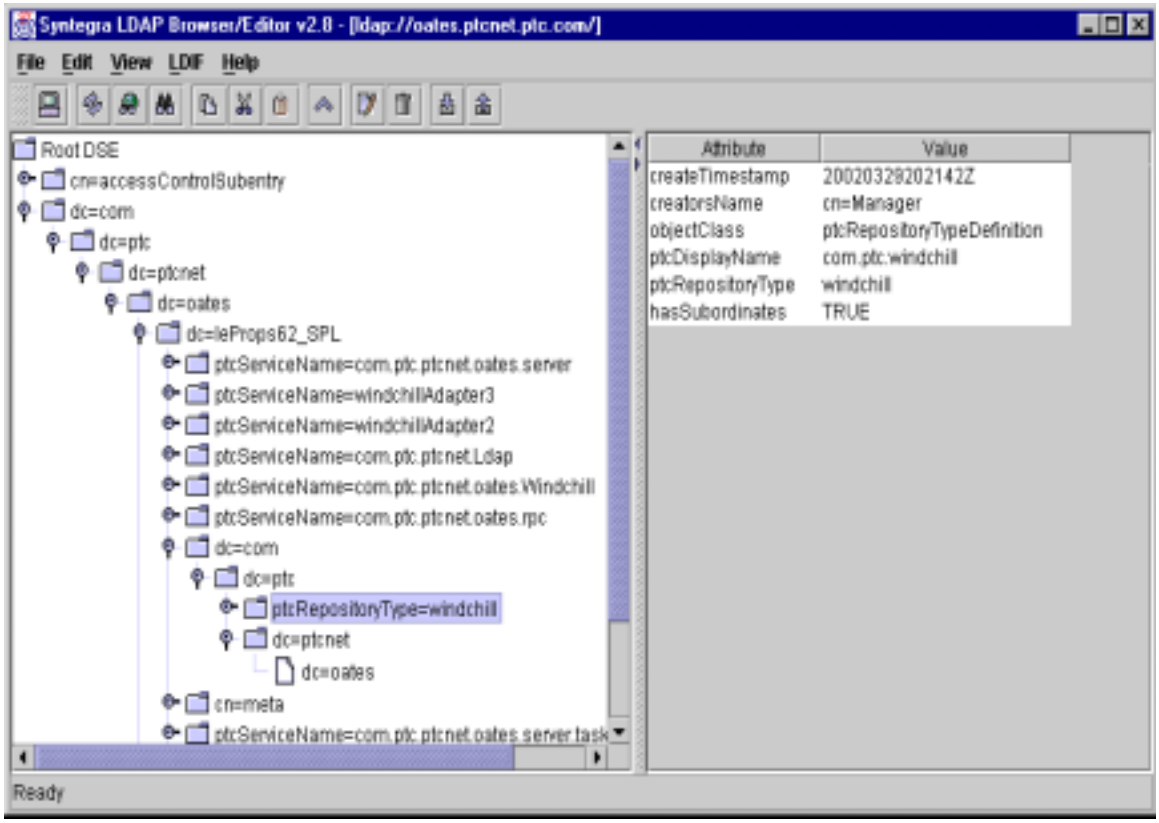
Changing LDAP Properties for Load-Balancing Among Instances of Windchill Adapter

Once you have defined instances of Windchill Adapters for each foreground Method Server, you have to create additional entries for these Adapters in your

Aphelion LDAP to ensure that Info*Engine can load balance among these instances.

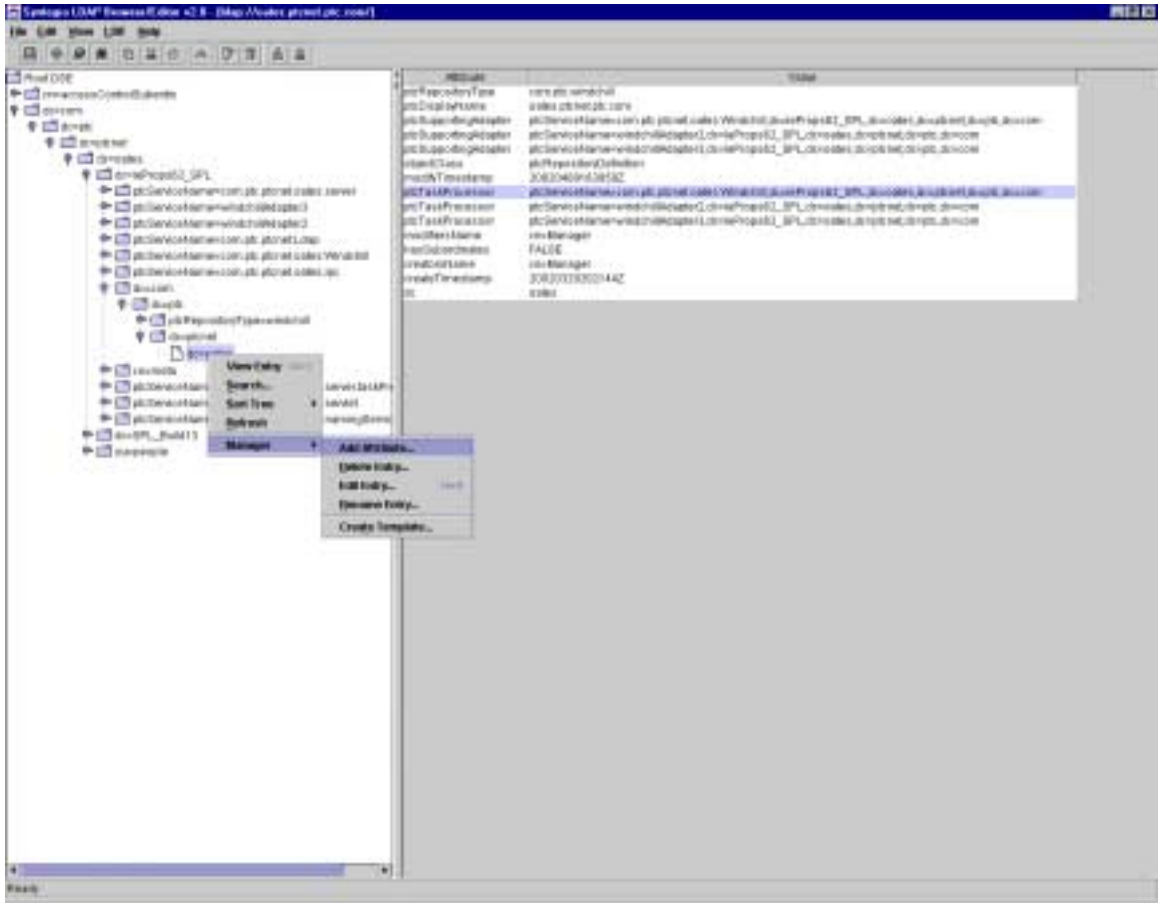
1. Open your LDAP Browser and connect to the Aphelion LDAP your Windchill install is using to store Info*Engine settings.

2. Expand your Info*Engine properties subtree until you get to "dc = your_machine_name". For example, the expanded subtree (below IeProps), could be similar to the following: "dc=com, dc=ptc, dc=ptcnet,dc=oates", where the machine name is oates. Refer to the figure below.

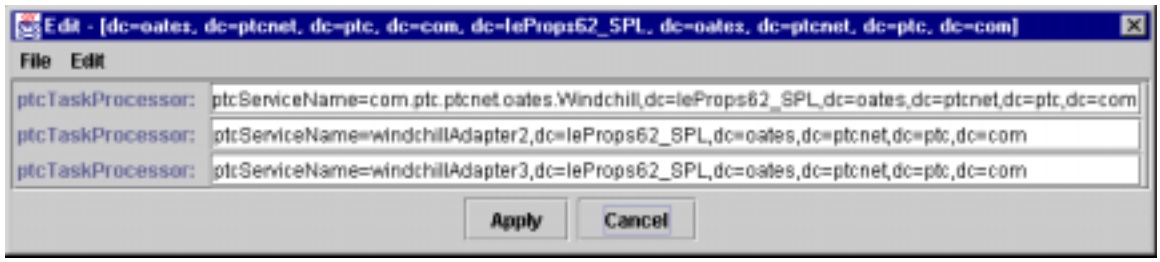


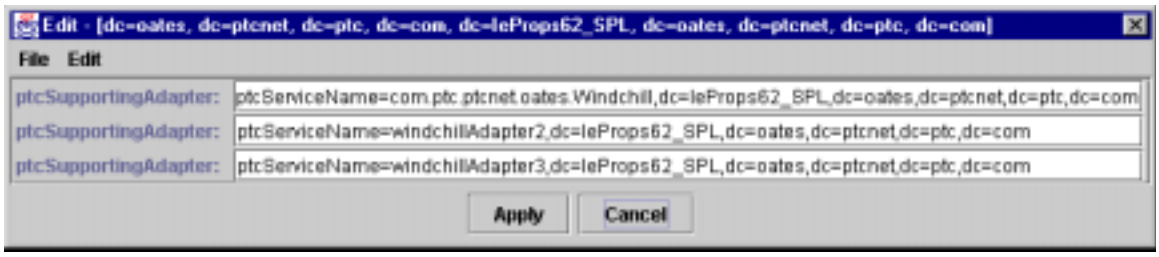
3. Add additional attributes to "dc=your_machine_name". To do this, right click on "dc=your_machine_name" (For example, you would right click on "dc=oates") and follow the process as shown in the figure below. The attributes that you have to add are ptcTaskProcessor and ptcSupportingAdapter. These are multi-valued attributes and there should be one instance of both attributes for each adapter. The installer creates one instance of both attributes for the Windchill Adapter it creates. You will have

to define additional values for both attributes for each Windchill Adapter instance that you create.



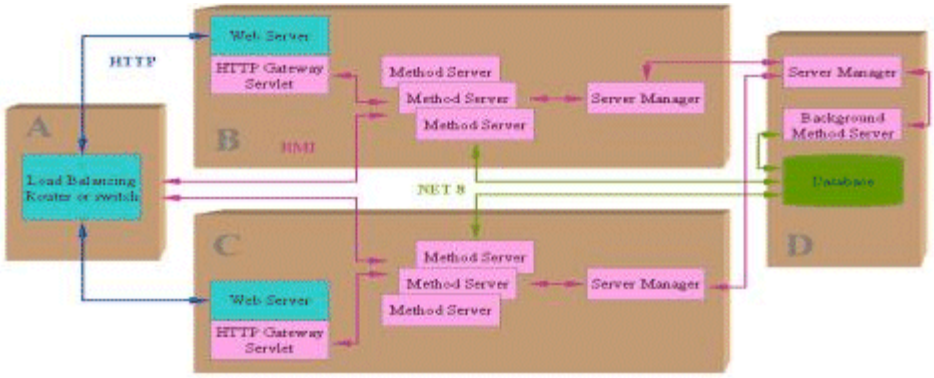
4. For example, we defined two additional Windchill Adapters and when the attributes are added, our ptcTaskProcessor and ptcSupportingAdapter look like the following:




Edit - [dc=oates, dc=ptcnet, dc=ptc, dc=com, dc=leProps62_SPL, dc=oates, dc=ptcnet, dc=ptc, dc=com]
File
Edit
ptcSupportingAdapter:
ptcServiceName=com.ptc.ptcnet.oates.Windchill,dc=leProps62_SPL,dc=oates,dc=ptcnet,dc=ptc,dc=com
ptcSupportingAdapter:
ptcServiceName=windchillAdapter2,dc=leProps62_SPL,dc=oates,dc=ptcnet,dc=ptc,dc=com
ptcSupportingAdapter:
ptcServiceName=windchillAdapter3,dc=leProps62_SPL,dc=oates,dc=ptcnet,dc=ptc,dc=com
Apply
Cancel

Server Cluster Configuration

A cluster is a collection of computers that can be accessed independently or as a single unit. The cluster can take a single work unit, defined as either a HTTP request or RMI method call, and distribute that request to one of many Windchill servers. The main advantages of a cluster over a single server are increased performance, scalability, and reliability. Performance increases because Windchill server processes have less competition on a given node. The system is scalable. As the load increases, additional Windchill servers can be added. This scalability also provides a more reliable system. In the event that a Windchill server fails, requests can be directed to the remaining servers.



Example of a Server Cluster

A typical Windchill cluster consists of three segments: a load-balancing router (A), a persistence storage server (D or Cache Master), and one or more Windchill servers (B, C or Cache Slaves).

The load balancing router takes requests from clients and distributes them among one of Windchill's many servers. This router may also act as a firewall, separating the Windchill cluster from the rest of the network. To the rest of the network, this router will appear as a single Windchill server. Configurations vary widely depending on load balancing software and hardware. This guide will only discuss the configuration of the Windchill server systems. Refer to the router documentation for more information.

The Windchill servers that participate in a cluster are similar to a simple autonomous Windchill server except that they share a database and caching mechanism with other members of the cluster. Since the caches are subject to updates from a top level cache master, these nodes are also known as Cache Slaves. They also share a common DNS name to direct clients to the cluster instead of the individual Windchill server.

The final segment of a Windchill cluster is the persistence storage server. A single database is used to store persistent data for the entire cluster. In a clustered configuration, one host is nominated the Cache Master whose responsibility it is

to synchronize all slave caches. Caching is managed by the server-manager process on each node in the cluster. In this example, the server-manager running on the persistence storage server also serves as the cluster's Cache Master. Additional properties must be set in wt.properties to properly configure the distributed caching mechanism. As in other multi-method server configurations, only one method server should process queued entries (the background method server). While the residency of the background method server is arbitrary, the following example places it on the persistence storage server.

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

Windchill Server Configuration (Cache Slaves)

All Windchill servers must be configured similarly so any one of them can respond to the same request with the same response. Clients access this system as if it were a single server, though there are several servers responding to the requests. The necessary changes will be listed and then explained in more detail later. The configuration in the following example makes all http access must go through the main router, which results in fully using the load-balancing and fail-over capability of the router. All properties describing URLs should point to host A.

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

To configure a server cluster, start with identical wt.properties files on each node and add the following properties to the end.

```
java.rmi.server.hostname=A

wt.server.hostname=

wt.queue.executeQueues=false

wt.cache.master.codebase=http://D/windchill

wt.httpgw.hostname=$(java.rmi.server.hostname)

wt.server.codebase=http://$(java.rmi.server.hostname)/windchill

wt.httpgw.url.anonymous=$(wt.server.codebase)/servlet/WindchillGW

wt.httpgw.url.authenticated=$(wt.server.codebase)/servlet/WindchillAuthGW

wt.sysadm.url=$(wt.server.codebase)/servlet/WindchillSAGW

wt.intralinkgw.url=$(wt.server.codebase)/servlet/IntralinkGWLogon

wt.pdmgw.url=$(wt.server.codebase)/servlet/PDMGWLogon
```

All slaves need the following entry in the /etc/hosts file. This configuration must not be done on the master:

```
127.0.0.1 A
```

java.rmi.server.hostname=A

Each node in the cluster must act as if it were the cluster in its entirety. A request to an individual node in the cluster needs to generate the same response as every other node in the cluster. Set the local machines host lookup to resolve the common name to the current node. The following entry should be added in the Servers hosts file (UNIX = /etc/hosts; NT = <nt>/system32/drivers/etc/lmhosts):

```
127.0.0.1 A
```

For clients to access these nodes as a single server cluster, the RMI hostname must be set to the commonly known name for the entire cluster, in this case A.

wt.server.hostname=

Verify that the Servers hostname is not set. This variable is used for communication between server instances for a given server machine. By setting this value to the empty string or removing it from wt.properties, the server will use the localhost address. Be sure that there is no trailing ‘white space’.

wt.queue.executeQueues=false

Queue entries are stored on the persistence server. Since multiple Windchill servers are accessing the same database, queue processing is limited to one method server. PTC recommends that queues be executed from a background method server running on the persistent store server. This will be discussed in the Persistence Storage Server Configuration section, below.

wt.cache.master.codebase=http://D/windchill

With multiple Windchill servers accessing a single database, a mechanism to maintain data integrity is needed. The master cache serves two purposes. All cache entries in a slave are replicated to the master for use by its peer caches to improve performance. When an entry is added to the master cache, slaves that hold a stale cache entry for this object are instructed to remove this entry from their cache. This allows multiple Windchill servers to share cache values while maintaining integrity. The master codebase property needs to be set to the codebase of the master cache and is used to retrieve the master’s wt.properties.

Advanced administrators may use the file protocol instead of HTTP – however, since the master cache wt.properties is read only at startup there is no appreciable performance improvement. If the file protocol is chosen, you must also set property wt.cache.master.hostname to the local host name of the top-level master cache. This allows the master cache server to recognize itself.

Persistence Storage Server Configuration (Master Cache)

The persistence storage server holds all persistent data. It also maintains the communication between cluster nodes. PTC recommends that a background method server be started on this server to be a single point for queue execution. This node has its own unique wt.properties since it is not a member of the cluster

itself. The configuration in the following example makes all http access must go through the main router, which results in fully using the load-balancing and fail-over capability of the router. All properties describing URLs should point to host A. See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

Again, the changes to wt.properties are listed and are explained in further detail.

```
java.rmi.server.hostname=D

wt.cache.master.slaveHosts=B C

wt.manager.monitor.services=BackgroundMethodServer

wt.manager.monitor.start.BackgroundMethodServer=1

wt.httpgw.hostname=A

wt.server.codebase=http://$(wt.httpgw.hostname)/windchill

wt.httpgw.url.anonymous=$(wt.server.codebase)/servlet/WindchillGW

wt.httpgw.url.authenticated=$(wt.server.codebase)/servlet/WindchillAuthGW

wt.sysadm.url=$(wt.server.codebase)/servlet/WindchillSAGW

wt.intralinkgw.url=$(wt.server.codebase)/servlet/IntralinkGWLogon

wt.pdmgw.url=$(wt.server.codebase)/servlet/PDMGWLogon
```

java.rmi.server.hostname=D

An alias is not needed on this server as it was on the cluster nodes. Members of the server cluster will use this name to communicate with the persistence storage server.

wt.cache.master.slaveHosts=B C

The server-manager on this machine acts as the master cache for all slave caches within the cluster nodes. The slaveHosts variable defines a list of slaves that have permission to retrieve and update the master cache. This variable is one of the few variables in wt.properties that is read at runtime, enabling administrators to add nodes to the cluster and this variable without requiring a restart of the master host.

wt.manager.monitor.services=BackgroundMethodServer
wt.manager.monitor.start.BackgroundMethodServer=1

Queue entries are stored in the database for later execution. Only one method server should be pulling entries from this queue and performing the execution. Queues should be executed from a background method server running on the persistence storage server. The monitor services variable defines the services that a server-manager will startup and monitor. All cluster servers (Cache Slaves) must have the executeQueues flag set to false. This flag is true by default, and will execute queues on this background method server.

wt.httpgw.hostname=A

Queue entries that yield URLs during processing, (WorkFlow notifications) should ensure that the hostname is the HTTP gateway hostname of the cluster and not the persistent data server.

wt.server.codebase=http://$(wt.httpgw.hostname)/windchill

All properties describing URLs should point to host A.

