NASTEL
Driving Business Transaction Performance ®

APM6/BP 600.005

AutoPilot® Best Practices Guide

Version 6.0.2

Published by:

R&D DEPARTMENT
NASTEL TECHNOLOGIES, INC.
48 SUNNYSIDE BLVD, SUITE 101
PLAINVIEW, NY 11803

TITLE: **AUTOPILOT BEST PRACTICES GUIDE**

DOCUMENT RELEASE DATE: **FEBRUARY 2019**

NASTEL DOCUMENT NUMBER: **APM6/BP 600.005**

ACKNOWLEDGEMENTS:

THE FOLLOWING TERMS ARE TRADEMARKS OF NASTEL TECHNOLOGIES CORPORATION IN THE UNITED STATES OR OTHER COUNTRIES OR BOTH: TRANSACTIONWORKS, AUTOPILOT M6, AUTOPILOT/IT, AUTOPILOT M6 FOR WMQ, AUTOPILOT/WMQ, M6 WEB SERVER, AUTOPILOT/WEB, M6 WEB CONSOLE, MQCONTROL, MQCONTROL EXPRESS, AUTOPILOT/TRANSACTION MONITOR, AUTOPILOT/WAS, AUTOPILOT/OS MONITOR

THE FOLLOWING TERMS ARE TRADEMARKS OF THE IBM CORPORATION IN THE UNITED STATES OR OTHER COUNTRIES OR BOTH: IBM, MQ, MQSERIES, WEBSPHERE, WEBSPHERE MQ WIN-OS/2, AS/400, OS/2, DB2, AND AIX, Z/OS

THE FOLLOWING TERMS ARE TRADEMARKS OF HEWLETT-PACKARD IN THE UNITED STATES OR OTHER COUNTRIES OR BOTH: OPENVIEW, HP-UX

COMPAQ, THE COMPAQ LOGO, ALPHASERVER, COMPAQ INSIGHT MANAGER, CDA, DEC, DECNET, TRUCLUSTER, ULTRIX, AND VAX REGISTERED IN U.S. PATENT AND TRADEMARK OFFICE. ALPHA AND TRU64 ARE TRADEMARKS OF COMPAQ INFORMATION TECHNOLOGIES GROUP, L.P IN THE UNITED STATES AND OTHER COUNTRIES

SNMPC, SNMPC, WORKGROUP, AND SNMPC ENTERPRISE ARE TRADEMARKS OF CASTLE ROCK COMPUTING IN THE UNITED STATES OR OTHER COUNTRIES, OR BOTH.

SUN, SUN MICROSYSTEMS, THE SUN LOGO, IFORCE, JAVA, NETRA, N1, SOLARIS, SUN FIRE, SUN RAY, SUNSPECTRUM, SUN STOREDGE, SUNTONE, THE NETWORK IS THE COMPUTER, ALL TRADEMARKS AND LOGOS THAT CONTAIN SUN, SOLARIS, OR JAVA, AND CERTAIN OTHER TRADEMARKS AND LOGOS ARE TRADEMARKS OR REGISTERED TRADEMARKS OF SUN MICROSYSTEMS, INC. IN THE UNITED STATES AND OTHER COUNTRIES.

INSTALLANYWHERE IS A REGISTERED TRADEMARK OF ZEROG SOFTWARE IN THE UNITED STATES OR OTHER COUNTRIES, OR BOTH.

THIS PRODUCT INCLUDES SOFTWARE DEVELOPED BY THE APACHE SOFTWARE FOUNDATION (HTTP://WWW.APACHE.ORG/). THE JAKARTA PROJECT" AND "TOMCAT" AND THE ASSOCIATED LOGOS ARE REGISTERED TRADEMARKS OF THE APACHE SOFTWARE FOUNDATION

INTEL, PENTIUM AND INTEL486 ARE TRADEMARKS OR REGISTERED TRADEMARKS OF INTEL CORPORATION IN THE UNITED STATES, OR OTHER COUNTRIES, OR BOTH

MICROSOFT, WINDOWS, WINDOWS NT, WINDOWS XP, AND THE WINDOWS LOGOS ARE REGISTERED TRADEMARKS OF THE MICROSOFT CORPORATION.

UNIX IS A REGISTERED TRADEMARK IN THE UNITED STATES AND OTHER COUNTRIES LICENSED EXCLUSIVELY THROUGH X/OPEN COMPANY LIMITED.

"LINUX" AND THE LINUX LOGOS ARE REGISTERED TRADEMARKS OF LINUS TORVALDS, THE ORIGINAL AUTHOR OF THE LINUX KERNEL. ALL OTHER TITLES, APPLICATIONS, PRODUCTS, AND SO FORTH ARE COPYRIGHTED AND/OR TRADEMARKED BY THEIR RESPECTIVE AUTHORS.

SCO CUSA, SCO DOCTOR, SCO DOCTOR FOR NETWORKS, SCO DOCTOR LITE, SCO GLOBAL ACCESS, SCO MPX, SCO MULTIVIEW, SCO NIHONGO OPENSERVER, SCO OK, THE SCO OK LOGO, SCO OPENSERVER, SCO OPEN SERVER, SCO PORTFOLIO, SCO POS SYSTEM, SCO TOOLWARE, AND THE WORLD NEVER STOPS ARE TRADEMARKS OR REGISTERED TRADEMARKS OF CALDERA INTERNATIONAL, INC. IN THE U.S.A. AND OTHER COUNTRIES, ALL RIGHTS RESERVED.

ORACLE® IS A REGISTERED TRADEMARK OF ORACLE CORPORATION AND/OR ITS AFFILIATES

OTHER COMPANY, PRODUCT, AND SERVICE NAMES, MAY BE TRADEMARKS OR SERVICE MARKS OF OTHERS.

Table of Contents

Figures

Tables

Chapter 1: Introduction

Welcome to the Nastel *AutoPilot M6 Best Practices Guide*. This guide describes AutoPilot M6 installation, deployment and usage. Nastel AutoPilot M6 will hereinafter be identified as AutoPilot M6 or M6. WebSphere MQ deployment and usage is described in Chapter 6.

1.1 Use of this Document

This guide is intended for anyone administering or using an AutoPilot M6 product including AutoPilot M6 Enterprise, AutoPilot M6 TM and AutoPilot M6 WMQ. The guide assumes that the user is familiar with general system and application management concepts, networking and TCP/IP.

1.2 Guide Organization

Chapter 1:	Introductory information relevant to the document and AutoPilot M6.
Chapter 2:	Describes installation and deployment guidelines for AutoPilot M6.
Chapter 3:	Contains rules and guidelines for using AutoPilot M6.
Chapter 4:	Contains troubleshooting techniques for AutoPilot M6.
Chapter 5:	Contains a listing of key performance indicators along with their definitions.
Chapter 6:	Describes deployment and usage for WebSphere MQ.
Chapter 7:	Describes typical AutoPilot M6 deployments.
Appendix A:	Detailed list of all reference information required for the operation of AP M6.
Appendix B:	Contains conventions used in AutoPilot M6 and documents typographical conventions.
Glossary:	Contains a listing of unique and common acronyms, words and definitions.

1.3 History of this Document

Table 1-1. Document History

Release Date	Document Number	AutoPilot M6 Version	Summary
February 2008	APM6/BP 600.001	6.0 and higher	Initial release
January 2009	APM6/BP 600.002	6.0 and higher	Performance Updates
August 2015	APM6/BP 600.003	6.0 and higher	General Update
July 2017	APM6/BP 600.004	6.0 and higher	Update Nastel's phone number and street address
February 2019	APM6/BP 600.005	6.0 and higher	Clarify LDAP information

1.3.1 User Feedback

Nastel encourages all users of AutoPilot M6 to submit comments, suggestions, corrections, and recommendations for improvement for all AutoPilot M6 documentation. Please send your comments via email. Send messages to: support@nastel.com. You will receive a response, along with status of any proposed change, update, or correction.

1.4 Related Documents

Complete listings of documents related to AutoPilot M6 can be found in Appendix A.

1.5 Intended Audience

This document is intended for personnel installing, customizing and using AutoPilot M6. The user who installs the product should be familiar with:

- J2EE (Tomcat, WebLogic, WAS) for AutoPilot M6 Web deployment
- Java Run Time Environment 15 (JRE 1.5) or later
- Basic understanding of TCP/IP

1.6 System Requirements

All software and hardware requirements are defined in *AutoPilot M6 Installation Guide*.

1.7 Terms and Abbreviations

A list of Terms and Abbreviations used in this document is located in the Glossary.

1.8 Technical Support

1.8.1 Nastel Support

If you need technical support, you can contact Nastel Technologies by telephone, email or through the online support system.

To contact Nastel technical support by phone, dial **800-963-9822 ext. 1**. If you are calling from outside the United States, dial **001-516-801-2100**.

To contact Nastel technical support by email, send a message to support@nastel.com

To access the Nastel automated support system (user id and Password required), go to: http://support.nastel.com/. Contact your local AutoPilot M6 Administrator for further information.

1.8.2 The Resource Center

The Resource Center is where AutoPilot M6 solution users solve problems, exchange ideas, and learn best practices from peers and Nastel staffers. This online community is a service provided by Nastel Support. Our experts often participate in these discussion groups to share their advice, but these groups are intended as a peer-to-peer resource.