Configuring the Master Cache Server into the cluster (Advanced users only)

The example above shows how to configure the cluster with the Cache Master on the Persistent Data Storage Server (Server D). In this configuration, the Cache Master server is not available to process interactive client requests since the load balancer only forwards requests to Cache Slaves (e.g. servers B and C).

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

Depending on the processing capacity of Server D and/or the location of the Oracle server, it might be beneficial to start additional Method Server(s) on server D. This is done by changing properties wt.manager.monitor.services and wt.manager.monitor.start.MethodServer. Server D must also be added to the cluster definition (A) in the load balancer. Remember to ensure only one Method Server has queue processing responsibility.

Having configured the Master Cache Server into the cluster, you must configure a second ServerManager with a separate wt.properties file. One of the ServerManagers will export an object stub containing the cluster name as defined by the java.rmi.server.hostname property. Since that hostname was set to resolve to the local host on cluster members, that stub is no good for connecting to a remote master server manager. A second ServerManager is required which will export an object with the host's real hostname or IP address. To accomplish this, you'll need to run a separate server manager on another port number.

For example, create a new directory (on host D - the master) under /Windchill/codebase called master and copy wt.properties into it. Then make the additions shown in the following examples to the wt.properties files.

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

wt.properties on D at /Windchill/codebase/master/wt.properties:

java.rmi.server.hostname=D

wt.manager.port=4999

wt.cache.master.slaveHosts=B, C

wt.manager.monitor.services=

Undefining the wt.manager.monitor.services property ensures no Method Servers are started by the second Server Manager.

wt.properties on D at /Windchill/codebase/wt.properties:

java.rmi.server.hostname=A

wt.cache.master.codebase=http://D/Windchill/master

wt.properties on Cache Slave (B or C):

java.rmi.server.hostname=A

wt.server.codebase=http://A/windchill

wt.cache.master.codebase=http://D/Windchill/master

wt.queue.executeQueues=false

Windchill servers find their wt.properties file as a class path resource, so to execute two server managers using different property files, you will have to add /Windchill/codebase/master to the Java class path when executing the master server manager. For example, to start both server managers on host D:

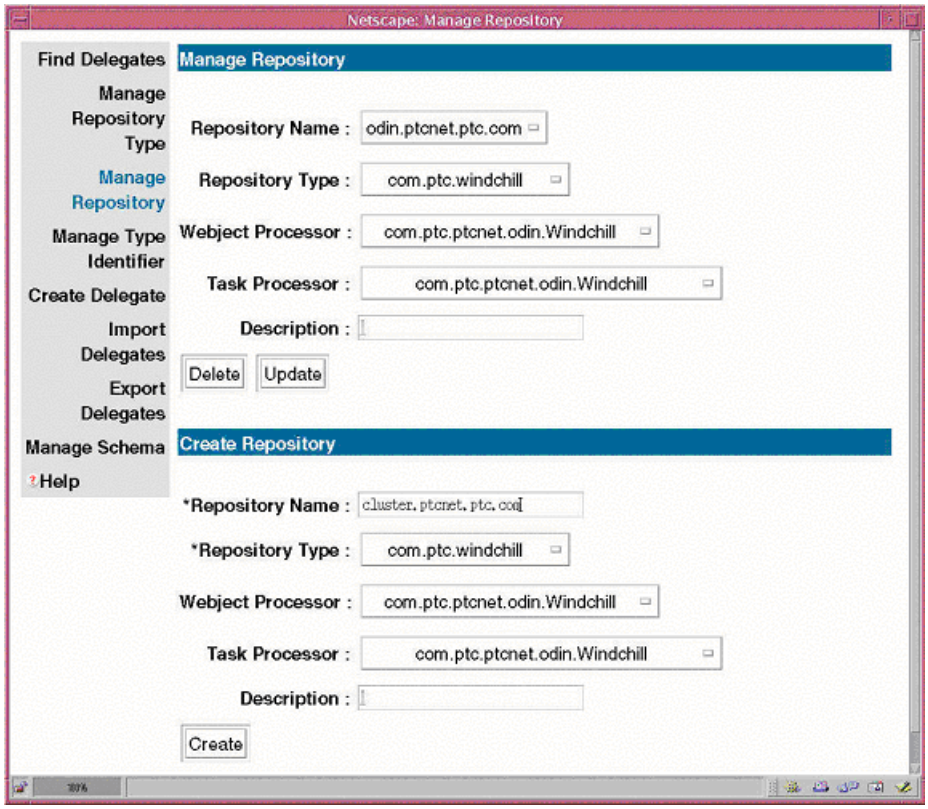
java wt.manager.ServerLauncher

java -cp /Windchill/codebase/master:$CLASSPATH wt.manager.ServerLauncher

Creating Repository in LDAP

A repository needs to be created in the LDAP for each of the cluster nodes (server B and server C). You should use the I*E Task Delegate Administrator to create these repository definitions. The Repository Name should be set to server A.

For example, when we added a repository definition, the Manage Repository page appeared as shown below. In this example, server A was cluster.ptcnet.ptc.com and server B was odin.ptcnet.ptc.com



Troubleshooting Cluster Problems

During the initial deployment of a cluster configuration it is strongly recommended that the following debug flags be set in wt.properties on the Master Cache and all Slave Cache hosts:

```
wt.cache.verboseServer=true
```

```
wt.cache.verboseClient=true
```

Monitor the Server-manager and Method Server log files paying particular attention to Security Exception: Access denied and/or Server Unreferenced messages, either of which could indicate name resolution misconfiguration.

Visit the **Server Status** tab in the SystemConfigurator page, which lists the servers configured in a cluster.

If you find that a host is not listed and you have verified that the machine is alive, you must check the configuration for that host against the parameters described above.

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

5

Windchill Client Tuning Guidelines

This chapter contains techniques to improve Windchill performance from the Windchill client.

Topic **Page**

Resource Contention 5-2
Browser Cache Settings 5-2
Using the Bootstrap Client 5-2
Netscape Browser on UNIX 5-3
RMI tunneling 5-3
Known issue with Internet Explorer 5 with iPlanet and Jrun 3.0 5-5
Measuring Network latency 5-5
Measuring Network Traffic 5-6
Implementing a Customized User Interface 5-7

Resource Contention

The Windchill client memory footprint varies based on the size of the database and the complexity of tasks being performed. To measure the amount of RAM used by an applet you can use the plugin's Java console and enter 'm'.

Windchill users sometimes require multiple applications to be active in addition to the Windchill User Interface. This means balancing the available RAM between active processes. If insufficient RAM is available to manage all active processes, the Operating System resorts to its disk as a secondary store. Check that client computers are configured with adequate paging space to accommodate this.

To avoid the resultant slowdown, users are recommended to limit the number of simultaneously open programs. On NT, users might benefit from the option to boot foreground application performance (see the Performance Tab on the System Properties dialog).

Browser Cache Settings

Ensure that the client's browser cache is configured to read objects from the local disk where appropriate. For Netscape, under the Preferences tab, specify that the document in the cache is compared with the document on the network once per session. For Internet Explorer, under the Internet Options tab (Temporary Internet Files->settings) specify Automatic version checking.

Using the Bootstrap Client

The bootstrap feature of Windchill allows Java applications that would normally be downloaded from a server to be loaded from locally cached JAR files. This improves performance by eliminating the need to load Java class files and other resources from across the network.

The bootstrap feature automatically manages a cache of local JAR files that correspond to remote server codebases. (A codebase is the URL to the root of a directory tree containing Java class and resource files.) The bootstrap feature provides the following functionality:

- Preserves namespace separation between codebases.
- Preserves the security of the sandbox to which code from each remote codebase is subject.
- Does not add codebase JAR files to the Java system class path (the CLASSPATH environment variable) of the client system.

A major benefit of using the bootstrap feature is that maintenance of each server codebase remains centralized and no additional per-client administrative responsibilities are incurred. Even if a codebase undergoes frequent changes, the bootstrap feature recognizes the existence of new JAR files and allows you to download the files.

To use the bootstrap feature, clients must have the Windchill bootstrap package installed, and servers must contain JAR files of the client codebase. If a client has the bootstrap feature installed but the Servers codebase does not contain the required JAR files, the bootstrap feature is ignored. Likewise, the existence of JAR files in a server codebase will not affect clients that do not have the bootstrap feature installed.

Use of the browser plugin JVM is mandatory beginning at R6. This allows applets to use the plugin's JAR caching instead of the bootstrap.

Netscape Browser on UNIX

Note: Browser performance is adversely affected when its executables and class libraries are network mounted. UNIX based Netscape browsers are typically NFS mounted, as are user home directories. A combination of a poor class loading algorithm and remote location of system and user class libraries can cause applet loading to be unacceptably slow.

Applet class loading with a network mounted Netscape browser may be improved by copying the $(wt.home)/codebase/3rdparty.jar file into the <installdir>/java/classes directory (where <installdir> is the Netscape installation directory). It is possible to copy the wt.jar file for additional performance, but this is only recommended where namespace separation is not an issue. For example, where there is only one Windchill codebase accessible by clients

Users are strongly encouraged to ensure the MOZILLA_HOME environment variable points to the Netscape installation directory to ensure consistent system class loading. This applies regardless of the location of 3rdparty and wt.jar files.

For more information about administering the bootstrap client, please see The System *Administrator's Guide for Windchill Technology*.

RMI tunneling

When tunneling RMI over HTTP, the Java RMI specification only supports HTTP on the default protocol port 80 and a fixed URL path of /cgi-bin/java-rmi.cgi.

The master RMI socket factory installed by the bootstrap package uses a series of secondary socket factories to connect to the RMI server.

They are as follows:

1. Direct socket connection to RMI server host and port identified in the RMI stub. This is the ideal RMI environment.
2. HTTP to RMI server host and port. This will leverage a client-side HTTP proxy if necessary to reach the RMI server.
3. HTTP to RMI server host using port 80 to post calls to /cgi-bin/java-rmi.cgi. This will get through server-side firewalls or reverse proxy servers if necessary to read the RMI server.

4. HTTPS to RMI server host using port 443 to post calls to /cgi-bin/java-rmi.cgi. This will leverage the SSL encryption capabilities of the browser and web server to provide private communication.

The master socket factory attempts each connection in turn and uses whatever one successfully connects. Note that if a firewall does not reject connections, this failover behavior is defeated. In that case, it will be necessary (for he time being) to configure the client to use the specific secondary socket factory that is required. This can be done by setting the rmiSocketFactory property in the client's .wtboot.properties file. The socket factory classes corresponding to the above master socket factory are:

1. sun.rmi.transport.proxy.RMIDirectSocketFactory
2. sun.rmi.transport.proxy.RMIHttpToPortSocketFactory
3. sun.rmi.transport.proxy.RMIHttpToCGISocketFactory
4. sun.rmi.transport.proxy.WTRMIHttpsToCGISocketFactory
5. sun.rmi.transport.proxy.WTRMIHttpToCodebaseSocketFactory

Setting the rmiSocketFactory property allows the client to bypass the sequential approach for opening an RMI connection by specifying the socket factory directly. This can provide faster initial connection by avoiding the need to wait for timers to expire before attempting the next socket factory.

wt.tools.javarmi.JavaRMIServlet

When direct connectivity to RMI servers is not allowed, forwarding of HTTP requests requires that the web server respond to requests for /cgi-bin/java-rmi.cgi. An actual .cgi file is provided in the Java JDK. The Java RMI specification expects this file to be added to the web server's cgi-bin directory.

To improve performance, security, and flexibility, Windchill delivers a servlet that can be mapped to the same URL. The servlet class is wt.tools.javarmi.JavaRMIServlet. It adds additional security because it can be configured through servlet initialization parameters to only forward connections to a predefined range of destination port numbers. The java-rmi.cgi file provided in the JDK, on the other hand, allows the HTTP request to identify any port number on the local host, opening other services to potential attack. It adds performance by not starting a new process for each RMI call. It adds flexibility by allowing itself to be configured to forward requests to a non-local RMI server host, thereby acting as a RMI proxy server. The servlet parameters are:

1. serverHost
2. minPort
3. maxPort

Known issue with Internet Explorer 5 with iPlanet and Jrun 3.0

When starting a Windchill applet in Internet Explorer using an iPlanet web server and Jrun 3.0 servlet engine users can experience a 30 second pause during initial connection. The reason is due to the use of HTTP Keep Alive and the mishandling of socket level communication between browser, web server and servlet engine. To work around this issue, a property wt.httpgw.HTTPAuthentication.setKeepAlive has been added to bypass KeepAlive messages on applet startup. The default value is false which is appropriate for those environments not observing the 30 second pause.

An alternative to setting the property in Windchill is to set the web server to close persistent connections immediately rather than after 30 seconds.

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

Measuring Network latency

To aid in debug and performance measurement of Windchill applets you can set the property wt.method.clientMethodTiming to true in wt.properties. This property can be changed without restarting the Method Server since the file is served by the web server when the browser initiates its first RMI connection with Windchill (i.e. the first time an applet is started).

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

When set true, this property causes every remote method call invoked by the client to be logged in the browser's Java console. For example, this sample message was produced when opening a folder in the Windchill Explorer:

```
INVOKE: start
wt.clients.folderexplorer.FolderBusinessObject$LiteObjectProvider.
getWTBusinessObjects
sending client authentication.
sending target object.
waiting for response...
received context id.
received server feedback.
received method result.
INVOKE: end
wt.clients.folderexplorer.FolderBusinessObject$LiteObjectProvider.
getWTBusinessObjects, 1344 ms
```

This extract shows that the elapsed time required to retrieve the contents of the selected folder was 1344 milliseconds.

You can estimate the latency introduced by the network by measuring the 'server-busy' component of the same call. This can be performed by setting the property

wt.method.serverMethodTiming to true. The Method Server(s) must be restarted following the change.

The following extract illustrates the text appearing in the MethodServer log file for the preceding folder operation.

```
INVOKE: start
wt.clients.folderexplorer.FolderBusinessObject$LiteObjectProvider.
getWTBusinessObjects
Registering context RMI TCP Connection(3092)-132.253.8.130, host =
132.253.8.130
reading client authentication.
reading arguments.
Invoking target method.
Sending feedback for Thread[RMI TCP Connection(3092)-
132.253.8.130,10,RMI Runtime]
sending result.
Unregistering context RMI TCP Connection(3092)132.253.8.130
INVOKE: end
wt.clients.folderexplorer.FolderBusinessObject$LiteObjectProvider.
getWTBusinessObjects, 1312 ms (15/1281/16)
```

As can be seen from this extract, 1312 ms of elapsed CPU time was required to process the request. By subtracting the ‘server busy’ time from the client’s elapsed time we can measure the latency caused by the network (e.g. 1344 minus 1312 equals 32ms).

Note the line beginning “Sending feedback for Thread ….” in the above extract. This line might be repeated multiple times during a single method call and indicates a feedback message is being sent to the client while the server is busy querying data from the RDBMS. By default one feedback message is sent for every 20 items read from the database. You can reduce the frequency of feedback messages to remote clients by increasing parameter wt.pom.feedbackInterval in db.properties (at the cost of less responsive client interrupts of the server during long running transactions).

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

Measuring Network Traffic

In order to measure the amount of Windchill specific data being transferred on the network, you can use the wt.method.RemoteMethodServer command line tool.

For example:

```
> java wt.method.RemoteMethodServer
Start date             = Fri Jun 01 13:29:27 CDT 2001
Total RMI invoke calls = 155
Total method contexts  = 156
Active method contexts = 0
Current free memory    = 98282168 bytes
Current total memory   = 133955584 bytes
RMI client sockets     = 3
RMI bytes in           = 1959595
```

```
RMI bytes out          = 2780382
Database connections   = 5
```

This extract shows the number of bytes read and written on the network by an individual Method Server. Note that the ‘bytes in’ field is inclusive of approximately 630 bytes from the call itself.

Alternatively both commercial and freeware tools are available to collect the network data. On Solaris the snoop utility can be used to capture and analyze network traffic. Web server log files can be useful for collecting data also.

Implementing a Customized User Interface

The state of current networks and technical issues in current Web browsers requires that a user interface specialist carefully consider the best technology to present user interface tasks. HTML is a better choice for most tasks because of greater download performance. Some tasks may require an applet, but applets should be used sparingly.

Avoid implementing applets where clients are likely to use a low speed WAN.

If applets are unavoidable, implement the business logic (‘heavy-lifting’) in the method server and, where possible, query and return only the required display attributes to the applet. Try to cache frequently used and resource intensive data structures in a customized service or static class and expose them to the client applet where appropriate.

Avoid loading classes until necessary. It is sometimes possible to avoid class loading in the client by implementing the method in the server and returning a lightweight object to the client. The following code extract illustrates how to implement a static inner class which exposes some complex or resource intensive immutable data structure (foo) to the client. It assumes that Footype is serializable or externalizable.

```
Package wt.clients.x.y
…
import wt.method.RemoteAccess;
import java.io.Serializable;
import java.rmi.RemoteException;
import wt.method.RemoteMethodServer;
import java.io.IOException;
import java.lang.reflect.InvocationTargetException;
import java.beans.PropertyVetoException;
…
public class AppletTaskLogic
{
private static FooType clientFoo = null;
…
   private static void initializeFoo()
   {
      RemoteMethodServer server;
      server = RemoteMethodServer.getDefault();
      try {
         // Invoke Server side method to get foo
```

```
        clientFoo = (FooType)server.invoke("getFoo",
"wt.clients.x.y.AppletTaskLogic$FooServer, (Object)null,
(Class[])null, (Object[])null);
      } catch (InvocationTargetException e) { etc, etc
      …
   }

   Public class ServerFoo implements wt.method.RemoteAccess
      {
         private static FooType serverFoo = null;
         private static Object lock = new Object();

         public static FooType getFoo()
         {
            if (serverFoo != null)
               return serverFoo;
            else
            {
               synchronized(lock)  // init singleton
               {
                  if (serverFoo == null)
                  {
                  // Instantiate serverFoo here
                  }
               }
            }
            return serverFoo;
         }
   }
}
```

Setting HTML Page Expiration

With the default settings, all generated pages will expire as soon as they are rendered. This can have an important impact on performance if, for example, the page is regenerated because the browser window was resized. Analyze page usage for your site to determine a reasonable expiration time. For example, the search criteria form should have a long expiration time because the information on the page does not change frequently. A setting of 900,000 milliseconds is recommended for this particular page. This is the default setting with this release. Expiration on some pages can be set in the *wt.properties* file. Please consult the `properties.html` file in the `codebase` directory for more information on these properties. For setting the page expiration on your custom HTML pages see below.

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

Setting Page Expiration Time

HTML page usage should be analyzed to determine the appropriate page expiration time. To set the expiration time, do the following:

Optionally define properties in *wt.properties* for the different page expiration times to make the actual times easily configurable. See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

Call `HTTPResponse.setDateHeader` before calling `HTTPResponse.getOutputStream()` to set the expiration time in milliseconds.

```
public void handleRequest (HTTPRequest req, HTTPResponse resp)
throws WTException
{
resp.setDateHeader (“expires”, System.currentTimeMillis () +
PAGE_EXPIRATION);
…
OutputStream responsePage = resp.getOutputStream ();
…
}
```

Note: Please note that in this example, the `PAGE_EXPIRATION` value from wt.properties has been applied from the static initializer of this class.

6

Tuning Oracle

This chapter provides information on general Oracle tuning, as well as techniques to make your own applications run more efficiently through additional Oracle tuning.

Topic **Page**

Improving Oracle Database Performance 6-2
Tuning Oracle for Customized Applications.....6-7
Oracle Tools to Collect Database Statistics 6-17

By far the most effective approach to tuning is to work proactive. Therefore, the best time to start thinking about performance is during the design phase of your Windchill customizations. It is much more efficient to tune during the design phase rather than tuning after implementing your system.

When designing customizations, keep the following things in mind:

- Are there objects that should be cached in memory?
- Are the database queries optimized?
- Will an index improve the speed of the function without degrading overall performance?

If you have done extensive customizations that introduce new queries, it is critical that you figure out which indexes will help for those customizations. Because only you know the function and type of data involved in your customization, only you can assess the proper indexing approach. This guide provides a process to follow, not a set of suggested indexes.

The information presented in this section is meant to provide an implementation team with an overview of what they would need to do to tune Oracle customizations for an initial pilot. Ideally, each project should have a knowledgeable Oracle DBA. Depending on the size and scope of the Windchill implementation, administering and tuning Oracle for Windchill can be a full time and very important job.

Improving Oracle Database Performance

Oracle tuning has a substantial impact on performance. Recommended Oracle tuning activities include the following:

- Setting instance parameters
- Using cost-based optimization
- Using Oracle indexes (discussed in greater detail later in this chapter)

Setting Instance-Based Parameters

The next table contains recommended Oracle instance parameters for two different systems.

Parameter	Setting for systems with 512 megabytes of memory	Setting for system with 1 gigabyte of memory
DB_BLOCK_SIZE	16K	16K
DB_CACHE_SIZE	80M	200M
LOG_BUFFERS	20971520	20971520
LOG_CHECKPOINT_INTERVAL	20000	20000
SHARED_POOL_SIZE	24M	96M

Dynamic SGA Configuration

In Oracle 9i, the SGA can be configured as in prior releases to be static, or can now be dynamically configured. The size of the dynamic SGA is determined mainly by the values of the following initialization parameters: DB_BLOCK_SIZE, DB_CACHE_SIZE, SHARED_POOL_SIZE, LARGE_POOL_SIZE, and LOG_BUFFER.

How to Approximate SGA Sizing

The Oracle SGA under 9i is calculated based on the following formula.

db_cache_size + db_recycle_cache_size + db_nk_cache_size + shared_pool_size + large_pool_size + java_pool_size + Log_buffer

Using this formula, the following example shows the SGA sizing based on the init.ora parameter values derived from one of PTC's solution production servers.

- Server and Database configuration
  - CPU -- 8
  - Memory -- 16gb
- Init.ora Parameters
  - db_cache_size = 209715200 Bytes
  - db_recycle_cache_size=0
  - shared_pool_size=41943040 bytes
  - Large_pool_size=10485760bytes

- Java_pool_size=20971520bytes
- Log_buffer=1048576

- Total SGA SIZE

  209715200 + 41943040 + 10485760 + 20971520 + 1048576 = 284164096 Bytes

  This total is approximately 285 megabytes.

  Add in each db_nk_CACHE_SIZE. There can be up to 4DB_nk_CACHE_SIZE (2,4,8,16,32K) defined. One of the Block sizes is the default BLOCK SIZE, which is 16K for Windchill, and its CACHE SIZE is defined by DB_CACHE_SIZE. The size of the db_cache_size which is given as an integer value, shared_pool_size,Log_buffers, can be increased on decreased according to the available system memory for Oracle SGA.

For Systems that have more or less than 1GB size of memory, the size of the DB_CACHE_SIZE,LOG_BUFFERS and the SHARED_POOL should be decreased or increased to allocate approximately 25 percent of available of memory to the SGA. The DB_BLOCK_SIZE should be 16k for all installations regardless of memory size. A detailed description of each of the preceding parameters follows.

DB_BLOCK_SIZE

The default data block size for every Oracle server is specific to an operating system. The Oracle data block size is typically either 2K or 4K. The default data block size is not adequate for Windchill. Windchill's use of binary large objects (BLOBs) necessitates a larger value for the Oracle block size to reduce chaining of rows. A block size of 16K is recommended.

Each database's block size is set during database creation by the initialization parameter `DB_BLOCK_SIZE`. The block size cannot be changed after database creation except by re-creating the database. Although this parameter can be changed in the init.ora file, changing the parameter after database creation will have no effect unless the database is re-created.

DB_CACHE_SIZE

Specifies the size of the default buffer pool for buffers with the Primary block size (the block size defined by the db_block_size parameter). The value of this parameter will determine the size of the default buffer_cache, which also will affect the buffer cache, hit ratio.

DB_KEEP_CACHE_SIZE and DB_RECYCLE_CACHE_SIZE

These two parameters replace the Buffer_pool_keep and Buffer_pool_recycle parameters used in Oracle 8i. These two parameters are independent of DB_CACHE_SIZE. The default value of these two parameters is 0.

SGA_MAX_SIZE

Specifies the maximum size of SGA for the lifetime of the instance. In an Oracle database, there is no control over the SGA size once the instance is started. Oracle9i allows a DBA to modify the SGA size dynamically. This provides an SGA that will grow and shrink in response to a DBA command. For more information about cache advisory, please read the Oracle9i database performance tuning Guide and reference.

LOG_BUFFERS

The redo log buffer is a circular buffer in the SGA that holds information about changes made to the database. This information is stored in redo entries. Redo entries contain the information necessary to reconstruct, or redo, changes made to the database by INSERT, UPDATE, DELETE, CREATE, ALTER, or DROP operations.

Oracle server processes copy redo entries from the user's memory space to the log buffer in the SGA. The redo entries take up continuous, sequential space in the buffer. The background process `LGWR` writes the redo log buffer to the active online redo log file (or group of files) on disk.

The initialization parameter `LOG_BUFFER` determines the size (in bytes) of the redo log buffer. In general, larger values reduce log file I/O, particularly if transactions are large or numerous.

LOG_CHECKPOINT_INTERVAL

A checkpoint is the event during which the Database Writer process (DBWR) writes all modified database buffers in the SGA to the appropriate datafiles. When your database uses large online redo log files, you can set additional database checkpoints to take place automatically at predetermined intervals, between the checkpoints that automatically occur at log switches. The time necessary to recover from an instance failure decreases when more database checkpoints are set. However, there may be a performance impact on the Oracle server due to the extra I/O necessary for the checkpoint to complete.

Generally, unless your database consistently requires instance recovery on startup, you should set database checkpoint intervals so checkpoints occur only at redo log switches.

SHARED_POOL_SIZE

The total size of the shared pool is determined by the initialization parameter sHARED_POOL_SIZE. The default value of this parameter is 64,000,000 bytes if 64-bit, otherwise 16,000,000bytes.

Increasing the value of this parameter increases the amount of memory reserved for the shared pool and therefore increases the space reserved for shared SQL areas.

Oracle allocates memory from the shared pool when an SQL statement is parsed. The size of this memory depends on the complexity of the statement. In cases

where an SQL statement requires a new shared SQL area and the entire shared pool has already been allocated, Oracle can deallocate items from the pool using a modified least recently used algorithm until there is enough free space for the new statement's shared SQL area. If Oracle deallocates a shared SQL area, the associated SQL statement must be reparsed and reassigned to another shared SQL area the next time it is run.

Init.ora Performance Improvement Features

Two parameters allow performance improvement: PGA_AGGREGATE_TARGET and WORKAREA_SIZE_POLICY.

- PGA_AGGREGATE_TARGET -- specifies the target aggregate PGA memory available to all server processes attached to the instance. This parameter is enabled under Windchill to take advantage of the automatic enabling of SQL working area sizes used by intensive SQL operators such as SORT, GROUP BY, HASH-JOIN, BITMAP MERGE, and BITMAP create. Oracle uses this parameter as a target for PGA memory. Use this parameter to determine the optimal size of each work areas allocated in AUTO mode, which occurs when WORKAREA_POLICY is set to AUTO. Depending on the memory availability, the size of PGA_AGGREGATE_TARGET value can be increased and decreased. This parameter replaces the usage of SORT_AREA_SIZE within the database.
  - Minimum Value -- 10MB
  - Max Value -- 4GB
  - Recommended Value -- 60% of SGA memory

Caution: Setting pga_aggregate_target Parameter while running Oracle on HP-UX is not recommended. Verify that this parameter is commented out from the init.ora. There are reported problems of memory leak with Oracle on HP. One of them relates to pga_aggregate_target. (Oracle Bug # 2740084)

While allocating PGA_AGGREGATE_TARGET verify that you have enough memory for your SGA and also for non-Oracle processes.

- Tuning PGA_AGGREGATE_TARGET -- You can monitor the performance of PGA_AGGREGATE_TARGET using available PGA statistics and observe whether PGA_AGGREGATE_TARGET is under-sized or over-sized. Several dynamic performance views are available for this purpose. The following query on V$pgastat, for example, will provide instance-level statistics on the PGA memory usage.

  ```
  Select * from v$pgastat;
  ```

  For more information on PGA_AGGREGATE_TARGET tuning, please read Oracle 9i database performance tuning guide.

- SESSION_CACHED_CURSORS -- This init.ora parameter was basically introduced to reduce the number of parses within the SQL statement. This

parameter provides short cut access to frequently parsed statements, which results in less use of CPU time.

– Minimum Value -- 0 to operating system-dependent

– Recommended -- 10 -30

Care must be taken not to set this parameter to a high value, because the amount of fragmentation in the shared pool may be increased.

Automatic Undo Management

Oracle9i databases are capable of managing their own undo (rollback) segments. No longer will administrators need to carefully plan and tune the number and sizes of rollback segments or decide how to strategically assign transactions to a particular rollback segment. Oracle9i also allows administrators to allocate their undo space in a single undo tablespace with the database taking care of issues such as undo block contention, consistent read retention, and space utilization. With this design, you allocate undo space in a single undo tablespace, instead of maintaining a set of statically allocated rollback segments. For more information regarding automatic undo management please refer to Oracle documentation.

Implementing Cost-Based Optimization

PTC recommends that you use Oracle cost-based optimization. The cost-based approach generally chooses an execution plan that is as good as or better than the plan chosen by the rule-based approach, especially for large queries with multiple joins or multiple indexes. The cost-based approach also improves productivity by eliminating the need for an administrator to tune SQL statements by hand. Finally, many Oracle performance features are available only through the cost-based optimization. For more information about Oracle cost-based optimization, please see the *Oracle Reference Guide*.

Tuning Oracle for Customized Applications

When trying to tune the performance of an Oracle system there are two fundamental things to consider:

- The first is the SQL code being run against the database. If this code is inefficient, poorly indexed or accesses large amounts of data, the database performance may suffer.
- The other main area is the database internal memory and processes. Generally the memory areas and processes result in performance degradation as the result of small memory areas and interference between internal processes, both of which are usually the result of database requests (SQL) to perform more work than the machine/Oracle setup will allow for.

Before starting to tune Oracle characterize the CPU usage on the server machine. There are two possibilities:

- Poor performance accompanied by moderate to high CPU consumption -- It is likely the SQL being run against the database is inefficient and should be changed. See the section Identifying and Tuning SQL Statements below.
- Poor performance accompanied by low to moderate CPU consumption -- There are three possibilities:
  - There is internal database contention.
  - Some memory areas are too small.
  - There is not enough server utilization to generate high CPU usage.

If you suspect the problem is the internal database contention, see the next section, Identifying Internal Oracle Bottlenecks

Identifying and Tuning SQL Statements

Whenever an Oracle system appears to have a performance problem, the first thing that should be investigated is the SQL that the system is running.

Tuning SQL involves the following steps:

- Pre-tuning steps
- Identifying the poorly performing SQL
- Determining if the columns in the Where clause are indexed

Pre-Tuning Steps

Before starting any tuning exercise make sure the following two items have been done:

1. timed_statistics = true in the init.ora file or has been turned on at the system level with the command -alter system set timed_statistics =true;
2. The cost based optimization statistics are up to date. To update the statistics use the following command. Using Cascade=true will analyze the indexes as well. By default the value is false.

```
Execute dbms_stats.gather_ schema_stats(ownname=>'SCOTT',cascade=>True);
```

Identifying the Poorly Performing SQL

This is the most important step. The goal is to identify the top 5-10 statements using the most system resources. Tune those statements, and then re-identify the top statements, until “acceptable” performance has been achieved.

In order to determine which are the worst performing SQL statements, use a query like the following:

```
set linesize 150

set pagesize 100
```

```
col factor for 999,999,999,999

col  buffer_gets for 999,999,999,999

col disk_reads  for 999,999,999,999

col sql_text for a64

select buffer_gets,disk_reads,executions,sql_text,

(disk_reads*100)+buffer_gets Factor

from v$sqlarea

where disk_reads>100000

and buffer_gets>1000000

order by factor

/
```

In the results of this query, the worst performing statements appear at the bottom of the report. The following definitions apply the above statement.

1. buffer_gets: is the total number of times an SQL statement reads a database block from the buffer cache in the SGA. Since every SQL statement passes through the buffer cache, this value represents a metric for how much work is being performed.
2. disk_reads: is the number of times an SQL statement needed to have database block read from disk. (Rule of thumb: 50+ disk I/O can occur in one second).
3. Executions**:** represents the number of times that the listed statement has been run.
4. Factor: is the statement cost relative to other statements being executed. This can be expressed in the formula:

   `(disk_reads * 100) + buffer_gets.`

This formula is an adequate metric of the amount of work being performed by SQL statements. Using the weighting factor of 100 is arbitrary and reflects the fact that disk reads are inherently more expensive than buffer gets.

Determining if the Columns in the Where Clause are Indexed

Determine if all of the columns used in each of the Where clauses from the SQL statements returned from above have indexes defined for them. The following query prompts for the table and column names and will return indexes defined for the entered criteria.

```
col TABLE_OWNER for a10

col TABLE_NAME for a25

col COLUMN_NAME for a25

set linesize 150
```

```
select
a.INDEX_NAME,a.TABLE_OWNER,a.TABLE_NAME,a.COLUMN_NAME,a.COLUMN_POS
ITION, b.UNIQUENESS

from dba_ind_columns a, dba_indexes b where a.index_name in (

select index_name from dba_ind_columns where column_name =
upper('&enter_column_name') and

 table_name = upper('&enter_table_name')  ) and

a.index_name = b.index_name

order by column_position;
```

If there are no indexes defined, the first and easiest thing to do is make an index. The more unique and specific the better. When appropriate, create a unique concatenated index before a unique non-concatenated one. If a concatenated index can be created will depend on the business logic of the Windchill rose. When in doubt create a non-concatenated index. Do not over index doing as doing so could have a negative impact on performance.

The following code may be used to create indexes:

Non-unique single column:

```
create index owner.index_name on owner.table_name (column)
tablespace index_tablespace_name;
```

Non-unique multiple column:

```
create index owner.index_name on owner.table_name (column1,
column2, column3 etc..) tablespace index_tablespace_name;
```

Unique single column:

```
create unique index owner.index_name on owner.table_name (column)
tablespace index_tablespace_name;
```

Unique multiple column:

```
create unique index owner.index_name on owner.table_name (column1,
column2, column3 etc..) tablespace index_tablespace_name;
```

Please see the Oracle documentation for more information about creating indexes.

Pre-Production Windchill Analysis

In some cases, in pre-production Windchill environments, it may be necessary to see the SQL execution time by a particular method server connection/method server operation. This can be done by using the following procedure.

Using SQL trace to trace method server connections:

1. Find the method server connections -- Sid’s and serial numbers

   Select sid, serial#, machine,username,program from v$session;

2. Enabling tracing in sql*plus for each of the method server connections found above:

   ```
   execute dbms_system.set_sql_trace_in_session (sid, serial#,
   TRUE);
   ```

3. Perform the operation(s) to trace, then disable tracing. Disabling tracing can be done by:

   ```
   execute dbms_system.set_sql_trace_in_session (sid, serial#,
   FALSE);
   ```

4. Disabling tracing will cause one or more trace files to be generated in the user_dump_dest.

5. Go to the directory that will contain the trace file created in step 4. This directory can be found by using the query:

   ```
   select NAME,VALUE from v$parameter where name='user_dump_dest';
   ```

6. Use tkprof to interpret the output file.

   ```
   tkprof <tracefile> <outputfile> explain=username/password
   sys=no sort=exeqry,fchqry,execu,fchcu print =10
   ```

Where *tracefile* is the name of the trace file Oracle generated.

Outputfile, is the name of the file in which the results of the analysis should be placed.

Explain=username/password is the username and password of the Windchill user.

Print limits the output to show only the ten worst performing statements. To see all statements being executed set this value to1000.

This will limit the output to the ten worst statements, with the worst statements at the top of the output file. The total time spent executing the statement will be listed under the column name “elapsed”. However using this column as a guide is not particularly useful as the time includes all other processes that were contending for the CPU during the time the statement was executing. The column that gives a more accurate idea of the time spent executing a statement is “CPU”. For more information on how to interpret this file, please see the Oracle documentation.

With the preceding query, tracing is done for a single user session. In addition, with the following query, SQL tracing information can be collected for both single and multiple sessions based on Method Server Connections. The spool file can be modified if tracing is needed only for specific sid and serial#.

1. Login as sqlplus.

2. Use 'sqlplus / as sysdba' to run this script.

3. Enter username for tracing

4. Enter Spool file name before running each script, for example, trace_enable.lst before step 6 and trace_disable before step 7.

5. Execute each spooled file above separately within SQLPLUS as sysdba.

6. Run the following query to enable tracing for a given user -- '&username'

```
Select'EXECUTE DBMS_SYSTEM.SET_SQL_TRACE_IN_SESSION('||sid||','||serial#||',TRUE);'
from V$SESSION WHERE USERNAME = '&&username';
```

7. Run the following query to disable tracing for the same above given user.

```
'EXECUTE DBMS_SYSTEM.SET_SQL_TRACE_IN_SESSION('||sid||','||serial#||',FALSE);' from
V$SESSION WHERE USERNAME = '&&username';
```

Using Indexes Provided With Windchill

Windchill has built-in indexes that you can use. This process assumes you are using a standard Windchill system with no added indexes. If you have a system with customized indexes, you will need to compare your index list with a list from a standard system.