The Resource Center also provides access to downloads, updates, documentation, support articles, monthly newsletter, product news, and a lot more. Registration and access are free to all users. We encourage everyone who uses Nastel products to join the Nastel Resource Center at http://www.nastel.com/resources.

1.9 Conventions

Refer to Appendix B for conventions used in this guide.

Chapter 2: Installation and Deployment

2.1 Prerequisites

The following will make installation and deployment run more efficiently:

- Define environment variables for mail server, email groups, nodes, workgroup server, in node.properties.
- Define actions using the Action Manger instead of placing them directly into the business view.
- Build application specific business views as well as overall health business views. For better Dashboard performance, determine what objects belong to each specific application.
- Select appropriate *Service category* on the *General* tab of the *Properties* screen of each business view for use with the Dashboard.
- Determine database type and space requirements using a standard formula of 512 bytes. Take into account the *number of logging sensors * polling interval*.
- Request from Nastel support a CPU-based license to replace the old license.

2.2 Deploying an Expert

One of the major causes of performance degradation is redundancy. This means duplication in fact collection and duplication in sensors. To avoid performance degradation, perform the following steps before deploying an expert:

1. **Define Requirements:** It is very important to define monitoring requirements clearly.
    - Define all the following monitoring requirements.
        - The technical requirement
        - The people/groups to be alerted
        - The action to be executed.
        - The security requirements for the business view
    - For each requirement, define the proposed solution.
        - The expert you will use
        - The sensor you will create.
2. **Experts:** Create and maintain a spreadsheet with the following: (See Figure 2-1 below.)
    - List all the experts you need to use.
    - List all the types of experts you need to use.
    - Define an environment variable for each expert in node properties.
    - Estimate the number of objects the expert will monitor.
    - Estimate the number of facts produced for each object.
    - Determine the total facts produced by the expert by multiplying:
      *Number of Objects * Number of Facts per Object*
    - If the expert is based on polling, define the polling intervals you will use.
    - Estimate the total number of facts produced by all experts – Sum of *Total Facts* column.

Expert Name	Expert Type	Environment Variable Used	Estimated Number of Objects	Facts Per Object	Total Facts	Polling Intervals

Figure 2-1. Example of Spreadsheet Column Headers

3. **Business Views:** Create a spreadsheet with the following:
    - List all business views.
    - List all sensors you will create.
    - List facts used by each sensor in alphabetical order. (Ensure that you are using environment variables for expert names that you have defined in the node properties).
    - List all local environment variables you will use. Follow a naming convention, (see Chapter 3), if you reference an object of a particular type with an environment variable. Use the same variable for objects of that type in all your sensors.
    - List the action you are using in the sensor.
    - List the location where the sensor will be stored.
4. **Avoid Duplication**
    - Browse through your list of experts and determine if any are collecting similar information.
    - If it is possible to gather all required information using one expert, eliminate all secondary experts.
    - Sort all the sensors created by the facts used and eliminate any duplicates you discover.
    - Repeat this process every time you add a new sensor.

Chapter 3: Guidelines and Considerations

AutoPilot M6 is designed to monitor and control distributed IT services such as application servers, middleware, user applications, workflow engines, brokers, Service Oriented Architecture (SOA) and Enterprise Service Bus (ESB) based applications and their impact on business services.

3.1 Naming Conventions

The following rules should be used when naming services, policies, experts and managers:

- Avoid blanks in service, policy and user definitions
- Avoid special characters such as $ * # ; :
- Name Experts as <*prefix*>_Monitor
- Name Managers as <*prefix*>_Manager
- Name Policies as <*prefix*>_Policy
    - Use manager‘s auto-naming convention feature to enforce it
- Use service **Context** property to categorize services
    - By category
    - By usage or purpose

3.2 Guidelines for Building Policies

3.2.1 Modeling

The following rules should be used when modeling policies:

- Avoid hard-coded names – use environment variables.
    - Global environment variables. For example: Java variables or CEP server variables (node.properties).
    - Policy environment variables.
- Avoid duplication – promote modularity and reuse.
    - Define and enforce naming conventions.
    - Focus on creating modular, reusable business views.
    - Create complex policies by combining simple ones.
- Use the Keep it Small, Stupid and Simple (KISSS) approach.
    - Build small, simple and self-contained policies.
    - Link your policies together. Do NOT use copy and paste functions.
    - Reference existing policies – reuse your existing logic.
- Complex Event Processing using business views.
    - A business view is an event processor and should be used as one.
- Sensors should model local facts.
    - In a multiple CEP-server deployment environment, avoid cross reference of facts (increases performance).
    - For global views, combine policies together using links (increases performance).

3.2.2 Deployment

The following rules should be used when deploying policies:

- Deploy policies close to their source:
  - Use the same CEP server as the origin of most facts (performance).
  - Use the same server as the origin of most facts.
  - Use a set of smaller policies. Smaller policies perform better than one large one.
- Always consider security:
  - Determine if the policy should be available on the web. If it should, then determine what the correct permissions model should be.
  - Edit the policy security properties to set permissions and ownership.
  - Do not create a user name longer than 12 characters; otherwise, it would be truncated in translation – even if `xxxxx@NASTEL.MAIN` is created in permits database. Specify the first qualifier of the user name only.
- Each CEP server should:
  - Be self-sufficient and independent of other CEP servers.
  - Contain all the monitors and policies for monitoring.
- Make use of multiple manager instances to improve performance and response time. Group by policy category or purpose.
- Check policy integrity when designing and deploying policies. Check Integrity button is located on the Business View editor‘s tool bar. The integrity check will verify that all environment variables are defined, all services referenced actually do exist, and many other common mistakes.

3.2.3 Circular References

Circular references are links between policies that create a logical loop between two managers deployed within one or more CEP servers. Circular references create a situation where fact updates in one manager cause updates in another and vice versa. Use *Integrity Check* on policies deployed on both managers to determine if there are any circular references. A common way to resolve/break such references is to deploy one of the policies into a different manager instance. (See Figure 3-1 below.)

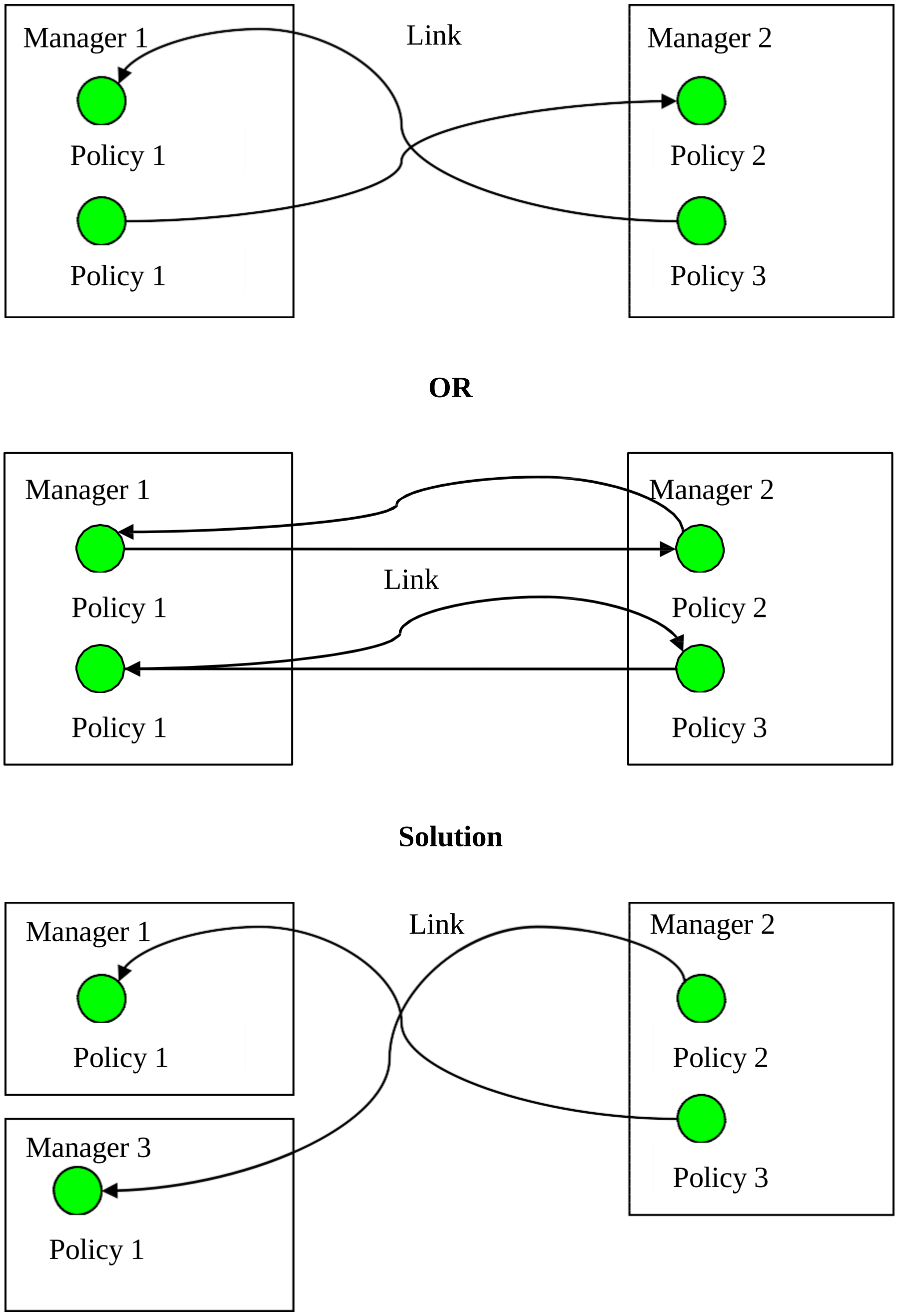


Figure 3-1. Circular Reference Solution

3.2.4 Link Versus Clone Sensors

Sensor links allows users to create references to the run-time state of previously deployed policies. This is useful when the result of a deployed policy needs to be factored into another policy – similar to the linking of HTML pages.