Follow the steps below to get a list of indexes from a standard system:

1. Run create_ddl_wt from a command prompt.

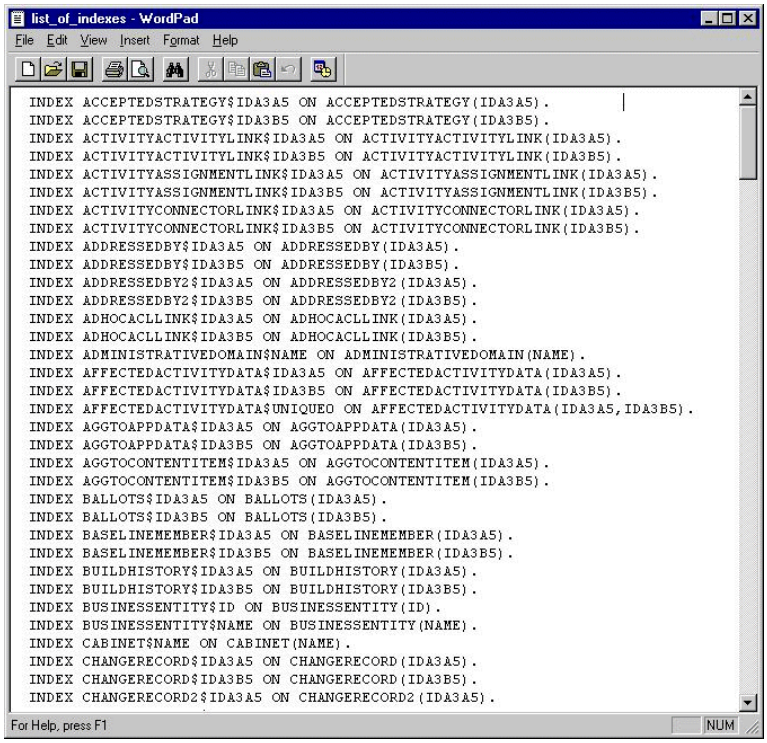
   Note: Because this script will drop and recreate all of the tables in your database, you may not want to run it on an existing Windchill Schema.

2. From SQLPLUS, run the SQL script find_all_index_info (refer to Appendix A). This script will create a list of all of the indexes on your system.

3. An output file, list_of_indexes.out, will be spooled to a temporary directory.

This output location has been coded into the script. For NT systems, it will default to C:\temp. For UNIX systems, you should change this to the path of your temporary directory, /usr/temp for example.

Sample Output File



Modeling Indexes

Add indexes to the Rose model to improve the performance of queries generated by your customized Windchill code. An example within the Windchill codebase would be to add a composite index on the IteratedFolderMemberLink-branchId and latest attributes to improve the performance of finding the folder that a given Iteration belongs to. This query is done every time a document or part is viewed, through an HTML property page or Java view task.

Be aware that any case insensitive String attribute searches will benefit from a Function Index (see below). Part and document names also fall into this category.

Modeling large BLOB (binary large object) columns into high volume classes, for example, documents and parts, is not recommended. Small BLOBs or IBA are a better alternative.

Composite Indexes In Windchill

Composite indexes are defined in the `CompositeIndexN` class model property. Currently only three composite indexes can be defined. The presence of a leading plus sign means that the value of the corresponding `CompositeIndexN` property in all superclasses of the given class will be *inherited*.

You can use the Index property of an attribute to index a single attribute that is part of the persistent class. If the attribute is aggregated to the *persistable* then the CompositeIndex property has to be used even for a single attribute.

For attributes of type ObjectReference the attribute name used in the index ends in *key.id*. Here projectId is the name of an aggregated ProjectReference to the ProjectManaged interface. *key.id* is included to navigate to the attribute id in ObjectIdentifier that is the actual column in the database that can be indexed.

If there is any confusion about what attribute name should be used in a CompositeIndex property refer to the ddl script for the class. Use the InfoReport utility to list all attribute names and their corresponding columns for the class of interest.

Function Indexes

Windchill can take advantage of function indexes when performing case insensitive text searches, for example, when searching for parts or documents by name.

To enable function indexes in Windchill Systems, make sure that the following two parameters are set in the initSID.ora file. In the absence of using Oracle Configuration Utility, these two parameters need to be manually added in the initSID.ora file:

```
query_rewrite_integrity=trusted

query_rewrite_enabled=true

compatible = "9.2.0"
```

Additionally the Windchill schema owner (wt.pom.dbUser) must be granted the "query rewrite" privilege. This privilege is granted automatically as part of Oracle Configuration Utility, otherwise it must be granted manually.

```
Sql> grant query rewrite to wtadmin
```

Function indexes were added to WTDocumentMaster and WTPartMaster in R6 to speed queries on the object's name attribute. WTPart and WTDocument customizations implemented following R6 will inherit these function indexes. However, customizations migrated to R6 will not automatically benefit. In this case the function indexes will need to be manually created. For example, if the customized class is AcmePartMaster you should create a function index in sqlplus as follows:

```
create index acmepartmaster$upper1 on acmepartmaster (UPPER(NAME));
```

You should also add the function index to the customized class in the Rose Model.

Defragmenting Indexes

Indexes occasionally become fragmented, leading to performance slowdowns. To defragment an index, use the command

Alter index <*index-name*> rebuild nologging;

where <*index-name*> is the name of the index to rebuild.

Make sure that you have enough free space available within your index tablespace. By having sufficient room in the index's tablespace to hold both the original index and the new version concurrently, you can avoid ORA-1652 errors.

Setting the Storage Clause

The default amount of disk space allocated when an index is defined is based on the TableSize property. The actual space associated with each TableSize is specified in tools.properties:

```
wt.generation.sql.tinyTableSize=8k

wt.generation.sql.smallTableSize=20k

wt.generation.sql.mediumTableSize=50k

wt.generation.sql.largeTableSize=100k

wt.generation.sql.hugeTableSize=1m
```

Separating Tablespace and Script Files for Indexes

Windchill provides the capability for you to specify a separate tablespace for your index. Putting the indexes into a separate tablespace allows the DBA to map the index tablespace to a dedicated hard drive for maximum disk access performance.

Windchill system generation will also create separate SQL script files for indexes. Moving the indexes into separate files allows database administrators to manage indexes easily. Periodically the indexes should be dropped and recreated to defragment them.

Identifying Internal Oracle Bottlenecks

To identify internal contention with a database use the following script to show where the bottleneck is occurring within the system. Before proceeding ensure that timed_statistics has been set in the init.ora file, or it can be set at the system level using the following command:

```
-alter system set timed_statistics =true;
```

The information generated will not be particularly useful unless the database has been running under load for two or more hours since timed statistics was enabled.

```
set pagesize 50

col event  format        a25  heading "Wait Event" trunc

col tw     format    999,999,999.99  heading "Time(sec)|Waited"

select event,time_waited/100 tw

from   v$system_event
```

```
order by time_waited desc;
```

This will return all of the “idle” and “non-idle” system events within Oracle.

The next step is to identify the event non-idle events taking the most time. These events will be closest to the top of the report. Idle events are activities Oracle performs which do not indicate an internal problem, whereas non-idle events if taking a significant portion of the time should be investigated.

Idle Events (Can be Ignored)

Large amounts of time spent performing these events do not indicate an internal database problem and can usually be considered insignificant.

- SQL*Net message to client -- event is triggered when a server is unable to send data from the server process to the client process. This could be due to network congestion. Check the network.
- SQL*Net message from client -- Server is waiting for client to do something. That something could be data or a carriage return. The server cannot continue processing until the client provides the stuff.
- SQL*Net more data from client -- Process waiting for data from the client application
- rdbms ipc message -- Usually background process waiting for work
- pipe get -- DBMS_PIPE read waiting for data
- Null event -- Miscellaneous
- pmon timer -- PMON waiting for work
- smon timer -- SMON waiting for work
- parallel query dequeue -- PQ slave or QC is waiting for a message.
- virtual circuit status -- Idle event that occurs with MTS, means MTS is not busy
- dispatcher timer -- Idle event that occurs with MTS, means MTS is not busy

Non-Idle Events (Should Be Addressed If Significant)

If the following events are near the top of the report and are taking a considerable percentage of the non-idle time spent, steps should be undertaken to address those events.

Note: Steps for resolving non-idle event problems are out of the scope of this document. Please contact your Oracle database administrator for assistance.

Non-idle events

- buffer busy waits
- db file parallel write

- db file scattered reads
- db file sequential read
- enqueue
- free buffer waits
- latch free
- log buffer space
- log file switch
- log file sync
- log file parallel write
- write complete waits

Oracle Tools to Collect Database Statistics

Effective data collection and analysis is essential for identifying and collecting system performance problems. Oracle provides a collection of tools that allow a performance engineer to gather information regarding instance and database performance. Important tools in the testing, development, and production environments are the Statspack and the Performance Manager.The following is a brief review on some of those available Oracle tuning tools and their relative advantages.

- Oracle Tuning Tools
  - SQL Trace and TKPROF -- To determine execution statistics for SQL Statements. The TKPROF facility accepts as input an SQL trace file and produces a formatted output file. For full syntax of TKPROF and information about how to interpret the trace files, please see the USING SQL TRACE and TKPROF chapter within *Oracle9i Database Performance Tuning Guide and Reference*.
  - UTLBSTAT/UTLESTAT -- This is a command line tool known as Bstat/Estat/ It is a set of sql scripts located under your $ORACLE_HOME/rdbms/admin directory that are useful for capturing a snapshot of system wide database performance statistics.
  - Statspack -- Command Line tool within Oracle Enterprise Manager suite.
  - Oracle Enterprise Manager, Performance Manager graphical user interface tool.
- Statspack -- Oracle Diagnostic Tool. Statspack is a set of performance monitoring and diagnostic tool provided by Oracle. It collects performance statistics data permanently in Oracle tables, which can later be used for reporting and analysis. The data collected can be analyzed using the report provided, which includes an "instance health and load" summary page, high

resource SQL statements, as well as the traditional wait events and initialization parameters. The Tool provides a systematic and effective approach for database tuning. Unlike BSTAT/ESTAT, Statspack is a very flexible tool that helps to diagnose both application-specific performance problems as well as instance-wide problems.

Advantages of Statspack Over Bstat/Estat

The script collection Bstat/Estat has some disadvantages when it is contrasted with the Statspack:

- Statspack collects more data, including high-resource SQL.
- Statspack precalculates many ratios useful when performance tuning, such as cache hit ratios, rates, and transaction statistics. Many of these ratios must be calculated manually when using Bstat/Estat.
- Statspack uses permanent tables owned by the PERFSTAT user to store performance statistics. Rather than creating and dropping tables each time, data is inserted into the existing tables, making it easier to compare historical data.

Using Statspack Proactively

Before you use Statspack most effectively for reactive problem investigations, you need to take some proactive steps. The most important step is that you need to establish performance baselines by capturing data you can use later for comparison purposes. It's a good idea to examine data on a regular basis even when no problems are occurring, to see how the load is changing over time and how changes in the system, such as newly installed applications or releases, are affecting it.

To get more information on Statspack tool, please check the following URL:

http://download-west.oracle.com/docs/cd/B10501_01/server.920/a96533/statspac.htm - 22560

Performance Manager Oracle Diagnostic tool

The Performance Manager (PM) is an Oracle diagnostic tool is installed as part of the following Packs:

- Diagnostic Pack
- Standard Management Pack
- Management Pack for Oracle applications
- Management Pack for SAP R/3

The Oracle Performance Manager captures, computes, and presents performance data for your database and operating system, allowing you to monitor key metrics required to effectively use memory, minimize disk input and output, and to avoid

resource contention. You can also record the data for replay. In addition, the Oracle Performance Manager provides a focused view of database activity by database session. The 9i Intelligent Agent acts as a collection agent to collect the performance data. Performance manager can be run in Enterprise Mode using the repository or in a standalone mode. If you connect in a standalone mode, your recordings and used-defined charts will not be saved. If you exit the Performance Manager, you will not be able to play back any recordings or access those user friendly charts in future.

For more information on Configuring Oracle Enterprise Manager, Performance Manager Environment, please refer to Oracle Documentation.

Note: A new feature of 9i is that Performance Manager can now connect directly to a database and by-pass the intelligent Agent's collection Service.

Note: Although Oracle Enterprise Manager, Statspack and BSTAT/ESTAT, and the V$ views have different interfaces (GUI, command line,SQL), the majority of data they collect and report on is extracted from the V$ views. Because V$ views are based on memory-resident data, when an instance is shutdown, the data related to the instance is lost.

Oracle Version Compatibility Matrix report for the Preceding Tools

The following table shows Oracle versions and the tools discussed in this section that the versions support.

Oracle Version	Supported Tools			
	TKPROF	Bstat/Estat	Statspack	(OEM) Performance Manager
9i Enterprise Edition	Yes	Yes	Yes	Yes
9i Standard Edition	Yes	Yes	Yes	No -- extra cost option

For more information on Oracle diagnostic tools features and options support for 9i, please refer to Oracle documentation.

7

Tuning the Servlet Engine

Topic	Page
Implementing a Throttle	7-2
Heap Size	7-4
Logging	7-4

Implementing a Throttle

This section describes how to limit the number of simultaneous HTTPGW requests in process. This can influence transaction throughput and server saturation.

JRun 3.0

To configure JRun 3.0 to limit the number of concurrent calls to the Method Servers to 20 and the queue length to 100, edit the file:

<install-dir>/lib/global.properties

Change the following two properties:

```
control.endpoint.main.active.threads=20

control.endpoint.main.max.threads=100
```

Tomcat 3.2.1

For Tomcat 3.2.1 the following files must be changed:

tomcat.properties

pool=true

pool.capacity=20

pool.controller=org.apache.java.recycle.AdaptiveController

server.xml

```
<Connector
className="org.apache.tomcat.service.PoolTcpConnector">

<Parameter name="handler"

value="org.apache.tomcat.service.connector.

Ajp12ConnectionHandler"/>

<Parameter name="port" value="8007"/>

<Parameter name="max_threads" value="20"/>

<Parameter name="max_spare_threads" value="20"/>

<Parameter name="min_spare_threads" value="5"/>

</Connector>
```

Note: Tomcat 3.2.1 does not appear to limit queued requests. This must be done in the web server by restricting the number of simultaneous connections the web server will accept.

Tomcat 3.3.1

Tomcat's Ajp13Connector is used by default in Windchill 6.2.6. You need to add additional properties for this connector in your $TOMCAT_HOME/conf/server.xml to enable throttling. The following block in your server.xml file controls the properties of this connector:

```
<Ajp13Connector port="8009" tomcatAuthentication="false"/>
```

Search for the above in your server.xml and modify it as is shown below:

```
<Ajp13Connector port="8009"
pools="true"
maxThreads="50"
maxSpareThreads="20"
minSpareThreads="5"
tomcatAuthentication="false" />
```

Note: The values above are just examples. Configure the values according to the requirements at your site.

The following are details of the properties specified above:

Thread Pool Properties	Description	Default
maxThreads	Maximum number of threads in Thread pool.	200
maxSpareThreads	Maximum number of spare threads. Unused threads will be terminated as needed to keep the number of spare threads under this number.	50
minSpareThreads	Minimum number of spare threads. Additional threads will be created as needed to keep the number of spare threads up to this number.	4
pools	Enables use of a Thread pool.	true

Note: If you are not using the default configuration for Tomcat 3.3.1, you will have to determine which connector you are using and make the changes listed above for that connector.

Heap Size

The servlet engines default Java heap size may need to be increased. PTC recommends setting the maximum heap size at 128MB.

JRun

To increase the heap size for Jrun 302 to 128Mb make the following change to <install-dir>/lib/global.properties:

user.javaargs=-Xms128m -Xmx128m -Xnoclassgc

Note: to run Jrun with the server HotSpot VM you may need to add ‘-server’ as the first argument in user.javaargs.

Tomcat

For Tomcat, edit tomcat.sh or tomcat.bat to add the following variable at the top:

TOMCAT_OPTS="-Xms128m -Xmx128m -Xnoclassgc"

Note: To run tomcat with the server HotSpot VM you need to add '-server' as the first argument in TOMCAT_OPTS. Please refer to Java documentation for more details on this option.

iPlanet 6

Edit file jvm12.conf in directory <path-to-web-server-instance>/config

jvm.minHeapSize=XXX

jvm.maxHeapSize=XXX

note the values are in bytes, for example: 128MB is 134217728

Logging

If you are using Jrun, edit the file global.properties (found in the lib directory) and add the value ‘metrics’ to the property logging.loglevel as follows:

```
logging.loglevel=info,warning,error,metrics
```

This will cause a summary line to be written periodically to Jrun’s log file default-event.log. The log is useful for monitoring the number of active requests being sent to Windchill (Busy count) and the number of queued requests (Queued). During peak system activity it is expected that the number of queued requests will increase. Monitor the queue length during peak periods to decide if the server requires additional tuning.

```
06/04 17:02:09 metrics (JRun) (jcp+web) Heap=261696KB Listen=1
Idle=2 Queued=0 Busy=0 Total=3 Requests (count/total ms)=1/473
Delayed=0 TotalDelay=0 BytesIn=1306 BytesOut=0 Sessions (active/in
memory)=1/1
```

No such logging feature was found in tomcat.

If you are using Tomcat, a significant amount of information could be written to its servlet.log. You can modify server.xml (found in <TOMCAT_HOME/conf) to control the amount of information written to Tomcat's servlet.log. Tomcat has five logging levels to control the amount of information written into log files: FATAL, ERROR, WARNING, INFORMATION and DEBUG. If you don't specify a logging level, the default is set to INFORMATION. If you installed tomcat through Windchill CD's, its server.xml would have the following setting for servlet.log:

```
<Logger name="servlet_log"
        path="logs/servlet.log"
/>
```

The logging level can be changed to "FATAL" to get only error's in servlet.log as in the following example:

```
<Logger name="servlet_log"
        verbosityLevel = "FATAL"
        path="logs/servlet.log"
/>
```

In the example above, the logging level could also be set to ERROR, WARNING, INFORMATION or DEBUG.

Note: If you are running Tomcat 3.2.1 with the Ajp12 connector and iPlanet, ensure that LogLevel is set to ‘info’ in the tomcat configuration section in iPlanet’s obj.conf file. If you specified debug - as the tomcat installation instructions specify, tomcat will write enormous amounts of data to its nsapi.log file causing a major performance bottleneck.

8

Windchill Cache Mechanism

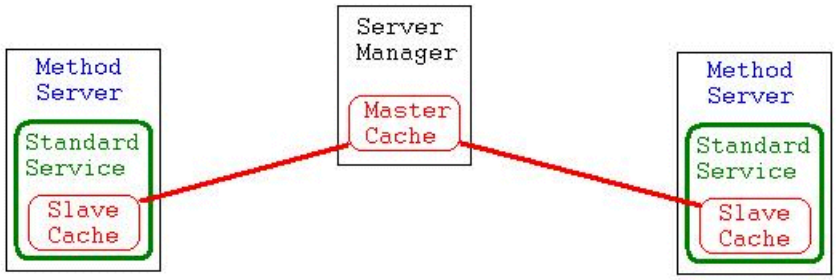
Topic **Page**

Caching....8-2
Implementing a Cache....8-2
CacheManager....8-6
ProjectLink Caches....8-6

Caching

Caching allows you to temporarily store an object in memory, allowing for quick retrieval. The Windchill software provides all the services and functionality necessary to create a generic caching service that remains in sync with database change by only caching objects which are known to accurately reflect the database contents.

The Windchill cache package addresses the maintenance and synchronization of multilevel caches where multiple cached objects (usually in separate virtual machines) have a master and slave relationship. Updates to a cache are pushed synchronously to the master cache, which asynchronously notifies other slave caches that an entry has been updated, as shown in the next figure:



To be effective, updates to cached objects must notify the cache so that master/slave caches may receive corresponding updates. It is the responsibility of the application to make this happen.

When a new entry is put into the cache, the entire object is sent to the master cache. When an update call is made, only the update is sent to the master. If the master has aged out its cache entry, it will attempt to retrieve the full object from the slave.

One thing to note about the behavior of the CacheManager is that when the master class receives a put of an object, a request is sent to all other slave caches to discard their version of the object.

Implementing a Cache

Implementing a cache is as simple as subclassing the CacheManager class. Caches are of fixed size and maintain most recently touched objects. Cache size can be configured with the property wt.cache.size. Otherwise the cache size defaults to 100. The Master cache is held by the ServerManager with the service name being the same as the created subclass. Cache name is configurable by overriding the appropriate method in the subclass. In a configuration where multiple Windchill systems exist, caches in different ServerManagers may have a master/slave relationship.

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

The main cache APIs are similar to those of a hash table get(key) and put(key,Object) are the primary calls for storing and retrieving objects from the cache. A special update method is provided that when overridden allows a client to implement a partial update of objects within a cache. For example, if the cache you implement caches hash tables, the update method would be designed to update a given hash table entry in the cache. This can be useful when dealing with large cached objects.

When implementing new services, look for repetitive database queries that could be avoided by reusing cached results. Services are responsible for guaranteeing consistency of cached results with the database. Proper maintenance of a cache involves listening for any deletes or updates of persistent data that might affect the cached results. Because events propagate synchronously within a transaction, try to defer any cache changes until the transactions are committed. Caching is further complicated by the fact that multiple servers may update the database. Relying on event listening is not sufficient because it will only notice deletes or updates originating in the same server.

Services can safely cache object identifiers and construct cached ObjectReference instances when they need to access the objects.

Windchill has an ObjectReference class that often acts as a foreign key from one object to a related object. Population of an ObjectReference target object requires database access. Many references are to administrative types of objects (for example, Principals, Projects, and Roles) that are read often, but seldom changed. Caching these references can have significant impact on performance.

The CachedObjectReference is a special ObjectReference that is backed by a memory cache in addition to the database. It has a fixed size MRU (most recently used) cache to limit memory consumption. Consistency with the database is maintained automatically when using this service. The cache reference service preserves ObjectReference semantics.

A local cache of actual targets and reference objects is maintained to satisfy requests for these objects. If the object is not found in the cache, it is queried from the database, cached, and returned to the requestor. Requests are satisfied in one of two ways, depending on the API that was used. Requests that use the getObject API are satisfied by cloning the object and returning the clone. The getReadOnlyObject API returns a reference to the object in the local cache itself. Code using the getReadOnlyObject API should not modify the object, because doing so contaminates the cache.

Maintaining Cache Integrity

In order to maintain cache integrity with the database cache implementations must do three additional things:

1. Subscribe to get relevant persistence service events (POST_MODIFY, and POST_DELETE).

2. Upon receipt of such events, subscribe with the transaction to capture the actual commits or rollbacks associated with these events.

3. Update the cache to reflect committed changes. The two subscription events are shown in the example that follows.

The general technique used to subscribe for persistence events is to override the performStartupProcess method of the StandardManager by putting subscription request in this method.

```
public void performStartupProcess()
   myListener = new MyListener() // subclass of
ServiceEventListener
   getManagerService().addEventListener(myListener,persistence
manager event key);
   // where persistence manager event keys are of the form
   //
PersistenceManagerEvent.generateEventKey(PersistenceManagerEvent.P
RE_STORE)
   }
```

Event listener classes are created by subclassing the ServiceEventListenerAdapter class and overriding the notifyVetoableEvent method. The event listener will receive event Object to test against and take action when appropriate. For example, set up with the current transaction to receive notification of commit or rollback, as follows.

```
public class myListener extends ServiceEventListenerAdapter() {
   public void notifyVetoableEvent (Object event) throws
WTException {
       if (!(event instanceof KeyedEvent)) {
          return;
       }
       KeyedEvent keyedEvent = (KeyedEvent) event;
       if (TRANS_VERBOSE) {
          System.out.println ("Inform:myListener:
notifyVetoableEvent,   Event: " + keyedEvent);
       }
       Object targetObject = keyedEvent.getEventTarget ();
       //
       Transaction.addTransactionListener(new
myTransactionListener());
   }
}
```

The transaction listener class must implement the interface TransactionListener. This class can then take appropriate action upon notification of a commit and generally ignore rollbacks.

Sample Cache Code:

```
// class for some Standard Service Manager
public class StandardSomeServiceManager extends StandardManager {
```

```
   MyCache myCache;
   // member class for service events
   class MyListener extends ServiceEventListenerAdapter {
      public void notifyVetoableEvent (Event e) {
         KeyedEvent keyedEvent = (KeyedEvent) e;
         Object targetObject = keyedEvent.getEventTarget ();
         //
         // check out event and object to see if we really care
what happens
         //
         // Set up with Transaction for commit
         if (keyedEvent.getEventType ().equals
(PersistenceManagerEvent.POST_DELETE)) {
         Transaction.addTransactionListener(new
Remover(targetObject));
         }
      }
   }
   // member class for transaction events
   class Remover implements TransactionListener {
      Object obj;
      public Remover (Object obj) {
         this.obj = obj;
      }
      public notifyCommit() {
         // get my cache and remove object
         getMyCache().remove(this.obj);
      }
      public notifyRollback() {
         // ignore
      }
   }
   public void performStartupProcess () {
      myListener = new MyListener() // subclass of
ServiceEventListenerAdapter
      // Set up to receive service events
      getManagerService().addEventListener(myListener,
         PersistenceManagerEvent.generateEventKey
(PersistenceManagerEvent.POST_DELETE))
   }
   public static MyCache getMyCache() {
      if (myCache == null) {
         try {
            return new MyCache();
         }
         catch (RemoteExcepiton re) { // do something }
      }
      return myCache;
   }
}
// class which implements the actual cache
public class MyCache extends CacheManager {
   public MyCache() {
      super();
   }
}
```

Using the Method Context Class

The MethodContext class can be used to cache information for the duration of a single remote method invocation (a method defined on a service interface). MethodContext extends Hashtable so it can hold arbitrary key/value pairs. Furthermore, the current MethodContext is easily obtained by calling wt.method.MethodContext.getContext(). This caching technique is useful when a top-level remote method is implemented by several submethods that execute the same database queries repeatedly. Using this approach, the query result can be put into the MethodContext one time and subsequently accessed by other methods when needed. Note that using the MethodContext to cache information does not require synchronization across method servers because the cached information only exists for the duration of a single method invocation.

CacheManager

The CacheManager of the wt.cache package acts as a signaling channel used to inform slave caches (caches in different method servers) that they should discard a given object if they hold it. It is not used to keep a set of caches in sync in terms of guaranteeing that all slave caches are equal and hold the same objects. The CacheManager is used with objects that have been deleted or modified and committed to persistent storage.

ProjectLink Caches

There are four caches:

1. The clientCacheLimit affects how many UI models we cache per user in the servlet engine. The model is cached in the session of the user. When a person goes to a page for the first time, the table model goes into the cache so that the next time that page needs to be rendered by either the user coming back to it from another page, or by invoking an action on the page such as delete/copy/paste/cut/.../, we will recall the cache and render without hitting the Method Server.

   Response times are greatly improved. The cache is smart to update itself with the result of the action that user's can invoke so that if a user picked the delete action, the model in the cache is smart enough to remove that row from its model.

   com.ptc.netmarkets.clientCacheLimit=5

2. The clientTabCacheLimit is the number of tab models cached in the servlet engine's user session, per user. These are pretty light weight so a higher/lower number will not effect memory too much. Each page has one tab model so it will cache the last 5 pages worth of tabs.

   com.ptc.netmarkets.clientTabCacheLimit=5

3. The clientCacheTimeOut affects how long a UI model cached in the servlet engine's user session will last before it is stale.

   com.ptc.netmarkets.clientCacheTimeOut=600000

4. The serverCacheLimit affects how many project models (the folder structure of folder/parts/docs) are cached in the Method server. This cache is shared among all users. These are a bit heavier weight so if there are memory constraints and there are thousands of projects, then this limit should be reduced.

   com.ptc.netmarkets.serverCacheLimit=1000

9

Tuning The Operating System

Properly tuning the operating system to deliver the best performance can have a great influence on how well Windchill performs. Operating system monitoring tools should be used to help determine bottlenecks. This section covers tuning tips for HP/UX , Windows NT, and Solaris.

Topic **Page**

Tuning Hp/Ux.....9-3
The Solaris Kernel.....9-3
Tuning Windows NT.....9-5

Tuning Hp/Ux

In general, to run Windchill on a Hewlett Packard system, you need the following software supplied by Hewlett Packard:

- HP-UX 11I or HP-UX 11.0 Extension Pack 9812 - If you are installing on an N4000, the minimum revision is HP-UX 11.0i Extension Pack 9905. All servers are supported on Extension Pack 9905, and it is best to use the latest revision whenever possible.
- Java Development Kit 1.3.1.
- Optional drivers to support specific I/O devices - For example, customers who use 100BaseT or Fibre Channel.

Configuring the hp-ux kernel

The HP-UX kernel should be configured to support the numerous processes and threads that will execute simultaneously when running Windchill. The `/usr/sbin/sam` tool is used to alter the machine and kernel configuration. The kernel configuration suggested below is intended to support running Windchill, HP-UX JVM 1.3, Oracle, and the Webserver from one server. This kernel configuration is intended to support hundreds of concurrent Windchill clients.

1. Select SAM > kernel-configuration > configurable-parameters.
2. Choose Actions from menu bar.
3. Select Apply tuned parameter set.
4. Select General OLTP/Database Monolithic System and apply
   - Modify the following parameters:dbc_min_pct and dbc_max_pct = percentage of main memory to get about 300Mb of buffer cache (e.g., 3% on a 10Gb system)
   - maxdsiz = 0xF0000000
   - maxfiles = 1024
   - maxssiz = 0x04000000
   - maxtsiz = 0x10000000
   - max_thread_proc = 2048
   - nkthread = 10000
   - ncallout = 10016
   - shmmax = 0x40000000 for 32-bit Oracle, 0xF0000000 for 64-bit Oracle
   - tcphashsz = 4096
5. Create a new kernel and reboot the system.

6. Make sure that you have at least as much swap as you have memory – ideally you should have swap size = 2 x main memory size. If you are limited in disk space and do not have enough for at least the same amount of swap as memory, use the swapmem_on kernel parameter by setting it to 1. Otherwise it should be left at 0.

HP maintains a useful web page at http://devresource.hp.com/JavaATC/JavaPerfTune which provides additional tips on Java Performance tuning.

Configuring Oracle Datafiles with Raw Logical Volumes

The performance of the Oracle database, in particular the BLOB tablespace, can be enhanced with the use of raw logical volumes and asynchronous I/O.

1. Add the asyncdsk device to the kernel and reboot
2. Create LVs using SAM with 10% more storage than planned for tablespaces
3. Change owner/group to $ORACLE_USER:dba for all rvols
4. Create links from the rlvols to the dbf files like:

   ln –s /dev/vgdb/rlvusers /oracle/dbs/users.dbf

5. To verify that your db is in async mode, using Glance to verify that the /dev/async file is open by the ora_dbw0_[$ORACLE_SID] process

Monitoring operating system and system resources

To get an idea of CPU utilization, operating system process management, and other global system resources, the HP-UX system activity reporter /usr/bin/sar is an effective and easily understood tool. For information on running sar refer to its man page.

Optional System Management products from Hewlett Packard include:

- HP GlancePlus/UX - for monitoring one system
- HP Measure Ware
- HP PerfView
- HP OpenView - for monitoring many network devices and systems

The Solaris Kernel

Tuning Solaris

The Solaris kernel should be configured to support the numerous processes and threads that will execute simultaneously when running Windchill. The file /etc/system is used to define `the` kernel configuration. The kernel configuration suggested below is intended to support running Windchill, Solaris 2.7, 2.8 JVM

1.3.1, Oracle, and Webserver from one server. This kernel configuration is intended to support hundreds of concurrent Windchill clients. These settings are meant for the server configuration only.

/etc/system:

- Set pt_cnt=1024
- Set npty=1024
- Set sadcnt=2048
- Set nautopush=1024
- Set dosynctodr=0
- Set shmsys:shminfo_shmmax=0xffffffff
- Set semsys:seminfo_semmns=600
- Set semsys:seminfo_semmnu=600
- Set symsys:seminfo_semmsl=200
- Set symsys:seminfo_semume=600
- Set rlim_fd_max=1024
- Set rlim_fd_cur=1024
- Set tcp:tcp_conn_hash_size=262144
- Set sq_max_size=0
- Set priority_paging=1(not Solaris 2.8)
- Set fastscan=65536
- Set maxpgio=65536

Further information on Java tuning is available at Sun's website: http://java.sun.com/.

Configuring Oracle datafiles with directio filesystem

The performance of the Oracle database, in particular the BLOB tablespace, can be enhanced with the use of either raw disk devices or filesystems with directio enabled. However, raw disk access can be administratively inconvenient. One alternative is to use a volume management tool such as Veritas to label and keep track of raw disk space.

UFS directio is a feature added to Solaris 2.6 to provide unbuffered access to a file in a normal file system. Significant performance improvements have been measured when placing the BLOB datafiles on filesystems mounted in directio mode.

For more information on directio refer to the man page for mount_ufs .

Further performance improvements may be gained by creating a striped filesystem for the BLOB datafiles.

Monitoring Operating System And System Resources

To get an idea of CPU utilization, operating system process management, and other global system resources, the Solaris system activity reporter /usr/bin/sar is an effective and easily understood tool. For information on running sar refer to its man page.

Tuning Windows NT

When using Windchill Release 6.2, make sure that your NT system has NT Service Pack 5 or higher installed. General tuning information for NT systems is available at the following URL:

http://msdn.microsoft.com/vstudio/downloads/scale/tuning.asp

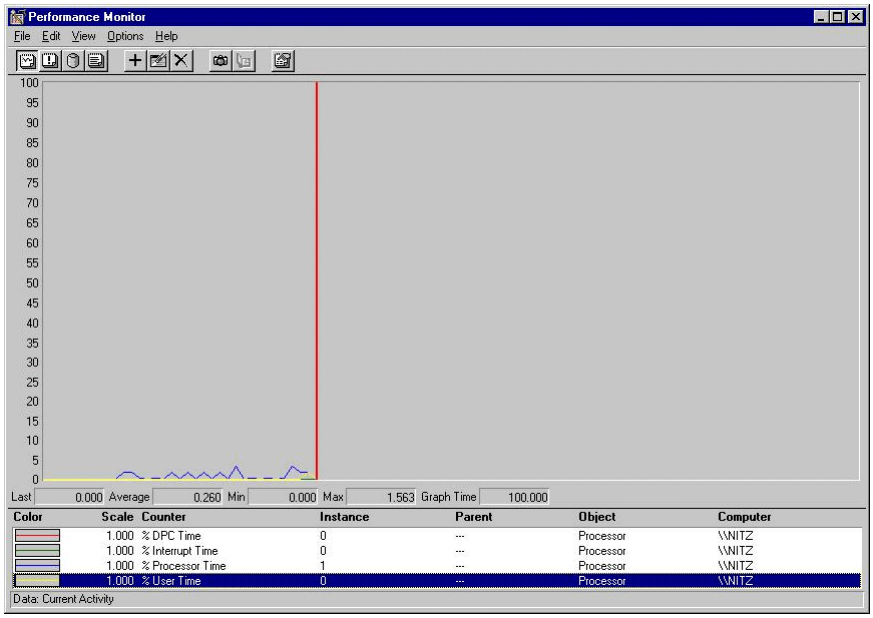
Windchill specific tuning information follows.

Monitoring System Resources

To get an idea of CPU utilization, operating system process management, and other global system resources, use the Performance Monitor and Task Manager tools that are provided with Windows NT.

Using performance monitor

Access Performance Monitor by selecting **Start > Programs > Administrative Tools > Performance Monitor**. You will see a screen that looks similar to this one:



You can customize the information that appears in the screen by adding different processes to the list.

To write the data to a file, do the following:

1. Select **View >log**.
2. Select **Edit >Add to log**.
3. Add all Processors, Network Interfaces, memory and paging file and click **Done**.
4. Select **Options >Log**
5. Select **Periodic Update** and enter an interval.
6. Enter a filename for the log and select **start log** – ensure the Status field shows “Collecting”.

When the test is complete, stop logging:

1. Select **Options >Log**.
2. Select **Stop Log**.

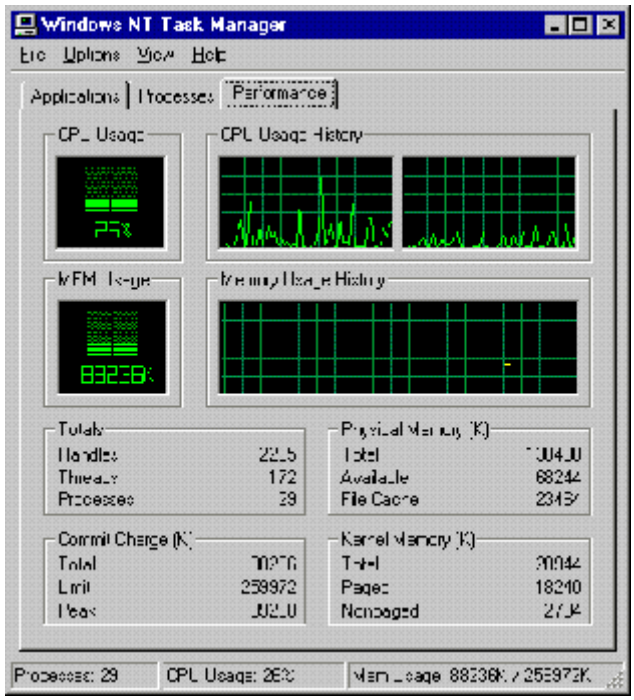
To view/export the collected data:

1. Select **Options > Data From**.
2. Select **Log File**, browse to the logfile, open it and select **OK.**
3. Select **View > Chart**.
4. Select **Edit >Add to Chart**.
5. Add %Processor time, bytes read/written, etc to chart.
6. Select **File > Export Chart**.