Clone sensors let users share sensor definitions (such as rules, alerting, etc.) among multiple sensors. Clone sensors load the template definition upon policy startup and copy it into the runtime instance. Changes in the template are reloaded into every clone sensor that references the template (policy restart is required). Use clone sensors when sensor definition sharing is required. This will avoid duplication of rules and definitions and decrease the overall administrative overhead associated with policy design deployment.

3.2.5 Sensor Rule Structure Usage

When creating and using policies:

- Avoid blocking scripts within sensors. This causes rule execution delays and decreases effective rule/sec processing speed.
- Avoid excessive dynamic sensor creation and deletion during policy execution. Structure rules so that dynamic sensors are only created when an error condition occurs and are removed when the condition is resolved.
- Avoid sensors that use multiple facts for evaluation (if possible). The more facts included in a sensor, the more processing is required during rule execution.
- Choose meaningful name for the sensor; include the key objects being monitored to be part of the name.
- Enable *Expire Dynamic Sensor* in xx amount of seconds for dynamic sensors.
- Enable desired severities for generating alerts for the sensor. These severities should match one of the possible severities selected in the previous step.
- Specify the email details. *From user*, *Mail server* and *Email to* fields are populated from environment variables defined in global.properties or domain.properties.

3.3 Performance Considerations

Every CEP server publishes information about its memory and fact utilization. This information is published under [*NODE_NAME*]_Facts\Java and [*NODE_NAME*]_Facts\Topic. These facts can be included into business views and can be monitored and alerted on.

3.3.1 CPU Utilization and Estimation

CPU utilization is proportional to the number and scope of the business views deployed with the CEP server. Estimate CPU utilization and load by using the following formula:

(*TopicFacts/sampling_rate*) * *totalSensors~operations/sec*

Example: (100000/100 (seconds)) * 1000 = 1,000,000 logical operations per second.

3.3.2 Memory Utilization and Estimation

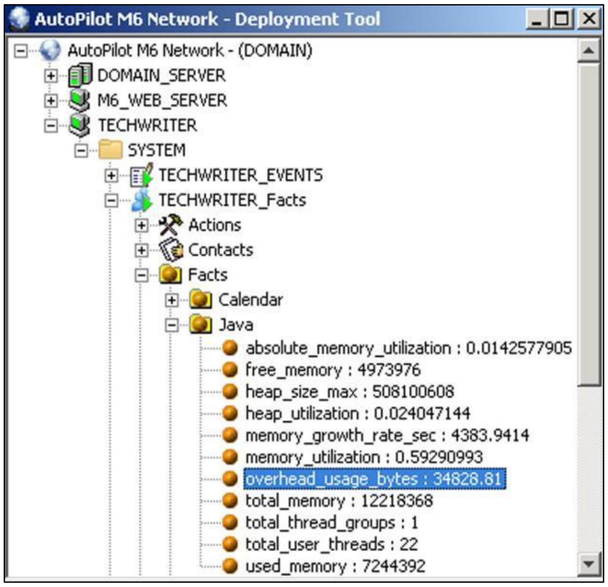
Memory utilization, *memory_utilization*, is proportional to:

- Number of experts and managers deployed
- Number of facts published within the CEP server

  (*totalFacts* should be less than *maxFacts*)
- Number and scope of business views deployed with the CEP server

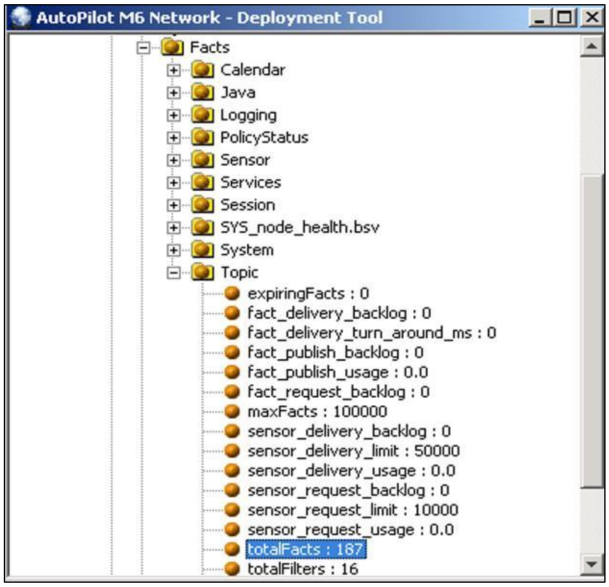
  (*estimated total_memory~totalSensors * totalFacts * overhead_usage_bytes*)

For example, average memory utilization for a CEP server with 10-15 experts and 50,000-75,000 facts may require 60-90 MB of virtual memory (platform dependant).



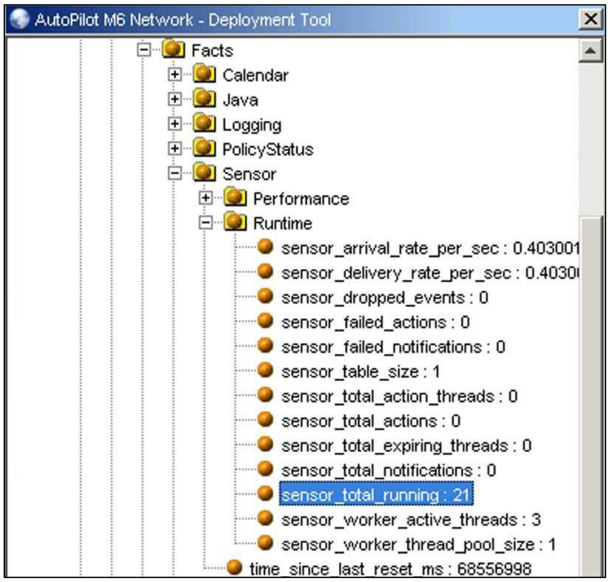
AutoPilot M6 Network - Deployment Tool

```
⊟ AutoPilot M6 Network - (DOMAIN)
    ⊞ DOMAIN_SERVER
    ⊞ M6_WEB_SERVER
    ⊟ TECHWRITER
        ⊟ SYSTEM
            ⊞ TECHWRITER_EVENTS
            ⊟ TECHWRITER_Facts
                ⊞ Actions
                ⊞ Contacts
                ⊟ Facts
                    ⊞ Calendar
                    ⊟ Java
                        absolute_memory_utilization : 0.0142577905
                        free_memory : 4973976
                        heap_size_max : 508100608
                        heap_utilization : 0.024047144
                        memory_growth_rate_sec : 4383.9414
                        memory_utilization : 0.59290993
                        overhead_usage_bytes : 34828.81
                        total_memory : 12218368
                        total_thread_groups : 1
                        total_user_threads : 22
                        used_memory : 7244392
```



AutoPilot M6 Network - Deployment Tool

```
⊟ Facts
    ⊞ Calendar
    ⊞ Java
    ⊞ Logging
    ⊞ PolicyStatus
    ⊞ Sensor
    ⊞ Services
    ⊞ Session
    ⊞ SYS_node_health.bsv
    ⊞ System
    ⊟ Topic
        expiringFacts : 0
        fact_delivery_backlog : 0
        fact_delivery_turn_around_ms : 0
        fact_publish_backlog : 0
        fact_publish_usage : 0.0
        fact_request_backlog : 0
        maxFacts : 100000
        sensor_delivery_backlog : 0
        sensor_delivery_limit : 50000
        sensor_delivery_usage : 0.0
        sensor_request_backlog : 0
        sensor_request_limit : 10000
        sensor_request_usage : 0.0
        totalFacts : 187
        totalFilters : 16
```



AutoPilot M6 Network - Deployment Tool

```
Facts
  Calendar
  Java
  Logging
  PolicyStatus
  Sensor
    Performance
    Runtime
      sensor_arrival_rate_per_sec : 0.403001
      sensor_delivery_rate_per_sec : 0.40300
      sensor_dropped_events : 0
      sensor_failed_actions : 0
      sensor_failed_notifications : 0
      sensor_table_size : 1
      sensor_total_action_threads : 0
      sensor_total_actions : 0
      sensor_total_expiring_threads : 0
      sensor_total_notifications : 0
      sensor_total_running : 21
      sensor_worker_active_threads : 3
      sensor_worker_thread_pool_size : 1
    time_since_last_reset_ms : 68556998
```

Figure 3-2. Performance Considerations

3.3.3 Workload Performance

AutoPilot Experts/Universal Agents work management must be configured properly to handle the requested workload. Poor performance results from uneven workloads and not configuring the Nastel product to handle shifting workloads.