Enter file name and type (csv format is useful for importing to spreadsheet packages) and **Save.**

Using task manager

Access Task Manager by right clicking on the Taskbar and selecting **Task Manager** from the menu that appears. You will see a screen that looks similar to this one:

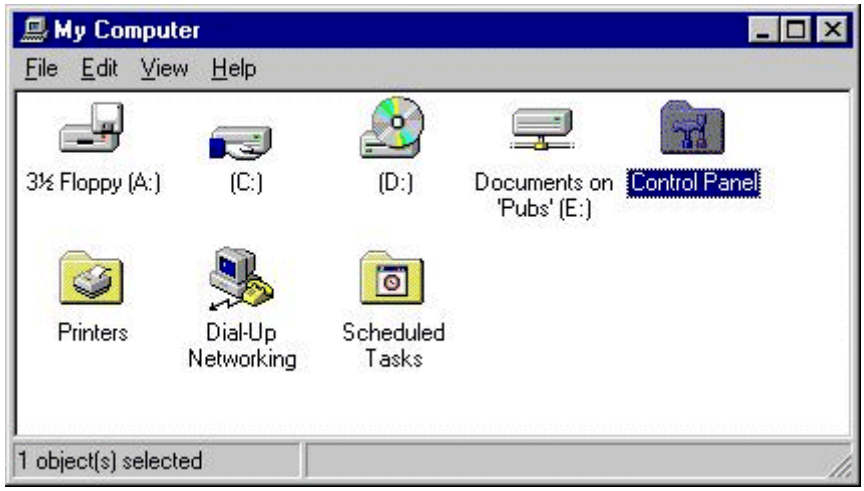


You can also view system performance, or see which applications are running by selecting the appropriate tab view.

Setting applications at equal priority

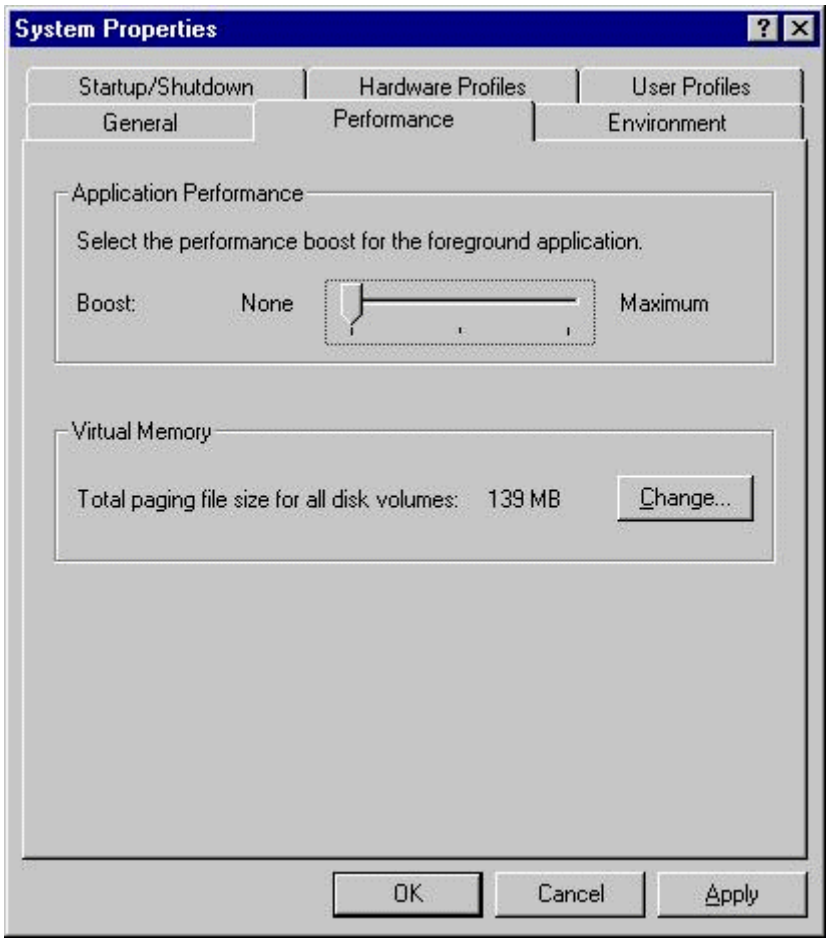
Do not boost performance for foreground applications. Instead, give all applications equal priority by setting the boost to **none** as shown below:

1. Click on the **My Computer** icon.
2. Click on the **Control Panel** icon.



3. Click on the **System** icon and select the **Performance** tab.

4. Slide the Boost selector to **None** and click **OK**. All applications will be set to use equal performance.



Setting Maximum Throughput for Network Applications

For an NT server, the network neighborhood setting should be set for maximum throughput for network applications. The default for this setting allows file sharing.

10

Troubleshooting

This chapter contains tips for troubleshooting common performance problems with your Windchill system. For more information on Java Performance Tuning refer to the web site at http://www.JavaPerformanceTuning.com/tips.shtml.

Topic **Page**

Performance Checklist 10-2
Analyzing Full Thread Dumps 10-6
Monitoring Method Server and ServerManager Memory Consumption 10-7
Resolving TCP Name Problems 10-8

Performance Checklist

To improve performance, check the disk, network, and CPU characteristics discussed in the following sub-sections.

Disk

Check that the disks containing the RDBMS BLOB tablespace and File Vaults are not overloaded. Use iostat or perfmon to monitor disk activity. Look for disks which are more than 5 per cent busy or show queued requests. You may need to stripe filesystems and tablespaces over multiple disks. Alternatively, the datafiles associated with the BLOBs tablespace may be placed on a raw device or directio filesystem.

If the Windchill Server has any NFS mounted directories check the disk performance of the host serving the mounted filesystems. Mounted File Vaults are common in clustered Windchill environments. Ensure that network bandwidth between the NFS server and the cluster is sufficient. A 10MB Ethernet will be overloaded easily in such and environment. Ideally use a switched 100MB full duplex connection.

Network

Check that the network has adequate bandwidth between the Oracle server and Windchill server or servers. Heavily utilized or misconfigured network interfaces can significantly degrade performance. One symptom of network problems is low CPU utilization when the server is under load. A rough estimate of network throughput can be made using ftp. In theory, a 10Mbit/sec network is capable transferring 1.25Mbytes per second. Similarly, a 100Mbit/sec network should transfer up to 12.5Mbytes per second. Use ftp in binary mode to transfer a several megabyte file between servers in the network. On completion ftp will report how many bytes were transferred per second. If the throughput reported by ftp is significantly different from the rating of the network, it is likely that the method servers will be starved of data.

Note that the performance of disks at either end of the transfer can affect the throughput reported by ftp.

Network bandwidth and latency plays a major role in the transfer of data over a Wide Area Network. In the case where remote clients are connecting to Windchill over dialup or low speed WAN connections data transfers can take many seconds to complete. For the duration of each such transfer a separate dedicated thread is required in the Method Server, (potentially also Servlet Engine and Web Server). With many simultaneous transfers over a WAN, transaction concurrency increases possibly causing a drop in transaction throughput due to contention for database connections.

The number of simultaneous HTTP requests can be configured in many Web Server and Servlet Engines. However, there is no distinction between requests originating from clients connecting via LAN and WAN. To minimize the number

of simultaneous WAN based requests you might want to consider implementing a network traffic management (Quality of Service or QoS) solution. Several hardware and software based products are available.

Windows Interval Setting and Upload Performance

Bulk data upload throughput, when the web server is on Windows, can in some cases be improved by altering the value of the Windows variable **TcpDelAckTicks**.

To learn about this variable, you can perform and internet search based on the string **rfc1122**, and visit the resulting URLs.

The following section, ProjectLink HTTP Bulk Data Upload, is a related topic that concerns ProjectLink and bulk data upload throughput between Netscape and Internet Explorer browsers on a Microsoft Operating System when the web server is on Unix.

ProjectLink HTTP Bulk Data Upload

There is a significant difference in HTTP bulk data upload throughput between Netscape and Internet Explorer browsers on a Microsoft Operating System when the web server is on Unix. Comparison of the two browsers' HTTP multipart form encoded POSTs have indicated that Internet Explorer may be limited by Microsoft's implementation of the Nagle (aka tinygram) algorithm. By waiting for partial packet acknowledgement (and not sending full segments when available), the Microsoft TCP driver severely limits data throughput.

To work around the Internet Explorer bottleneck it may be possible to reduce the delay imposed by the web server when acknowledging partial segments. On Solaris, use the ndd tool as follows:

(as root)

ndd /dev/tcp tcp_deferred_ack_interval

100

ndd -set /dev/tcp tcp_deferred_ack_interval 20

This first ndd command in the example displays the current delay in milliseconds, imposed by the TCP/IP driver, prior to acknowledging partially filled packets. The default value for tcp_deferred_ack_interval on Solaris 2.8 is 100 milliseconds. The second ndd command shows how to assign a value to the parameter. The minimum recommended value for the ack_interval is 5. Whilst there is no need to reboot the server after running ndd, you will need to ensure that the value is preserved following a reboot. This can be done by adding the 'ndd -set ...' command to the TCP/IP configuration script /etc/init.d/inetinit.

Refer to Appendix D for guidelines on network sizing.

CPU

If you find that CPU utilization is high yet throughput is low (or response times are poor), you should determine which processes are consuming CPU resource. On UNIX the ps command will list all active processes, such as /usr/ucb/ps aux will produce a list of runnable processes sorted by CPU utilization. On NT use the Task Manager to list all processes and sort on the column headed “CPU”.

Once you have found which processes are consuming the majority of CPU see if any processes are non-Windchill (not java, Web server, servlet engine, etc.) and investigate. Stop all non-critical programs and services.

If the CPU intensive processes are Java Virtual Machines there are a few things to check:

- Look at a supposedly quiet system to see what processes are running even when there's no input.
- Ensure that each JVM is using a jit compiler – ensure that the –nojit flag is not present in wt.properties and that no environment variable JAVA_COMPILER is defined. See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.
- Check that the Method Servers are running the latest supported HotSpot VM. Also check that the –Xnoclassgc is set.
- Ensure that sufficient heap space has been allocated to each Method Server. Add the –verbosegc flag on the wt.manager.cmd.MethodServer property to monitor the frequency and duration of garbage collection. See below for more information on monitoring memory consumption. See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.
- Determine which JVM has accumulated the most CPU time since launch (using ps or Task Manager). If the Background Method Server has consumed a large amount of CPU time when compared to the other Method Servers, use the Queue Administrator to check for failed process queue entries. Refer to the section Enabling Dedicated Queues and the *System Administrator’s Guide for Windchill Technology* for more information.
- Enable the access.csv file as described on page 1-4. Restart the Method Servers and monitor which transaction accounts for the most elapsed time or transactions that are being frequently invoked.
- Check that the Query Limit property wt.pom.queryLimit has been set. See page 3-9. See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.
- Check all log files for excessive debug statements and/or stack traces.
- In the Method Server log file find all the MethodSummary lines. This can be simplified with either grep (Unix) or find (NT). Using the summary lines

determine the level of client activity in terms of number of calls, average duration and average active calls.

- If the number of active calls is consistently above the number of database connections AND the system utilization (CPU, memory, etc.) shows spare capacity, either:

  a. Increase the number of database connections available to each Method Server in db.properties (wt.pom.maxDbConnections – default is 5)

  b. Configure additional Method Servers. If there are multiple CPUs installed on the server system, you might increase the number of Method Servers by increasing property wt.manager.monitor.start.MethodServer

     See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

  In all cases you should monitor the effect of the change on CPU utilization and system throughput. If neither change improves performance you may need to implement or adjust the servlet engine throttle on simultaneous client requests. Determine the ratio of HTTP vs RMI transactions being processed. This can be found from the access.csv file or the web server log file. If the number of HTTP requests is high consider lowering the throttle in the servlet engine.

- Monitor cache efficiency. Refer to the section on Measuring Performance for descriptions of the various data caches maintained in Windchill. Pay particular attention to the JDBC statement cache.

- Blobs vs File Vault. Benchmarks have shown that multi-user workloads involving content stored in File Vaults perform approximately 25% better than content stored in BLOBs. However, BLOB performance may be improved by use the use of raw logical volumes or directio.

- Monitor the number of clients connecting over dialup or slow WAN connections. Examine the web server log file to find a list of all IP addresses which have recently connected to the web server and then measure the network latency between the server and client using ‘ping’.

- Monitor transaction duration. Excessive duration might indicate that one or more users are requesting enormous result sets. Generate a thread dump as described below to help find the source of the ‘run away’ query. Refer to the Advanced Query section of the *Application Developer’s Guide for Windchill Technology* for information on limiting the size of result sets.

- If you are running multiple Method Servers, check that Method Server load balancing is configured correctly.

- Use the ‘top’ command or perfmon to determine the amount of time spent in system calls or blocked waiting for io. Trace the VM’s system calls and network i/o on Solaris using truss, strace and snoop. See the man pages for more information.

If the CPU intensive process is Oracle then refer to page 6-2 for hints on Oracle performance tuning.

If none of these methods work, you should generate a thread dump to see if any threads are deadlocked

The Aphelion LDAP server on Solaris 2.8 has been observed to utilize as much as 70% of the CPU under peak load. The high utilization may be mitigated using pbind to assign the 'lde' process to a specific CPU.

On Solaris, the RetrievalWare 6.8 web front end process 'cqfe' allocates additional memory per session. In a benchmark simulating 1000 active users the cqfe process' memory requirement has been seen to grow to several hundred megabytes and cause excessive virtual memory paging. The work around is to periodically restart the cqfe process.

Analyzing Full Thread Dumps

If a system performance monitor indicates high level of CPU consumption, a thread dump from the offending problem will help with troubleshooting.

To record a thread dump on Solaris or HP-UX where the method servers are running in terminal windows (xterm), use the following steps:

1. Type ps -ef | grep java at a command line to find the process id (pid).
2. Once you know the process id, type kill –3 pid to kill the process, where pid is the process id number.

If the method servers are not configured to run in xterm windows you need to redirect the ‘standard error’ of the shell running the method servers. This can be done by modifying the `wt.manager.cmd.MethodServer` property to launch the method server in a shell script. See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

```
wt.manager.cmd.MethodServer=/opt/windchill/scripts/launch.ksh
```

Where the launch.ksh script is simply:

```
#!/bin/ksh
/opt/java1.2/jre/bin/java –server -Xms128m -Xmx256m -Xnoclassgc
wt.method.MethodServerMain > /opt/ptc/threads.log 2>&1
```

When the kill –3 is sent to the jvm, the thread dump will be written to the file /opt/ptc/threads.log (along with any other System.stderr errors which would normally appear in a method server window”).

On NT a thread dump can be generated by clicking the “X” icon in the upper right corner of the command window in which the method server is running.

Monitoring Method Server and ServerManager Memory Consumption

A performance degradation that is relieved by restarting the Windchill server could indicate a slow memory leak. As the available heap size gets smaller, more time is spent garbage collecting. Large search results consume CPU and memory when building Java objects out of database query results. Configuring heap size is important to Windchill servers to contain memory leaks and maintain performance.

Monitor virtual memory utilization. A large amount of spare RAM or low paging activity could suggest that additional space could be allocated to the Java heap.

To monitor heap space utilization you can either add the –verbose:gc flag to all JVMs running or use the following two utility methods:

wt.method.RemoteMethodServer

From a command line, execute the class, java wt.method.RemoteMethodServer, to check on the heap size and available memory while the method servers are running.

The following extract shows example output:

```
Start date             = Fri Jun 01 13:29:27 CDT 2001
Total RMI invoke calls = 155
Total method contexts  = 156
Active method contexts = 0
Current free memory    = 98282168 bytes
Current total memory   = 133955584 bytes
RMI client sockets     = 3
RMI bytes in           = 1959595
RMI bytes out          = 2780382
Database connections   = 5
```

The output shows that the VM heap size is set at 128MB, of which approximately 98MB are free. The output will be repeated for every running Method Server showing its corresponding metrics.

The calls can be automated to measure heap usage over time. It is open to question how much free space should be available at any time. The rule of thumb states that 60-75% of total memory should be free following a full garbage collection.

wt.manager.RemoteServerManager

From a command line, execute the class, java wt.method.RemoteServerManager, to check on the heap size and available memory while the Server Manager is running.

The following extract shows the output from wt.manager.RemoteServerManager. Note that the default maximum heap size for a JVM is 16MB.

```
Number of registerServer calls   = 10
```

```
Number of getServer calls        = 9
Number of reportDeadServer calls = 0
Number of getInfo calls          = 1
Number of servers launched       = 9
Current free memory              = 3247032 bytes
Current totol memory             = 4128768 bytes
RMI client sockets               = 1
RMI bytes in                     = 17306
RMI bytes out                    = 10735
```

The output shows that the VM heap size is set at 4MB, of which approximately 3MB are free. Note that 1MB is the default initial heap size so we can infer that the heap has grown from 1MB to 4MB. Once the heap has grown to 16MB no further expansion is possible so if you observe a ServerManager with low free space and 16MB of total memory you should specify a larger maximum heap space for property wt.manager.cmd.ServerManager. See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

Resolving TCP Name Problems

Problems with TCP name resolution can appear as serious performance problems. Run the command java wt.tools.hostname.TestHostName.

This should result in quick time look-ups of no more than 400 milliseconds and should be less than 20 milliseconds. Any results that are consistently several seconds long indicate name look-up problems.

11

Workflow Performance Tuning

Topic **Page**

Queue Pooling 11-2
Dedicated Queues 11-3
Enabling Dedicated Queues 11-3
Combining Queue Pooling and Dedicated Queues 11-6
wt.queue.execEntriesCount 11-10

Queue Pooling

There are two Windchill queues that are used primarily for processing Workflow tasks:

1. *WfuserWork* queue for processing tasks related to Workflow robots
2. *WfPropagation* queue for processing tasks related to Workflow propagation

Both the queues are FIFO queues and they each have a single thread processing the jobs in the queue. This means that a single long running workflow task can block either queue.

To alleviate this problem, a mechanism called *queue pooling* was introduced in Windchill R5.1 DSU7. Queue pooling sets up a pool of *WfuserWork* and *WfPropagation* queues. The pools for these queues can be set through the following properties in *wt.properties*. The default value for these properties is 1. See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

wt.workflow.engine.userWorkPoolSize

wt.workflow.engine.propagationPoolSize

Note: The total number of process queues (*WfuserWork* and *WfPropagation* are process queues) that Windchill has, should not exceed the hard value defined in the following property:

wt.queue.max.processQueues

If this queue limit is reached, any process that tries to create a new queue will throw an exception and fail to start. Adhering to this hard limit is very important.

The default value for *wt.queue.max.processQueues* is 12.

Once the queue pools are defined, tasks are put into these queues in a circular manner, starting with the first and continuing through the queue and then starting over with the first again. Since each queue has a thread associated with it, a single long running workflow process will block only its own queue while allowing the processing to continue in other process queues in the corresponding pool.

Note: The capacity of the server on which Windchill is running should also be kept in mind while configuring these queue pools. If the server is bogged down and cannot handle more threads, adding more queues will not result in any performance improvement.

Dedicated Queues

Dedicated queues were introduced in R4.0. Queue pooling was introduced in R5.0 DSU 7 as a substitute for dedicated queues. So wherever possible, queue pooling should be used in lieu of dedicated queues. Current versions of Windchill continue to support dedicated queues for compatibility reasons.

Dedicated queues are queues that are tied to specific workflow templates. This might be helpful in scenarios where particular workflow templates are being used heavily or are using complex functionality resulting in generation of long running queue entries.

In order to enforce dedicated queue processing on specific workflow templates, first you have to enable dedicated queues in *wt.properties* and then set “has dedicated queue” flag to true in the corresponding workflow template.

Enabling Dedicated Queues

Enable dedicated queue processing in *wt.properties* by setting the following property value as appropriate. See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

wt.workflow.engine.dedicatedQueueMode

The following are possible values for this property:

none (default) – dedicated queue processing disabled

userWork – a dedicated *WfuserWork* queue for the Workflow templates that have the dedicated queue processing flag enabled

propagation – a dedicated *WfPropagation* queue for Workflow templates that have the dedicated queue processing flag enabled

both – a dedicated *WfuserWork* queue and a dedicated *WfPropagation* queue for Workflow templates that have the dedicated queue processing flag enabled

Note: After setting this property in *wt.properties*, Method Servers should be restarted for this property to take effect.

Setting “has dedicated queue” Flag to True

To set the “has dedicated queue” flag to true for the workflow template that needs the dedicated queues, do the following:

1. From a Windchill command prompt, launch the workflow *definer* tool:

Note: For the *windchill* command to work, your WT_HOME environment variable should be set to appropriate value; *windchill* command is located in $WT_HOME/bin directory.

windchill wt.clients.workflow.definer.SetConfiguration

2. You will be prompted for a login and so if needed, your *DISPLAY* environment variable should be set to appropriate value. You should login with your *administrator* id.
   a. Locate and record the serial number of the workflow template for which you are enabling dedicated queue processing.
   b. Login to windchill using the same administrator id that you used to login to the workflow *definer* tool.
3. Go to the *Process Administrator* page and click on **Workflow Administrator**.
4. Check out the workflow template for which you are enabling dedicated queue processing.
5. Refresh the *definer* tool list
   - Pick option **5**

Note: There should be a new entry in the list (corresponding to the checked-out version of the template) in addition to the original entry (corresponding to the original template).

6. Select the checked out entry:
   - Pick option **1**.

Pick the new entry (corresponding to the checked out version). You can make sure that you are picking the new entry by ensuring that the serial number is different from the one recorded earlier.

7. The checked out version of the template should now be selected. Select the **edit** option.
   - Pick option **2**

8. Switch the value of “Has Dedicated Queue” from false to true.
    - Pick option **28**
9. **Save** and return.
    - Pick option **31**

Check the Method Server log to ensure that the checked-out version of the template was edited. If the original version was edited by mistake, an exception similar to the following will appear in MethodServer.log file:

wt.fc.PersistenceManagerFwd.save from 132.253.9.68: (wt.vc.wip.wipResource/5) wt.vc.wip.WorkInProgressException: The object is the original version of a working copy and cannot be modified.

10. From the *Workflow Administrator*, check the template back in.
11. Refresh the *definer* tool list.
    - Pick option **5**

You should see that the number specifying workflow template configuration indicates that the template uses dedicated queue (there should be an 800 prior to the standard 1000).

- DedicatedQueueTemplate – (8001000)
- StandardQueueTemplate – (1000)

Setting Dedicated Queue Processing for all Newly Created Workflow Templates

If it is expected that every workflow process will require its own queue, then set the following wt.properties property value to true. See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

wt.workflow.engine.dedicatedQueue=true

Note: All workflow templates that are created after this property has been set to *true* will have “Has Dedicated Queue” flag set to *true*. Setting this property to *true* will not retroactively set the “Has Dedicated Queue” flag to *true* for all the workflow template that were already in existence. You will have to use the *definer* tool as described above to do that.

Setting Dedicated Queue Processing per Workflow Process Initiated From a Particular Workflow Template

Instead of forcing all the processes of a particular template to use the same dedicated queue, it is possible to configure Windchill to spawn a new queue for every instance of a process initiated from a template that has the "Has Dedicated Queue" flag set to *true*. All the new queues that are initiated are process queues. In order to achieve this, set the following property in wt.property file to *true*. See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

wt.workflow.engine.dedicatedQueuePerProcess=true

Note: This property is set globally; there is no way to set this property on a per template basis. Therefore special attention has to be paid to the hard limit on process queues; as was previously stated, once this limit is reached, any new processes will throw an exception and fail to start. Given that the number of Workflow processes can change on the fly, special caution must be exercised in use of this property.

Combining Queue Pooling and Dedicated Queues

If needed, a combination of queue pooling and dedicated queues could be used. The following example illustrates this. See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files. Refer to the *Windchill System Administrator's Guide* for more information on finding the status of background method server queues.

- Increase the maximum number of queues in *wt.properties*:

  wt.queue.max.processQueues=50

- Set the *wt.properties* required to enable queue pooling:

  wt.workflow.engine.userWorkPoolSize=10

  wt.workflow.engine.propagationPoolSize=10

After this change, workflow processes will push jobs into *WfUserWorkQueues* or *WfPropagationQueues* in a circular manner, starting with the first and continuing through the queue and then starting over with the first again. Since each queue has its own thread for processing its jobs, this change would create parallel processing of workflow tasks.

1. Set the *wt.properties* value required to enable dedicated queues for both *WfUserWork* and *WfPropagation* queues:

   wt.workflow.engine.dedicatedQueueMode=both

2. Restart the Method Servers.

   Use the *definer* tool to enable dedicated queues for the particular template (*IntensiveTemplate)*

   - windchill wt.clients.workflow.definer.SetConfiguration

3. Enter the *administrator* login for Windchill.
4. Locate *IntensiveTemplate* in the list. In our case, it has serial # 18.

```
15. Example Review - (1000)
16. Change Activity Process - (802d000)
17. Change Analysis Process - (802d000)
18. IntensiveTemplate - (1000)
```

5. Login to *Workflow Administrator* and check out **IntensiveTemplate**
6. Refresh the template list in the *definer* tool by selecting option **5**.

```
Set template configuration
--------------------------
1. Select process template
2. Show/edit process template configuration
3. Select activity template
4. Show/edit activity template configuration
5. Refresh templates
6. Exit
```

7. Locate *IntensiveTemplate* corresponding to the checked out version in the *definer* tool. In our case, it is the template with the serial # 19 (Note that we recorded the serial # of the original template to be 18 above; so serial # 18 cannot be our checked out version)

```
15. Example Review - (1000)
16. Change Activity Process - (802d000)
17. Change Analysis Process - (802d000)
18. IntensiveTemplate - (1000)
19. IntensiveTemplate - (1000)
```

8. Select the checked out *IntensiveTemplate* (serial # 19) by using option **1**.

```
                Set template configuration
                  -------------------------
1. Select process template
2. Show/edit process template configuration
3. Select activity template
4. Show/edit activity template configuration
5. Refresh templates
6. Exit

          >>> Choose an option: 1

         Select process template
         -----------------------

          >>> Choose an option: 19
```

9. Edit the template by choosing option **2**:

```
                Set template configuration
                  -------------------------
    1. Select process template
    2. Show/edit process template configuration
    3. Select activity template
    4. Show/edit activity template configuration
    5. Refresh templates
    6. Exit
```

10. Select option 28 to enable dedicated queue processing for *IntensiveTemplate*. It should toggle the value from *false* to *true*.

```
    26. Notify on approaching deadline:      false
    27. Asynchronous execution:              false
    28. Has dedicated queue (process):       true
    29. Ignore unresolved role (assign act): false
```

11. Save template and return by using option 31.

```
    30. Save template
    31. Save template and return
    32. Return (looses changes since last save)
```

12. Refresh the template list by using option 5. You should see that the checked out version of intensive template (serial # 19 in our case) has been updated with a prefix of 800 before the original 1000 to indicate that dedicated queue processing has been enabled.

```
       15. Example Review - (1000)
       16. Change Activity Process - (802d000)
       17. Change Analysis Process - (802d000)
       18. IntensiveTemplate - (1000)
       >> 19. IntensiveTemplate - (8001000)
```

13. From the *Workflow Administrator*, check the template back in.
14. If you refresh the list again in the *definer* tool (using option 5), you should see that the checked out *IntensiveTemplate* (serial # 19) goes away and the original *IntensiveTemplate* (serial # 18) shows the updated value of 8001000 indicating that the "Has Dedicate Queue" flag has been set to true for this template.

```
15. Example Review - (1000)
16. Change Activity Process - (802d000)
17. Change Analysis Process - (802d000)
18. IntensiveTemplate - (8001000)
```

At this point, any workflows initiated from the *IntensiveTemplate* will use their own single *WfUserWork* and a single *WfPropagation* queue dedicated only to *IntensiveTemplate* tasks. All other workflow tasks will be entered into their respective queue pools (1 queue pool of 10 *WfUserWork* queues and 1 queue pool of 10 *WfPropagation* queues in this example) in a circular manner, starting with the first and continuing through the queue and then starting over with the first again. So effectively, 10 userWork tasks and 10 propagation tasks could be processed in parallel. We will not be losing any entries because the hard limit on total process queues is 50 (*wt.queue.max.processQueues=50)* and we won't be approaching the hard limit in this example configuration.

Checking the Status of Background Method Server Queues

Please refer to the *Windchill System Administrator's Guide* for more information on finding the status of background method server queues.

wt.queue.execEntriesCount

The wt.queue.execEntriesCount property can be added to the wt.properties file. Its value specifies the number of queue entries returned for processing every time a call is made to the Windchill database. The default value is 6. Increasing this value results in fewer database calls and improves queue performance, especially in situations where the queue load is heavy. To improve queue throughput, test the effect of increasing the value by 100% or 200%.

See the appendix, Editing Windchill Properties Files, for a description of how you edit Windchill properties files.

A

Useful SQL Queries

These SQL queries can help you to optimize your Oracle database.

Topic **Page**

FILE Name: explain_display.sql A-2
FILE Name: explain_create.sql A-3
find_all_index_info.sql A-4
index_rebuild.sql A-5

FILE Name: explain_display.sql

```
REM Author: Vishal Arora, Apr 28-1999

REM Name: explain_display.sql

REM Purpose: Does structured display of EXPLAIN PLAN

REM Usage:   @explain_display <explain_id>


column query_plan format A54

REM column object_node format A10

column position format 990 heading "POS"

column cost format 990 heading "COST"

column cardinality format 999,990 heading "ROWS"

REM column bytes format 999,990 heading "BYTES"


SELECT

   DECODE(id, 0, '', LPAD(' ',2*(level-1))||level||'.'||position||
' ')

   ||operation

   ||decode(options,null,'',' ('||options||')')

   ||' '||object_name||' '||other_tag

   ||' '||decode(id, 0, 'Cost = '||position) query_plan,

   position,

   cost,

   cardinality

FROM plan_table

START WITH id = 0

AND statement_id = '&&1'

CONNECT BY prior id = parent_id

AND statement_id = '&&1';
```

FILE Name: explain_create.sql

```
REM Author:  Vishal Arora,Apr 28-1999
REM Name:    explain_create.sql
REM Purpose: Create PLAN_TABLE for explain plan
REM Usage:   @explain_create

drop table plan_table;
set termout off
create table plan_table (
    STATEMENT_ID     VARCHAR2(30)
, TIMESTAMP        DATE
, REMARKS          VARCHAR2(80)
, OPERATION        VARCHAR2(30)
, OPTIONS          VARCHAR2(30)
, OBJECT_NODE      VARCHAR2(128)
, OBJECT_OWNER     VARCHAR2(30)
, OBJECT_NAME      VARCHAR2(30)
, OBJECT_INSTANCE  NUMBER(38)
, OBJECT_TYPE      VARCHAR2(30)
, OPTIMIZER        VARCHAR2(255)
, SEARCH_COLUMNS   NUMBER(38)
, ID               NUMBER(38)
, PARENT_ID        NUMBER(38)
, POSITION         NUMBER(38)
, COST             NUMBER(38)
, CARDINALITY      NUMBER(38)
, BYTES            NUMBER(38)
, OTHER_TAG        VARCHAR2(255)
, OTHER            LONG
);
set termout on
```

find_all_index_info.sql

```
create or replace procedure list_all_indexes as
v_index_name varchar2(30);
v_table_name varchar2(30);
v_column_name varchar2(50);
v_column_position number;
v_column_list varchar2(400);

cursor find_index is
    select distinct index_name,table_name
    from user_ind_columns
    where index_name not like 'PK%'
        and index_name not like 'SYS%'
    order by table_name,index_name;

cursor find_column_info(v_table varchar2, v_index varchar2) is
    select column_name,column_position
    from user_ind_columns
    where index_name = v_index and
        table_name = v_table
    order by column_position;
begin
    dbms_output.enable(1000000);
    open find_index;
    loop
        fetch find_index into v_index_name,v_table_name;
        exit when find_index%notfound or find_index%notfound is
null;

        FOR v_ind_col IN find_column_info(v_table_name,v_index_name)
LOOP
           if (v_ind_col.column_position = 1) then
               v_column_list := v_ind_col.column_name;
```

```
          else
             v_column_list := v_column_list||','||
v_ind_col.column_name;
          end if;
   END LOOP;
```

index_rebuild.sql

```
/* Script : Index_rebuild.sql                                          */

/* This script is used as part of Reorganizing the Indexes to a Different
tablespace.*/

/* For Eg.. To isolate all the Windchill Index objects from a tablespace called
*/

/* USERS to a new tablespace called INDX for a given schema Owner
WTADMIN.          */

/* This script is designed  to rebuild indexes belonging to one schema owner at
time.*/

/* You need to login as 'SYSTEM' to run this script.                   */


accept Schema_owner    prompt 'Enter the name of the Owner who owns the
Index Objects :'

accept tablespace_new  prompt 'Enter the new tablespace name to move all the
indexes : '


set termout off

set heading off

set feedback off

set pages 0


set verify off

spool INDREBUILD.SQL


select 'ALTER INDEX ' || owner || '.' ||index_name || ' REBUILD TABLESPACE
&&tablespace_new; ' from

DBA_INDEXES
```

```
where owner = '&&Schema_owner';

spool off

@INDREBUILD.SQL
```

B

Basic Oracle Administration

This appendix provides a minimum guideline of administrative operations, which should be performed in order to maintain and safeguard an Oracle database.

Topic **Page**

Reference documents B-1
Understanding Space Use in Oracle B-1
General Oracle Tuning B-2
Example Commands B-3

Reference documents

The following URLs are extracted from Oracle9i database online documentation (Release 2(9.2), as parts of specific information relating to Statspack and the *Windchill Performance Tuning Guide* reference.

- Oracle Statspack tool

  http://download-west.oracle.com/docs/cd/B10501_01/server.920/a96533/statspac.htm#22560

- Oracle9i Database Performance Tuning Guide and Reference

  http://download-west.oracle.com/docs/cd/B10501_01/server.920/a96533/toc.htm

Understanding Space Use in Oracle

Oracle logical objects are items such as tables, indexes, and rollback segments. These objects are stored in a hierarchical structure, which is shown in order from largest to smallest:

Tablespaces > Datafiles > Segments > Extents > Blocks

An easier way to think of this hierarchy is to use the following analogy based on filing cabinets:

- Database = Multiple filing cabinets
- Tablespaces = Individual filing cabinets
- Datafiles = Drawers in filing cabinets
- Segments = Folders in the drawers
- Extents = Paper in the folders
- Blocks = Units that comprise the paper
- Data = Information written on the paper

All objects within Oracle are stored in tablespaces, and each tablespace is stored in one or more datafiles in the operating system. Each tablespace also contains one or more segments made up of multiple extents. In general, each object grows within its own single segment. Object growth is dependent on the parameters: initial_extent, next_extent, pct_free, max_extents and pct_increase, which are set at the creation time of the object.

Extents are groups of blocks allocated by an object that needs more space. Each object is created using the initial_extent parameter. The next_extent and the pct_increase parameters determine how an object grows. The next_extent parameter is the size of the next extent that will be allocated. If a pct_increase has been specified, each subsequent extent allocated will be a certain percentage larger than the last extent. For example, if the next_extent is 10K and the pct_increase is 50, the third extent allocated will be 15K, the fourth 22.5K, and so on. Therefore, if the pct_increase is too large, eventually the extents being allocated could be hundreds of megabytes. This will lead to wasted datafile space and difficulty in allocating additional extents when necessary because their size is approaching the size of the datafiles themselves. Extents can only be allocated within existing datafiles. The max_extents parameter is the maximum number of extents that can be allocated for any given object. For example, the Oracle default for max_extents is 121, so if an object tries to allocate the 122nd extent, the allocation will fail and an error message will occur.

General Oracle Tuning

The document *Performance Tuning + Client/Server Consideration* provides a good overview of what has to be done in order to tune Oracle. Search for the document from the following URL:

http://www.orafaq.org/faqmain.htm

For more information, select **White papers > Oracle DBA Topics > 1**

Example Commands

```
ALTER TABLESPACE <tablespace object resides in> ADD DATAFILE <full
absolute path for the datafile that will be created> SIZE
```

For example, to add a datafile that is 100 MB and located in the c:\datefile directory to the TEMP tablespace, the command is the following:

```
Alter tablespace temp add datafile 'c:\datafile\temp_2.dbf' size
100M;
```

ALTER DATABASE DATAFILE 'full path name file is residing' **resize <K or M>**

The new size will be the total datafile size, not just the amount of space that will be added to it. For example, to add 100 MB to the temp_2.dbf datafile that is already 200 MB, the command is the following:

```
alter database datafile 'c:\datafile\temp_2.dbf' resize 300M;
```

C

Oracle Database Sizing for Windchill

This chapter describes the formula for use in determining a target size for your database.

Topic **Page**

Information About PTC Setup C-2
Sizing the Database C-3

Information About PTC Setup

The following table provides information on the configuration automatically established through PTC setup when an installer selects a small, medium or large database on configuration. The table provides information on disk space requirements, memory requirements, and the number of CPUs required for Oracle. The sizing information presented is for Oracle only and does not take the disk space required for the Windchill product files or the memory and CPU requirements to run the rest of the Windchill product into account.

The table calls out separate sizing based on whether you choose to use the external file vault option or to have all your data reside in Oracle tables.

The sizes presented are initial database sizes. The small figure is about twice the size of the Windchill demonstration database and should be adequate for very small pilots. The database sizes for both medium and large are conservative and take into account that most installations do not want to pre-allocate large amounts of disk space, therefore, an initial size is given. The table below does not include sizes for Oracle Executables. That data is shown as part of Load point.

	Blobs/Internal File vault	**External File vault**
Small / Demo	2GB (Disk) 512MB (Memory) CPUs = 1	2GB (Disk) – Oracle 1.4GB and File Vault 600 MB 512MB (Memory) CPUs = 1
Medium	3GB (Disk) 700MB (Memory) CPUs = 2	3G (Disk) – Oracle 2GB and File Vault 1GB 700M (Memory) CPUs = 2
Large	5GB (Disk) 1GB (Memory) CPUs = 4	5GB (Disk) – Oracle 3GB and File Vault 2 GB 1GB (Memory) CPUs = 4

Sizing the Database

The following formula can be used to give you an approximation of the target database size for your implementation when all data has been loaded. A more detailed example is presented at the end of this discussion of sizing the database

Load point + Content Data + Metadata = Total Database Size

Load Point

Load point disk space = 2.5GB. This constant represents the amount of disk space required for Oracle home and Oracle executables.