- Deploy only experts that meet your monitoring requirements.
- Avoid collection of metrics that are not being monitored.

Below are AutoPilot M6 for WebSphere MQ Universal Agents performance tuning parameters which are displayed in Universal Agent properties, *Discovery* tab. Adjust each parameter as follows:

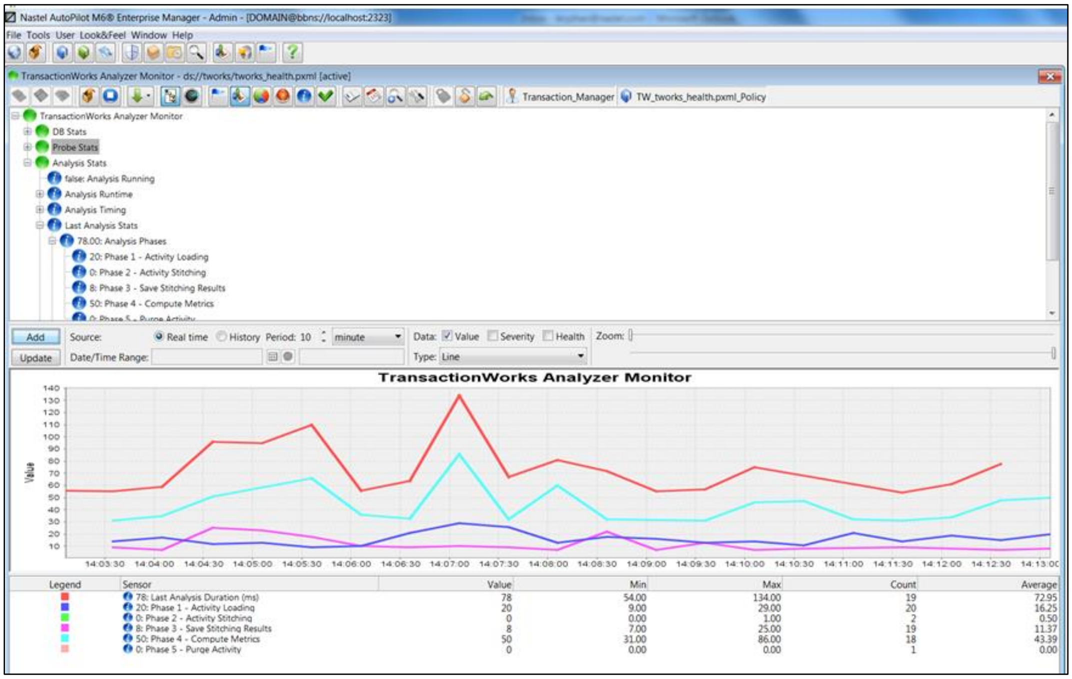
- **Command Timeout:** Command timeout in milliseconds. Commands issued to Workgroup Agent will timeout after the specified time interval. Default is 60000 ms. For best results, the time should be set to 20000 ms.
- **Discovery Period:** Period in seconds. Object discovery poll interval. Use 60-120 seconds for up to 1000 objects, 180-300 seconds for up to 5000 objects, and 500+ seconds for 5000+ objects.
- **Poll Delay Interval:** Interval in milliseconds. Delay between each request to Workgroup Agent within a single discovery interval. Default is 200 ms. For best results, the time should be set to 15-30 ms.

3.3.4 Optimizing Performance

- Performance figures vary greatly due to hardware, CPU and OS environment. For maximum performance using AutoPilot M6, consider high CPU speed and at least 2 CPU configurations, with a minimum of 4 GIG of RAM, x64 architecture.
- Workflow can be optimized to improve performance by minimizing size of business views.
- Increase discovery time for all Experts and Universal Agents.
- Eliminate duplications of experts and business views. (Refer to Chapter 2).

3.3.5 Performance Data

The charting option is available in business view by clicking the **Chart** icon his provides the capability to graph data in real-time to monitor health and view historical performance data which will help to project and recover from a failure.



Nastel AutoPilot M6® Enterprise Manager - Admin - [DOMAIN@bbns://localhost:2323]

TransactionWorks Analyzer Monitor - ds://tworks/tworks_health.pxml [active]

Transaction_Manager TW_tworks_health.pxml_Policy

- TransactionWorks Analyzer Monitor
  - DB Stats
  - Probe Stats
  - Analysis Stats
    - false: Analysis Running
    - Analysis Runtime
    - Analysis Timing
    - Last Analysis Stats
      - 78.00: Analysis Phases
        - 20: Phase 1 - Activity Loading
        - 0: Phase 2 - Activity Stitching
        - 8: Phase 3 - Save Stitching Results
        - 50: Phase 4 - Compute Metrics
        - 0: Phase 5 - Purge Activity

Source: ● Real time ○ History Period: 10 minute

Data: ☑ Value □ Severity □ Health Zoom:

Date/Time Range:

Type: Line

Legend	Sensor	Value	Min	Max	Count	Average
🟥	78: Last Analysis Duration (ms)	78	54.00	134.00	19	72.95
🟦	20: Phase 1 - Activity Loading	20	9.00	29.00	20	16.25
🟩	0: Phase 2 - Activity Stitching	0	0.00	1.00	2	0.50
🟪	8: Phase 3 - Save Stitching Results	8	7.00	25.00	19	11.37
🟦	50: Phase 4 - Compute Metrics	50	31.00	86.00	18	43.39
🟥	0: Phase 5 - Purge Activity	0	0.00	0.00	1	0.00

Figure 3-3. Chart Data

3.4 Estimating Memory, CPU and Latency

3.4.1 Memory Utilization

To estimate CEP server memory utilization, use the following formula:

*Java\overhead_usage_bytes * (Topic\totalFacts + Sensor\Runtime\sensor_total_running*)

When evaluating required memory for a CEP server, use the following formula:

*Java\overhead_usage_bytes * (planned_total_facts + planned_total_sensors) = ~Required_Max_Heap _Size*

3.4.2 CPU Utilization

To estimate CPU utilization and load, use the following formula:

(*Topic\totalFacts\(Avg Sampling rate)) * Sensor\Runtime\sensor_total_running*.

For example: (100000/100 (seconds)) * 1000 = 1,000,000 logical operations per second.

where:

The average sampling rate can be estimated based on the sum of all sampling intervals for all monitors divided by the total number of monitors.

3.4.3 Time Latency

To estimate total time latency in milliseconds, use the following formula:

(*Sensor\Performance\sensor_turn_around_time_ms + Topic\fact_delivery_turn_around_time_ms*)

This is the time that the user would experience from the moment a fact is published to it being processed and displayed by business views.

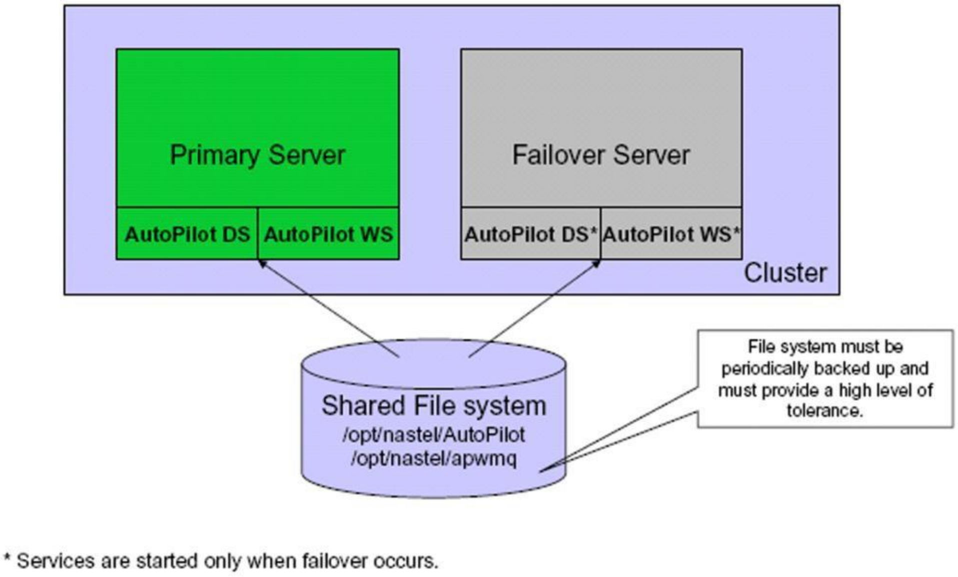
Logging\db_latency_ms reflects the database logging latency, which is the time it would take to drain all outstanding SQL updates. This value depends on the logging frequency and the speed of the database.

It is recommended that users monitor the above mentioned facts and alert when user thresholds are breached. Instead of creating a new policy to monitor the above facts, make use of the supplied **ds://SYSTEM/nodes/node_health.bsv** business view which is automatically loaded on every CEP server by the <*node*> Facts service. This policy already monitors time latency and can automatically stop/start services when thresholds are breached.