Content Data

[(Number of parts) + (Number of documents)] * (average size of all content in Megabytes)

In the detailed example at the end of this discussion of sizing the database, the value 0.925 is assumed for the average size of all content in Megabytes.

The value calculated for total content will be applicable for either an external file vault or content stored in Oracle.

Metadata

Metadata = (X+Y) * 0.8 * C

X = Number of parts * average number of iterations

Y = Number of documents or change objects * average number of iterations

C = (average size of all content in megabytes)

Example

This example gives a more precise method of executing the approximation described earlier and refers to a realistic set of values.

Assume the following example for a site that needs to size a database that after an initial legacy load will contain a total of 174,000 Windchill business objects. The characteristics of the data for this site include the following:

- 14,000 parts
- 160,000 documents
- 850K for documents associated content
- 1 M for part associated content

Disk Space Calculation Example

Load point + Content Data + Metadata = Total Disk Space Required

2.5GB + 153GB + 129GB = 284.5 gigabytes of total disk space required

The following sub-sections show the calculations that develop the numbers that sum to 284.5GB.

Content Calculation

(Number of Parts * Average part content) + (Number of documents * Average document content) + (Total Windchill Business Objects / 50 gigabytes)

(14,000 * 1) + (160,000 * 0.850) + (174,000 / 50 GB) = 153GB

The value 174,000 given for "Total Windchill Business Objects" is the sum of the numbers of parts and documents.

Metadata Calculation

[(Number of parts) + (Number of documents)] * 0.8 * (average size of all content)

Metadata = (174,000) * 0.8 * (0.925) = 129GB

Total Space Requirement for BLOB

Based on the above disk space calculation, we can arrive at the total space requirement for the BLOB.

Blobs / Internal File Vault	External File Vault
Disk (284.5GB)	Disk (284.5GB) composed of: Oracle - 129GB File Vault - 153GB Load Point 2.5GB

Note: Once the database has been running in pre-production Windchill environment or goes to production, it is essential to monitor the growth of the database. Using the various resources provided by Oracle you could determine whether there is enough space currently for any fast growing objects in the database to extend.

D

Windchill Network Sizing Guidelines

This section proposes a worksheet for calculating the required bandwidth based upon anticipated workload at a customer site. The workload used by R&D for bandwidth testing comprises PTC's Windmill database and GSO workload definition (as used at the Sun and HP Sizing Tests).

This worksheet is intended as a guidance only and can help you identify where and when you will need bandwidth; it is not intended to be a set of empirical rules.

Before proposing the methodology, there are some of observations of the workload tested in R&D: The Windchill HTML portion of the HTTP traffic was typically between 13-15 %. The remaining 85-87% comprised document content downloaded to the client. The RMI component was very low – perhaps unrealistically so. However, the average data transfer size associated with an RMI transaction is approximately 50KB and the average Windchill HTML page size was 16 KB.

Based on the above observations, the following worksheet calculates the expected hourly bandwidth and suggests a corresponding network configuration:

1. Estimate the total number of HTTP transactions per hour (including document download requests). Multiply it by 16 KB to calculate the amount of data contributed by Windchill HTML pages.

   Ex. if there are 1024 hits per hour, the data contributed by Windchill HTML pages is: 16 KB * 1024 = 16 MB

2. Estimate the average size of a document download. Multiply it by the number of document downloads per hour to calculate the amount of data contributed by document downloads. Where Replicated File Vaults are planned, consider the ratio of content served locally and adjust the number of downloads over the WAN accordingly.

Ex. if the average document download size is 800 KB and the total number of downloads is 100 then total data because of document downloads is: 800 KB * 100 ~= 78 MB

3. Estimate the average size of a document upload. Multiply it by the number of document uploads per hour to calculate the amount of data contributed by document uploads

   Ex. if the average uploaded document size is 64 KB and the total number of uploads is 200, then total data because of document uploads is: 64*200 ~= 12.5 MB

4. Estimate the number of RMI transactions per hour. RMI transactions are those conducted with either the Windchill Explorer or Product Information Explorer such as Create/Revise/Checkout Part, Navigate folder/part structure. Multiply the number of RMI transactions by the average RMI transaction size of 50KB.

   Ex. If 200 RMI transactions are expected per hour, the total RMI data transfer will be: 200 * 50KB ~= 10MB

5. Combine the totals for items 1 through 4 to obtain your total hourly traffic.

   Ex. Using our example, total hourly traffic is: 16+78+12.5+10 = 116.5 MB.

6. Convert the hourly bandwidth (MB/hr) requirement to Kb/sec. (a) Take the value from line 5 and multiply by 8388608 (number of bits in a Megabyte) then divide the result by 3600. (b) Express the result in terms of Kilobits by further dividing by 1024.

   a. Ex. (116.5 * 8388608) / 3600 = 271464 bits/sec

   b. 271464/1024 ~= 265 Kb/sec

Using the Kb/sec bandwidth requirement obtained from line 6 and the table below, find which bandwidth best suits the anticipated traffic. Keep in mind the increased average response times when network utilization is high. Therefore it would be prudent to size the network allowing for a peak 60% utilization[1]. Given the 265Kb/sec requirement from the example, a DSL connection would be sufficient, however since it would be at around 70% utilization, the average response times could be unacceptable. A half T1 would likely be more suitable by providing spare capacity for peak demand.

	Kbps Max	**Kbps 60%**
100baseT	102400	61440
T3	46080	27648

1. Different networks and protocols saturate at different utilization levels so a 40% peak might be preferable.

	Kbps Max	Kbps 60%
10baseT	10240	6144
T2	6463	3877
T1	1577	946
Half T1	768	461
DSL	384	231
ISDN	64	38
56Kbsp	56	34

E

Editing Windchill Properties Files

This appendix informs you about a special utility for editing Windchill Properties files and the environment in which you execute that utility. The environment is discussed before the utility.

Topic **Page**

About the windchill Command E-2
About the windchill shell E-4
About the xconfmanager Utility..... E-5

About the windchill Command

PTC has provided a command, windchill, to invoke Windchill actions. For example, the command can be used to stop and start Windchill, check the status of the Windchill server, and create a new shell and set the environment variables. It can also be used as a Java wrapper. In that regard, it can accept a Class file as an argument, just like Java, and execute it without a predefined environment (Windchill classes in CLASSPATH, Java in PATH, and so on).

The windchill command should be used to execute any server-side Windchill Java code. This will insure that the environment that the command is executed in is properly setup. The environment that actions are executed within, including the windchill shell action, is defined by the wt.env properties in the wt.properties file. For example, the wt.env.CLASSPATH property will set the CLASSPATH environment variable for the action that is being invoked.

The windchill command is a Perl script that has also been compiled into a Windows binary executable. For UNIX systems, Perl 5.0 or greater must be installed. The windchill script assumes that Perl is installed in the standard install location of /usr/bin/perl. If Perl is not installed at this location, you can either create a symbolic link (recommended method) to the Perl install location or edit the windchill script to reference the Perl install location. To modify the windchill script, edit the <*Windchill*>/bin/windchill file. Locate the #! entry (for example, #!/usr/bin/perl -w) and change the Perl directory to the location where Perl is installed.

The windchill command is located in the <*Windchill*>\ bin directory. If you receive a command not found message when you execute the windchill command, add the <*Windchill*>\bin directory to your PATH environment variable. The syntax of the windchill command is:

```
windchill [args] action
```

You can display the help for the windchill command by executing windchill with the -h argument or with no argument.

The following tables list some of the arguments and actions applicable to the windchill command. To see a complete list of the arguments, use the report generated from the help (argument).

windchill Arguments:

Arguments (optional)	Description
- h, --help	Displays help and exits.
-v, --[no]verbose	Explains what is being done when a command is executed. Default is noverbose.

Arguments (optional)	Description
-w, --wthome=DIR	Sets the Windchill home directory. Default is the parent directory containing the windchill script.
--java=JAVA_EXE	The Java executable. Default is the wt.java.cmd variable value specified in the $WT_HOME/codebase/wt.properties file.
-cp, --classpath=PATH	Java classpath. Default is the wt.java.classpath variable value specified in the $WT_HOME/codebase/wt.properties file.
--javaargs=JAVAARGS	Java command line arguments.

windchill Actions

Action	Description
shell	Sets up a Windchill environment in a new instance of the currently running shell.
start	Starts the Windchill server.
stop	Stops the Windchill server.
status	Retrieves the status of the Windchill server.
version	Displays the Windchill install version.

Action	Description
properties *<resource>[,...][?key[&key2]...]*	Displays the properties as seen by Windchill for the given resource with substitution, etc. executed. It can be limited to a given set of keys. For example: windchill properties wt.properties — lists all wt.properties windchill properties wt.properties?wt.server.codebase — lists server codebase windchill properties wt.properties?wt.env.* — lists all the environment variables use by windchill shell windchill properties — with no arguments generates the help report
CLASS [CLASS_ARGS]	Run a Windchill class with optional class arguments. For example: windchill wt.load.Developer -UAOps

About the windchill shell

The windchill shell brings up a new command shell, from the parent shell that is setup for the Windchill environment. This includes setting all environment variables defined in wt.env property in the wt.properties file.

To execute the windchill shell, at the command prompt enter the following command:

```
windchill shell
```

When you are finished using the windchill shell, you can exit the shell and return to the parent shell.

PTC recommends running all server-side Windchill applications, tools, and utilities from the windchill shell. Also, you can use the windchill shell to set up your development environment to use javac or Java directly.

About the xconfmanager Utility

The xconfmanager is a command line utility that you can run to add, remove, or modify properties in any Windchill property file. With one exception, the following files are managed by Windchill Information Modeler and should not be edited manually or edited with the xconfmanager:

- associationRegistry.properties
- classRegistry.properties
- descendentRegistry.properties
- modelRegistry.properties

The xconfmanager utility saves your changes in the site.xconf file and provides an option to generate updated property files using those changes. The site.xconf file contains changes made to Windchill property files, starting with installation and continuing with each use of the xconfmanager utility or the System Configurator. The xconfmanager utility is located in the <*Windchill*>/bin directory.

This chapter describes only the information and instructions necessary to modify specific Windchill properties. A full description of the xconfmanager utility and management of the Windchill property files is documented in the *Windchill System Administrator's Guide* in the Administering Runtime Services chapter.

Anyone with write access to the XCONF files and the property files under the Windchill installation directory can successfully run the xconfmanager utility. The xconfigmanger is executed from the command line from within a windchill shell. See the About the windchill Command for more information about the windchill shell.

The syntax of xconfmanager command is as follows:

```
xconfmanager {-FhuwvV} {-r <product_root>} {-s <property_pair>
{-t <property_file>}} {--reset <property_names>}
{--undefine <property_names>} {-d <property_names>} {-p}
```

For the purposes of modifying Windchill properties, you will primarily use the set (-s), targeFile (-t), and propagate (-p) parameters.

- The set (-s) parameter is used to identify the relevant property and specify the new property value. See the Formatting Property Value Guidelines section (below) for information about formatting the <*property_pair*) value.
- The targetFile (-t) property is used to specify the directory location of the property file. If the file name or path contains spaces, you must enclose the <*property_file*> value in double quotes (" "). It is recommended to use a fully qualified file name to ensure an accurate reference to the file is made.
- The propagate (-p) property is used to propagate the changes made to the XCONF files into the property file being modified in order to keep the XCONF and the property files in synch with one another.

- help is used to view the help for xconfmanager.

Some examples of using the xconfmanager utility are as follows:

- xconfmanager is run from the windchill shell. To open a windchill shell, execute the following command at a command prompt:

  ```
  windchill shell
  ```

- To display xconfmanager help, execute the following command from the windchill shell:

  ```
  xconfmanager -h
  ```

- To display the current settings for a property, execute the following command from the windchill shell:

  ```
  xconfmanager -d <property_names>
  ```

- To change a property value, execute the following command from the windchill shell:

  ```
  xconfmanager -s <property_pair>=<property_value>
  -t <property_file> -p
  ```

  Tip: Use the fully qualified name of the property file to ensure an accurate reference.

Formatting Property Value Guidelines

The property values you set must conform to the specification for java.util.Properties. The following guidelines will help ensure that you set properties correctly:

- Use forward slashes (/) in file paths so that the platform designation is not an issue.
- To specify a property whose value contains characters that might be interpreted by your shell (such as spaces and special characters), escape them using the appropriate technique for the shell you are using.

  For example, on a Windows system you can include spaces in a value by enclosing the argument with doubles quotes. For example, use the following:

  ```
  -s "wt.inf.container.SiteOrganization.name=ACME Corporation"
  ```

- On a UNIX system, you can use doubles quotes or you can escape the space character with a backslash. For example, use the following:

  ```
  -s wt.inf.container.SiteOrganization.name=ACME\ Corporation"
  ```

- On UNIX, dollar signs are usually interpreted by shells as variable prefixes. To set a property value that has a dollar symbol in it, use single quotes around the argument so that the shell does not interpreted it or use backslash to escape the dollar symbols. For example, use either of the following:

```
-s 'wt.homepage.jsp=$(wt.server.codebase)/wtcore/jsp/wt/portal/
index.jsp'
```

or

```
-s wt.homepage.jsp=
‘\$(wt.server.codebase)/wtcore/jsp/wt/portal/index.jsp
```

Other than escaping arguments so that the command-line shell does not misinterpret them, you should not need to escape other values to be compatible with XML or property file syntaxes. The xconfmanager escapes property names and values automatically if necessary.

Index

A

About This Guide, xi
ACL cache size, 3-6
administering Oracle, B-1
Analyzing full thread dumps, 10-6

B

background method server, 4-15
background server, 4-3
  background queues, 4-3
  number of servers, 4-5
Balancing system memory, 3-2
Basic Oracle Administration, B-1
BLOB space, C-4
bootstrap client, 5-2

C

cache master, 4-14
cache slaves, 4-13
CacheManager, 8-6
Caching, 8-2
Changing LDAP Properties for Load-Balancing Among Multiple Instances of Windchill Adapter, 4-8
cluster configuration, 4-12
  LDAP repositories, 4-17
  troubleshooting, 4-18
Collecting Oracle database statistics, 6-17
Composite indexes in Windchill, 6-13
Content Data, C-3
cost-based optimization, 6-7

D

DB_BLOCK_SIZE, 6-4
DB_CACHE_SIZE, 6-4
DB_KEEP_CACHE_SIZE, 6-4
DB_RECYCLE_CACHE_SIZE, 6-4
De-bug tracing, 3-10
Defining Multiple Instances of Windchill Adapter on a Server, 4-7
Defining Windchill Adapters in a Cluster Environment, 4-8
Defragmenting Indexes, 6-14
documentation
  conventions, xii
  references for Oracle, B-1
Dynamic SGA Configuration, 6-3

E

Editing Windchill Properties Files, E-1
Example Commands in Oracle Administration, B-3
execute queues, 4-14

F

FILE Name: explain_create.sql, A-3
find_all_index_info.sql, A-4
Function Indexes in Windchill, 6-14

G

General Oracle Tuning, B-2

H

hardware configuration, 4-2
HP-UX
  configuring kernel, 9-2
  monitoring system resources, 9-3

I

Identifying Internal Oracle Bottlenecks, 6-15
Implementing a cache, 8-2
Implementing a customized user interface, 5-7
Improving Oracle database performance, 6-2
index_rebuild.sql, A-5
indexes provided with Windchill, 6-12
Information about PTC setup, C-2
Init.ora Performance Improvement Features, 6-6
instance based parameter, 6-3
instance parameter, 6-3

J

Java heap size, 3-3
Java RMI server hostname, 4-15
java.rmi.server.hostname, 4-14

L

Load Point, C-3
load-balancing
  method servers, 4-5
  Windchill Adapter, 4-8
LOG_BUFFERS, 6-5
LOG_CHECKPOINT_INTERVAL, 6-5

M

Maintaining cache integrity, 8-3
master codebase, 4-14
master slave host, 4-15
Metadata, C-3
Modeling indexes, 6-13
monitor services, 4-15
Monitoring method server memory consumption, 10-7
monolithic hardware configuration, 4-2
multiple instances of Windchill Adapter, 4-7
Multiple Method Servers, 4-4, 4-5
  load balancing, 4-5
  Selecting a server, 4-5
multi-tiered hardware configuration, 4-2

N

Norton AntiVirus, 3-10

O

Operating system tuning, 9-1
Oracle, 6-2
  Basic Administration, B-1
  database performance, 6-2
  Database Sizing for Windchill, C-1
  general tuning, B-2
  Load Point, C-3
  performance, 6-2
  reference documents, B-1
  space use, B-1
  Tools to Collect Database Statistics, 6-17
  tuning, 6-1
  tuning for customized applications, 6-7

P

Performance checklist, 10-2
Performance Testing Guidelines, 2-1
persistence storage server, 4-14
persistent object manager, 3-10
Pre-Production Windchill Analysis, 6-10
properties files editing, E-4
PTC setup, C-2

R

Recommended Memory Ratios, 3-2
Reference Documents, B-1
Related Documents, xi
Resolving TCP name problems, 10-8

S

Selecting a server, 4-5
Separating tablespace and script files for indexes, 6-15
server cluster
  configuration of master, 4-14
  configuration of slaves, 4-13
server hostname, 4-14
Server selector Description, 4-5
Setting HTML page expiration, 5-8
Setting page expiration time, 5-8
Setting the storage clause, 6-15
SGA sizing, 6-3
SGA_MAX_SIZE, 6-5
SHARED_POOL_SIZE, 6-5
single method server, 4-2
Sizing the Database, C-3
Solaris
  monitoring system resources, 9-5
  recording thread dumps, 10-6
SQL queries, A-1
Symantec VirusScan, 3-10
system memory, 3-2

T

Third-Party Products, 1-xiii
thread dump, 10-6
three-tier architecture, 4-2
Threshold detection, 4-6
Tools to collect Oracle database statistics, 6-17
troubleshooting, 10-1
  cluster configuration, 4-18
  full thread dumps, 10-6
  Identifying Internal Oracle Bottlenecks, 6-15
  memory consumption, 10-7

performance checklist, 10-2
TCP name problems, 10-8
tuning SQL statements, 6-8
tuning
HP-UX, 9-2
Solaris, 9-3
Windows NT, 9-5
Tuning Oracle, 6-1
Tuning Oracle for customized applications, 6-7
Tuning Servlet Engines, 7-1
Tuning SQL Statements, 6-8

U

Understanding Space Use in Oracle, B-1
Useful SQL Queries, A-1
Using a server cluster, 4-12
Using anti-virus software, 3-10
Using indexes provided with Windchill, 6-12
Using the Method Context class, 8-6

W

Windchill Adapter Performance Issues, 4-5, 4-7
Windchill Cache Mechanism, 8-1
Windchill Client Tuning Guidelines, 5-1
windchill command, E-1
Windchill Configuration Options, 4-1
Windchill Network Sizing Guidelines, D-1, E-1
Windchill Server Tuning Guidelines, 3-1
windchill shell, E-1
Windows NT, 9-5
Equal application priority, 9-8
monitoring system resources, 9-5
network applications, 9-9
Performance Monitor, 9-6
Task Manager, 9-7

X

xconfmanager, E-5