3.4.4 Failover

3.4.4.1 Overview

- Make use of 2 or more servers
- Servers must be joined in a cluster
  - Veritas
  - Sun Clusters
  - Windows Clusters
- AutoPilot, AutoPilot/WMQ installed on a shared file system
- File system mounted on every system within the cluster
- Failover is performed using clustering software
- All failover policies, failover-failback is done using clustering facilities



* Services are started only when failover occurs.

Figure 3-4. Failover

3.4.4.2 Benefits

- Handles server and software failover
- Requires no custom scripts
- Requires no file replication
- Failover can be accomplished using uniform mechanism
- Easy to maintain and use for all failover scenarios
- Policy based failover approach
- Few manual steps reduces errors.

3.4.4.3 Software Failover Sequence

Refer to the Figures 3-5 through 3-7 below.

Primary Server

- Monitor Secondary state
- If secondary down
  - Run start_script.sh
  - Use a prestartdelay of several minutes

Secondary Server

- Monitor Primary state
- If primary down
  - Run start_script.sh
- If primary is up
  - Run stop_script.sh

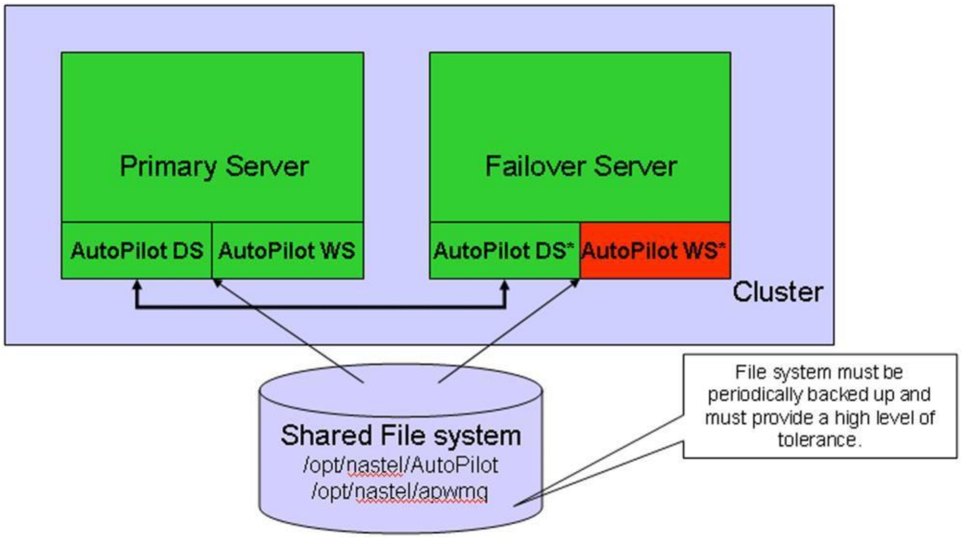


Figure 3-5. Primary UP

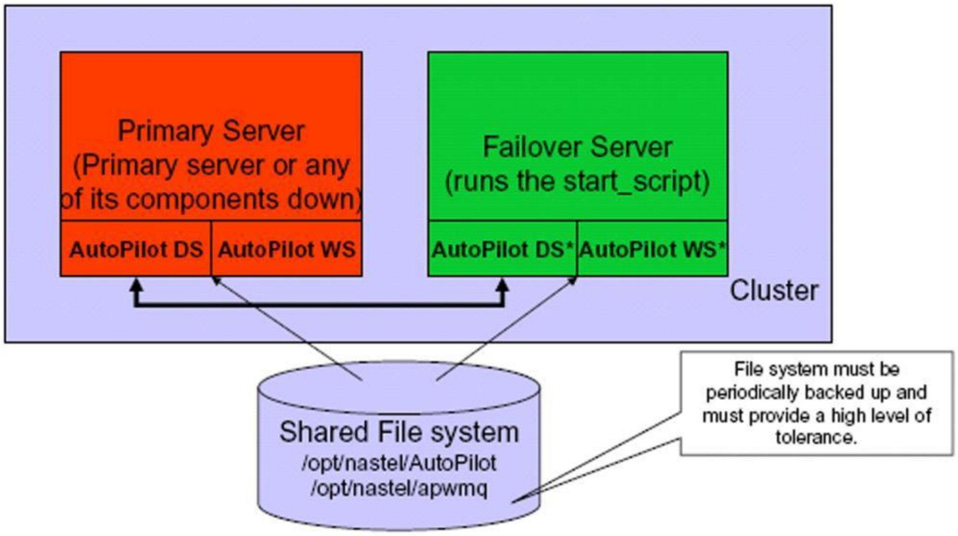


Figure 3-6. Failover

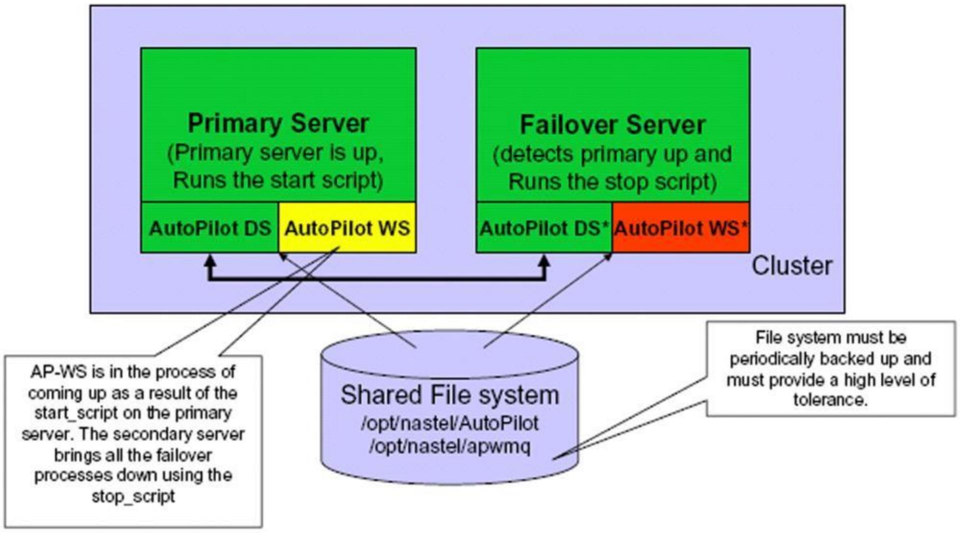

Primary Server
(Primary server is up,
Runs the start script)
AutoPilot DS
AutoPilot WS
Failover Server
(detects primary up and
Runs the stop script)
AutoPilot DS*
AutoPilot WS*
Cluster
AP-WS is in the process of coming up as a result of the start_script on the primary server. The secondary server brings all the failover processes down using the stop_script
Shared File system
/opt/nastel/AutoPilot
/opt/nastel/apwmq
File system must be periodically backed up and must provide a high level of tolerance.

Figure 3-7. Failback

3.4.4.4 Start/Stop Scripts

This process assumes that all file systems have been synchronized.

Start Script

- Start CEP server
- Start workgroup servers
- Other start procedures

Stop Script

- Stop CEP server
- Stop workgroup servers
- Other stop procedures if necessary

3.4.4.5 DNS Recommendations

- Primary and Secondary servers must have
  - Private IP address/hostname (static)
  - Public IP address/hostname (floating)
- Primary and secondary servers always communicate using private addresses
- Failover and failback occurs using private addresses
- Failover or public addresses should occur before the failover or failback scripts are launched.

3.4.4.6 Automatic Switchover from Failed IP

For automatic switchover to occur, *TCP Keep Alive* option must be enabled to reconnect to the fail-over domain.

	NOTE:	The keepalive timeout is an OS setting - not AutoPilot.

When the keepalive option is set for a TCP socket and no data has been exchanged across the socket in either direction for 2 hours, TCP automatically sends a keepalive probe to the peer. This probe is a TCP segment to which the peer must respond. One of three responses is expected:

- The peer responds with the expected ACK. The application is not notified (since everything is OK). TCP will send another probe following another 2 hours of inactivity.
- The peer responds with an RST, which tells the local TCP that the peer host has crashed and rebooted. The socket is closed.
- There is no response from the peer. The socket is closed. The purpose of this option is to detect if the peer host crashes. Valid only for TCP socket: SocketImpl

The following syntax should be used:

```
property server.socket.option.SO_KEEPALIVE=true
```

When an IP failure is detected, it disconnects. The console will retry the connection and the IP will be switched.

3.4.4.7 Transaction Monitor (TM) Failover Considerations

- TM maintains much of the information in memory and therefore failover process will lose the in-memory data.
- Database failover is recommended in case database fails.
- Database performance must be tuned to support the flow of transactions and keep up with transaction rate per sec.
- If Transaction Analyzer (TA) is down, messages will be accumulating on the individual queue managers and queues must be sized accordingly to accumulate messages for the maximum downtime.

  Example: if you are expecting 50msg/sec transaction rates, you should plan for 1 full day of backlog of around 5M messages on a queue per queue manager.

  (1440*60 (sec))*50msg/sec=4320000 (msgs)

Chapter 4: Troubleshooting

Use the following for general troubleshooting techniques for M6 components.

- Use Performance Utilities for M6 CEP servers
  - jdump allows users to generate thread stacks traces as well as pin-point potential deadlocks remotely.
  - jtop monitors CPU usage of all threads within a CEP server or domain server.
- Use utilities that quickly determine if the problem is application, network or hardware related while isolating specific application calls that are responsible for poor performance.
- Use `[AUTOPILOT_HOME]\localhost\ATPNODE -console -debug` when a CEP server problem exists. This displays all java-related properties when starting the node.
- Use `[AUTOPILOT_HOME]\naming\ATPNAMES -console -debug` when a Domain Server problem exists. This displays all java-related properties when starting the Domain Server.

This page intentionally left blank.

Chapter 5: Key System Indicators

The following indicators in this chapter are run-time performance indicators and are found under SYSTEM under each CEP server.

Table 5-1. System Calendar

Calendar	Type	Description
day_of_month	Integer	Current day count within the current month.
day_of_week	Integer	Current day count within the current week.
day_of_week_in_month	Integer	Current day of week count within the current month.
day_of_year	Integer	Current day count within the current year.
month	Integer	Month count within current year. For Example: 6 – indicates June.
time_zone	String	Current time zone (EDT, CDT, GMT, etc.)
week_of_month	Integer	Current week count within the current month.
week_of_year	Integer	Current week count within the current year.
year	Integer	Current year.

Table 5-2. Java Memory

Java	Type	Description
absolute_memory_utilization	Float	Ratio of memory used to maximum heap size. Multiply this number by 100 to get % of used memory.
free_memory	Long	Number of bytes of available memory based on the currently allocated heap size (not based on the max heap size)
heap_size_max	Long	Maximum memory that can be claimed by the server. If consumed 100%, no more memory can be allocated and CEP server will fail.
heap_utilization	Float	Current heap utilization. Multiply by 100 to get % usage. If reaches 1.0 (100%), CEP server will no longer be able to claim more memory and will fail.
memory_growth_rate_sec	Float	Number of bytes claimed per second. If the number is positive, the memory is growing. If the number is negative, the memory is shrinking. The value is computed based on the previous sample (every 30 sec).
memory_utilization	Float	Current memory utilization based on the currently allocated heap size (which could be less than or equal to max heap size).
overhead_usage_bytes	Float	Number of bytes used per functional unit which is defined as (fact + sensor). This number can be used to estimate required memory for a given number of facts + sensors. For example: If over_head_usage_bytes = 6000, then to support 10000 facts + 10000 sensors, the user will need 6000*(20000) bytes.
total_memory	Long	Total number of bytes allocated. This number will always be less than or equal to heap_size_max. This number will usually be close to the number of bytes claimed from the Operating System.
total_thread_groups	Integer	Total number of allocated thread groups by the CEP server. Each group may hold 0 or more threads.
total_user_threads	Integer	Total number of threads running. Large number of threads may reduce CEP server performance due to thread management overhead.
used_memory	Long	Current number of bytes used by the CEP server within the Java Virtual Machine. This number does not indicate the amount of memory claimed from the Operating System.

Table 5-3. Database Logging

Logging	Type	Description
db_artrival_rate_per_sec	Float	Number of SQL insert/update requests coming per second. Requests usually come from sensors which are enabled for logging.
db_delivery_rate_per_sec	Float	Number of SQL insert/update requests processed by the database.
db_flow	Boolean	Indicates whether database logging is enabled or disabled. The flow is disabled when db_queue_size reaches db_max_queue_limit to prevent memory exhaustion.
db_last_error	String	Last error encountered during SQL logging.
db_latency_ms	Long	Number of milliseconds that an SQL request sits on a queue before being processed by the database.
db_max_queue_limit	Integer	Maximum queue size for holding all outstanding SQL operations.
db_queue_size	Integer	Current number of outstanding SQL requests.
db_total_arrived	Long	Total number of issued SQL requests.
db_total_dropped	Long	Total number of dropped SQL requests. SQL requests are dropped when db_flow is false.
db_total_errors	Long	Total number of SQL related errors encountered during logging.
db_total_processed	Long	Total number of successfully processed SQL requests.

Table 5-4. Sensor Performance

Sensor\Performance	Type	Description
absolute_rate_rules_per_sec	Float	Number of rules processed per second since the start of the CEP server.
absolute_rate_sensors_per_sec	Float	Number of sensors processed per second since the start of the CEP server.
last_sensor_exec_time_ms	Float	Time taken to execute last sensor (in milliseconds).
max_rule_exec_time_ms	Float	Longest time taken to execute a rule.
max_rule_hotspot	String	Name of the rule associated with max_rule_exec_time_ms.
max_sensor_hotspot	String	Name of the sensor associated with max_rule_hotspot.
min_sensor_exec_time_ms	Float	Minimum time taken to execute a sensor.
rate_rules_per_sec	Float	Number of rules processed per second, based on the total time spent processing rules.
rate_sensors_per_sec	Float	Number of sensors processed per second, based on the total time spent processing sensors.
sensor_busy_percent	Float	Percentage of time used to process sensors.
sensor_idle_percent	Float	Percentage of sensors spent in idle state (not processing rules).
sensor_turn_around_time_ms	Long	Time it takes to process a given metric by one or more sensors. The larger the value the longer it takes from the point of metric discovery to it being processed by the sensor and showing up within business views. Growth of this value usually means that the rule engine is overloaded and cannot keep up with rate of fact updates.
total_processed_rules	Long	Total number of processed rules since last reset.
total_processed_sensors	Long	Total number of processed sensors since last reset.
total_sensor_time_ms	Long	Total time spent processing sensors since last reset (in milliseconds).

Table 5-5. Sensor Runtime

Sensor\Runtime	Type	Description
sensor_arrival_rate_per_sec	Float	Number of sensor events queued up for processing per second.
sensor_delivery_rate_per_sec	Float	Number of sensor events processed per second.
sensor_dropped_events	Long	Total number of dropped sensor events due to overload.
sensor_failed_actions	Long	Total number of failed user actions (launched by sensors).
sensor_failed_notifications	Long	Total number of failed notifications (SMTP).
sensor_table_size	Long	Total number of outstanding sensor requests.
sensor_total_action_threads	Integer	Total number of threads allocated to execute user actions.
sensor_total_expiring_threads	Integer	Total number of threads associated with expiring sensors (dynamic child sensors).
sensor_total_notifications	Long	Total number of successfully sent notifications (SMTP).
sensor_total_running	Long	Total number of actively running sensors, including user defined and dynamically created sensors.
sensor_worker_active_threads	Integer	Total number of threads allocated to process all sensors.
sensor_worker_thread_pool_size	Integer	Total number of threads within the thread pool designated for processing rules.

Table 5-6. Sessions

Session	Type	Description
avg_pipe_arrival_rate_per_sec	Float	Average rate at which objects are published for delivery to remote clients (subscribers).
avg_pipe_delivery_rate_per_sec	Float	Average rate at which objects are delivered to remote clients (subscribers).
avg_pipe_usage	Float	Average communication usage across all connections. Multiply by 100 to get % value.
max_resp_time	Long	Maximum response time for remotely executed actions.
max_send_queue_limit	Long	Maximum queue size of holding outbound requests for each communication session (between client/server).
total_bytes_sent	Long	Total number of bytes sent across all connections.
total_objects_dropped	Long	Total number of objects dropped due to communication overload – the remote end unable to keep up with consumption of data.
total_objects_ooband	Long	Total number of requests/replies.
total_objects_recvd	Long	Total number of objects received across all connections.
total_objects_sent	Long	Total number of objects sent across all connections.
total_pipes	Integer	Total number of active communications between client and server.
total_pipes_disabled	Integer	Total number of pipes disabled due to communication overload. Disabled pipes will drop objects.
total_pipes_full	Integer	Total number of connections in full state – 100% capacity utilization – their max queues are utilized to 100%.
total_pipes_idle		Total number of pipes in idle state.
total_requests	Long	Total number of issued requests (commands) to remote locations.
total_resp_timeouts	Long	Total number of timed out requests.
total_send_queue_depth	Long	Total number of objects queued up for outbound delivery.
total_user_timeouts	Long	Total number of timed out user initiated requests.

Table 5-7. Topic/Facts

Topic	Type	Description
expiringFacts	Long	Total number of expiring facts; facts marked for expiration.
fact_delivery_backlog	Long	Total number of facts waiting to be published to corresponding subscribers. Growing number indicates that publishing is at a much higher rate than can be consumed by subscribers.
fact_delivery_turn_around_time_ms	Long	Time it takes to deliver a given fact to a subscriber. Growing number indicates longer latency and delayed processing by the rule engine.
fact_publish_backlog	Long	Number of outstanding facts waiting to be published.
fact_publish_usage	Long	Number of facts that have been published.
fact_request_backlog	Long	Total number of outstanding requests for fact lookup.
maxFacts	Long	Maximum number of facts allowed within the CEP server. The limit is soft and does not prevent publishing more facts.
sensor_delivery_backlog	Long	Total number of sensors waiting to be processed by the rule engine. Growing number indicates that the rule engine is overloaded and may not process information in a timely fashion.
sensor_delivery_limit	Long	Maximum number of allowed sensors that can be in a waiting state before being dropped. Once the sensor_delivery_backlog exceeds this limit, some sensor activity will be dropped to prevent CEP server from consuming large amounts of memory and failing.
sensor_delivery_usage	Long	Total number of sensors that have been delivered.
sensor_request_backlog	Long	Total number of sensors waiting for metric lookup.
sensor_request_limit	Long	Maximum number of allowed sensors that can be requested.
sensor_request_usage	Long	Total number of sensors that have been requested.
totalFacts	Long	Total number of published facts.
totalFilters	Long	Total number of unique filters used by all created sensors.
totalSubscribers	Long	Total number of subscribers, which includes sensors as well as remote subscribers.
utilization	Float	Fact utilization totalFacts/maxFacts. Multiply by 100 to get % used.

This page intentionally left blank.

Chapter 6: M6 for WebSphere MQ

6.1 Prerequisites

The following will make installation and deployment run more efficiently:

- Determine where workgroup server will be installed and the number of nodes reporting to it. If the number of nodes is greater than 50 or determined growth over the next year will double, then consider using multiple workgroup servers. Group nodes logically. For example: geographically or by business units.
- Place CEP servers where the related workgroup server(s) is installed, preferably one CEP server per workgroup server.
- Deploy built-in business views.
- In UNIX, always use the same directory path on all agent machines. This allows the directory files to be copied over from machine to machine. By default, Nastel uses `/opt/nastel` but any directory path can be used.
- When building a Queue Manager, combine Nastel scripts for creating Nastel queues with other existing scripts.

Request from Nastel support a CPU-based license to replace the old license. Current boot license limits the number of deployed services to 50.

6.2 Usage

Use the following guidelines when creating/updating Expert Properties in M6 for WMQ:

- **Expert monitoring interval**
  - If very small monitoring intervals are required, adjust *Discovery Interval* downward, but ensure it is larger than *Sample Duration* maximum value. Usually statistics taken for 24 hours is sufficient to judge minimum and maximum expert intervals.
  - If monitoring scope is changed, look at *Sample Duration* and adjust *Discovery Interval* accordingly. *Sample Duration* information is located in the WebSphere MQ Health policy under Management Infrastructure Status\Monitoring Agents\Expert Duration sensor.
- **Expert fact expiration** – Consider carefully fact expiration interval. It is recommended to enable fact expiration on each expert and it must be greater than the monitoring interval. The interval is calculated as 120%-130% of (*Monitoring Interval* + *Sample Duration*).
- **Expert fact history size** – Do not enable history collection, located under *Fact Options* tab, unless there is good reason to do so. If it is enabled, ensure the memory requirements are sufficient. (Hardware and JVM).
- **Expert logging** – Do not create large logs. It is difficult to view large logs over a remote connection.

Consider polling differences between Universal Agent expert and regular WebSphere MQ experts by referring to the following table.

Table 6-1. Polling Differences

Universal Agent Expert	
PROs	CONs
Information received is normally up-to-date.	Polls information through Nastel monitoring infrastructure directly from WebSphere MQ queue managers and objects. Multiple Universal Agents will not share duplicate information among themselves. Therefore, each Universal Agent will add extra load to the Queue Manager.
Sends one global monitoring request for all objects. Returns individual responses for each object.	Requests are subject to Nastel and MQ security and can be permitted or restricted only on the command level.
Regular WebSphere MQ Experts	
PROs	CONs
Polls information through Nastel Workgroup and its database which offloads Queue Manager from extra monitoring traffic.	Information received may not be up-to-date and is subject to database updates.
Requests are subject to Nastel and MQ security and as granular as command per each object.	Experts send monitoring request for each object and receive back replies for each object which generates more network activity.

6.3 Publish Subscriber

The Publish Subscriber is required to manage multiple workgroups that are using the same node(s) in order to send events to all managing workgroup servers. All workgroups must have unique names.

Chapter 7: Typical AutoPilot Deployments

7.1 Hardware Requirements

The following are typical hardware requirements for single server deployments:

- Windows or UNIX
- 2 to 4 G Memory
- 10 G Disk Space
- 2 – 4 CPUs (64 bit processor recommended).

7.2 Deployments

1. Small/Medium – less than 100 servers

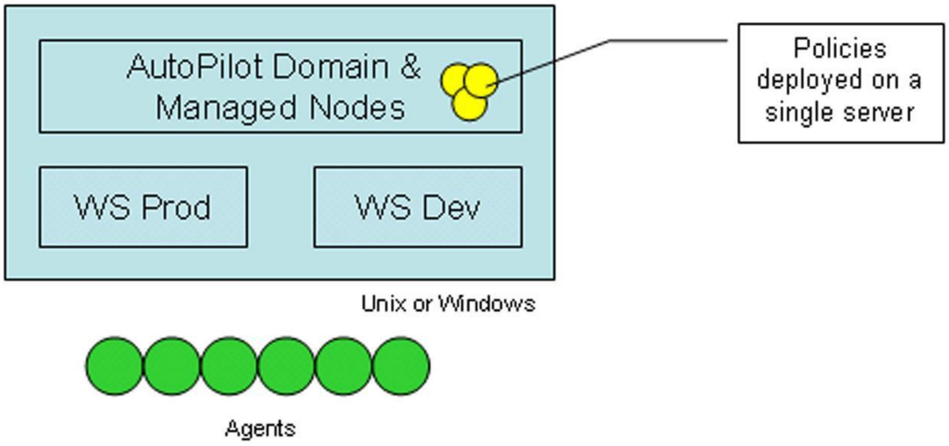

AutoPilot Domain & Managed Nodes
WS Prod
WS Dev
Unix or Windows
Agents
Policies deployed on a single server

Figure 7-1. Small-to-Medium Deployment

2. Medium/Large – great than 100 servers

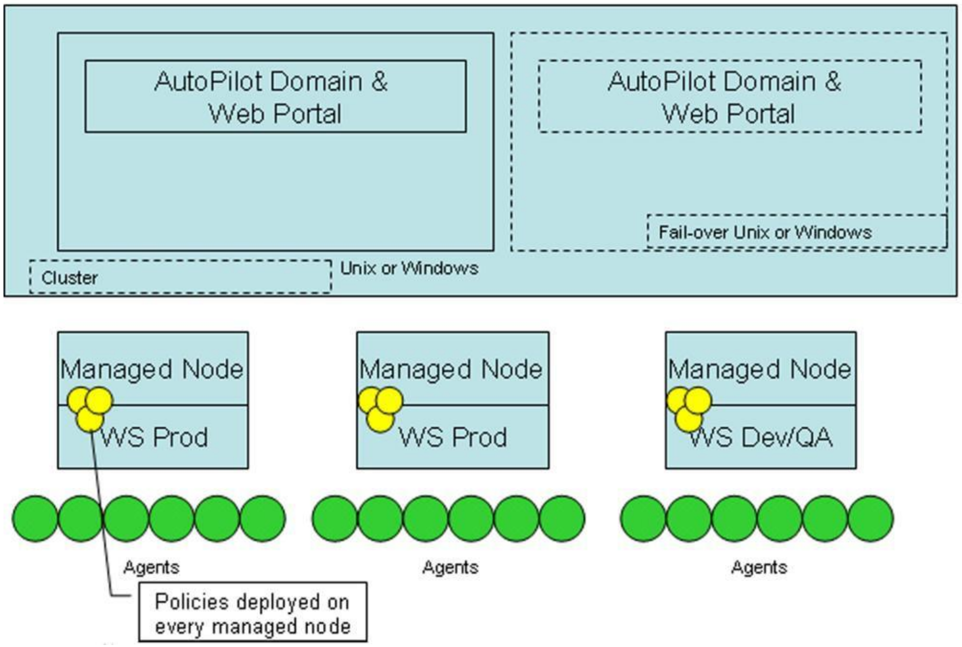

AutoPilot Domain & Web Portal
AutoPilot Domain & Web Portal
Fail-over Unix or Windows
Cluster
Unix or Windows
Managed Node
WS Prod
Managed Node
WS Prod
Managed Node
WS Dev/QA
Agents
Agents
Agents
Policies deployed on every managed node

Figure 7-2. Medium-to-Large Deployment

3. Integration with EMS

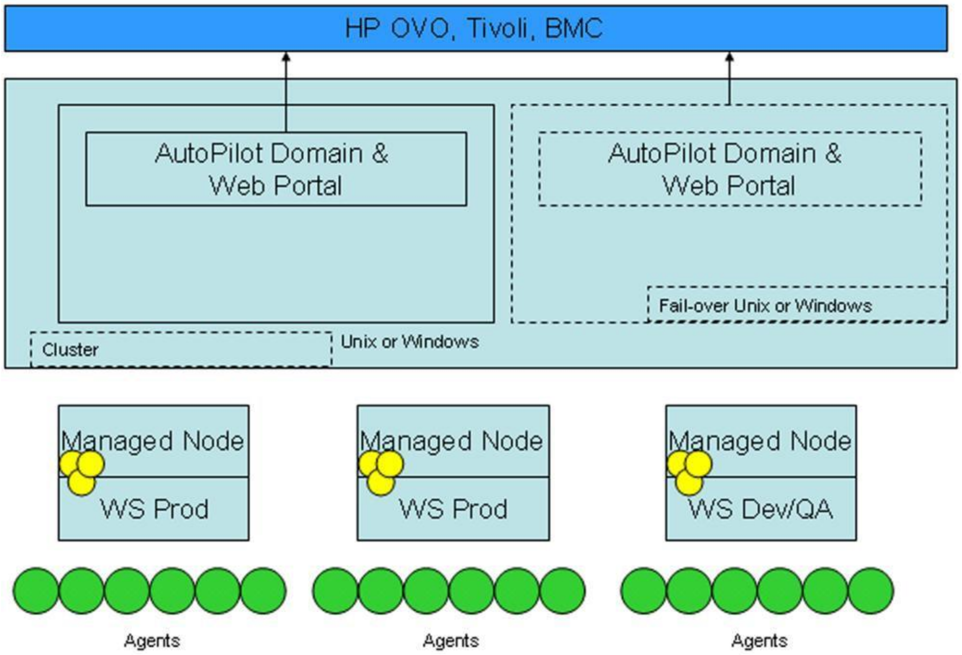


Figure 7-3. Integration with ESM Deployment

4. Multiple Geographies

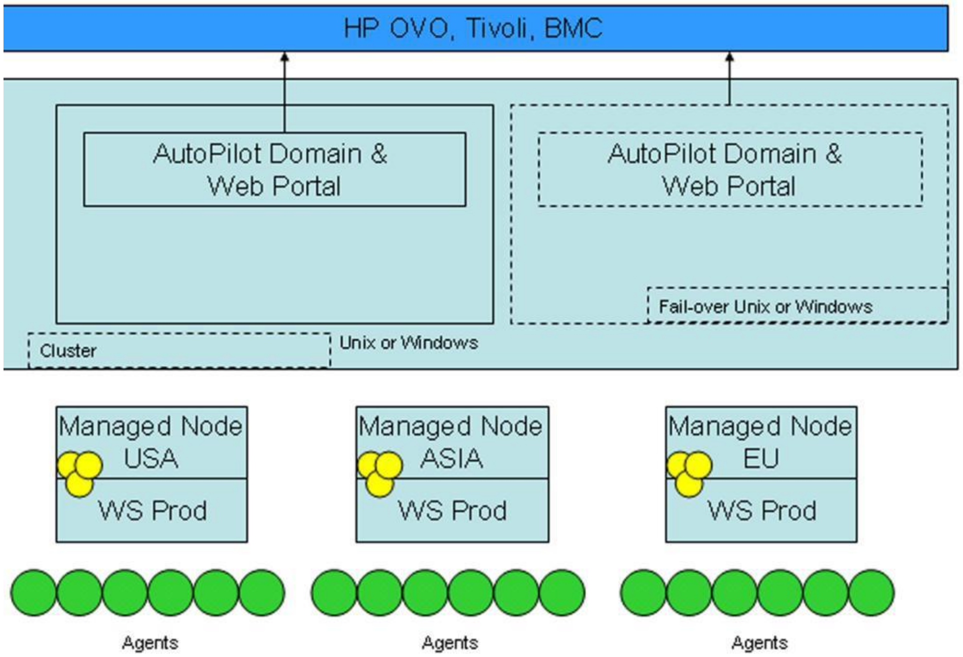


Figure 7-4. Multiple Geographies Deployment

Appendix A: References

A.1 Nastel Documentation

Table A-1. Nastel Documentation

Document Number (or higher)	Title
AP M6/USR 600.007	*Nastel AutoPilot M6 User's Guide*
M6/INS 600.008	*Nastel AutoPilot M6 Installation Guide*
M6/WMQ 600.002	*Nastel AutoPilot M6 Plug-in for WebSphere MQ*
M6WMQ-ADM 653.002	*Nastel AutoPilot M6 for WebSphere MQ Administrator's Guide*
M6WMQ-INS 653.003	*Nastel AutoPilot M6 for WebSphere MQ Installation Guide*
M6WMQ/SM 653.001	*Nastel AutoPilot M6 for WebSphere MQ Security Manager User's Guide*
M6/WMQ 600.002	*Nastel WebSphere MQ Plug-in for AutoPilot M6*

A.2 IBM Documentation

SC33-1872 *WebSphere MQ Intercommunications*

SC33-1369 *WebSphere MQ MQSC Command Reference*

SC34-5456 *WebSphere MQ Using Java*

http://www-306.ibm.com/software/websphere/

A.3 HP OpenView Documentation

http://www8.hp.com/us/en/software/enterprise-software.html?jumpid=&k_clickid=AMS|117|1094|baa40e71-4235-48b5-83d1-8d58a0207f7f

A.4 Java™ 2 J2SE™ for HP-UX Information Library

https://h20392.www2.hp.com/portal/swdepot/displayProductInfo.do?productNumber=HPUXJAVAINFOLIB

A.5 Oracle Online Documentation

http://www.oracle.com/technetwork/developer-tools/forms/documentation/index.html

A.6 Tru64 UNIX Online Documentation and References

www.hp.com/go/tru64unix-docs

This page intentionally left blank.

Appendix B: Conventions

B.1 Typographical Conventions

Table B-1. Typographical Conventions

Convention	Description
Blue/Underlined	Used to identify links to referenced material or websites. Example: support@nastel.com
Bold Print	Used to identify topical headings, glossary entries, and toggle or buttons used in procedural steps. Example: Click **EXIT**.
Italic Print	Used to place emphasis on a title, menu, screen name, or other category.
`Monospaced Bold`	Used to identify keystrokes/data entries, file names, directory names, etc.
`Monospaced Italic`	Used to identify variables in an address location. Example: `[C:\`*`AutoPilot_Home`*`]\documents`. Where the portion of the address in the brackets [ ] is variable.
`Monospaced Text`	Used to identify addresses, commands, scripts, etc.
Normal Text:	Typically used for general text throughout the document.
Table Text	Table text is generally a smaller size to conserve space. 10, 9, and 8 point type is used in tables throughout the AutoPilot product family of documents.

This page intentionally left blank.

Glossary

Asynchronous: Communication between computer and devices that can occur at any time and at irregular intervals.

AutoPilot M6: Nastel Technologies' Enterprise Application Management Platform. AutoPilot M6 is designed to monitor and control distributed IT services such as application servers, middleware, user applications, workflow engines, brokers, Service Oriented Architecture (SOA) and Enterprise Service Bus (ESB) based applications and their impact on business services.

AutoPilot M6 for WMQ: Nastel Technologies' WebSphere MQ management solution. Re-designated as M6 for WMQ with release 6.0, prior releases retain the AP-WMQ or MQControl trademark.

AutoPilot M6 Web: AutoPilot M6 Web is a browser-based interface that provides monitoring and operational control over managed resources and applications. It allows users to monitor health, recover from a failure, view historical performance graphs and visualize impacts of a failure.

AutoPilot M6/WebSphere Message Queue Integrator (AP/WMQI): Formerly AP/MQSI

AutoPilot/Message Tracking (AP/MT): Nastel's AutoPilot/Message Tracking plug-in that enables Nastel AutoPilot to intercept message exits and forward the statistical data to a Nastel AutoPilot expert.

BSV: *see* Business View

Business View (BSV): A collection of rules that define a desired state of an eBusiness environment. Business Views can be tailored to present information in the form most suited to a given user, as defined by the user.

CEP: *see* Complex Event Processing.

CEP Server: A container that can host any number of AutoPilot M6 services such as experts, managers, policies etc.

Client: Any programming component that uses the AutoPilot M6 infrastructure; for example, the AutoPilot M6 User Console.

Common Object Request Broker Architecture (CORBA): A Common Object Request Broker Architecture (CORBA) object can be invoked from a Web browser using CGI scripts or applets.

Complex Event Processing (CEP): A technology for building and managing event-driven information systems. CEP is primarily an event processing concept that deals with the task of processing multiple events from an event cloud with the goal of identifying the meaningful events within the event cloud. CEP employs techniques such as detection of complex patterns of many events, event correlation and abstraction, event hierarchies, and relationships between events such as causality, membership, and timing, and event-driven processes.

Console: The console acts as the graphical interface for AutoPilot M6.

Contacts: A subordinate to a given Manager or Expert.

CORBA: *see* Common Object Request Broker Architecture.

Data Source Name: A Data Source Name (DSN) is the logical name that is used by Open Database Connectivity (ODBC) to refer to the drive and other information that is required to access data. The name is use by Internet Information Services (IIS) for a connection to an ODBC data source, (Example: Microsoft SQL Server database). The ODBC tool in the Control Panel is used to set the DSN. When ODBC DSN entries are used to store the connection string values externally, you simplify the information that is needed in the connection string. This makes changes to the data source completely transparent to the code itself.

Decision Support System (DSS): An AutoPilot M6-based service designed to monitor, store, and display any event information generated by AutoPilot M6-enabled middleware and applications.

Deploy: To put to use, to position for use or action.

Domain Server: A specialized CEP server that maintains the directory of CEP servers, experts etc. The domain server is also capable of hosting experts, managers etc.

DSN: *see* Data Source Name

DSS: *see* Decision Support System

Enterprise Service Bus (ESB): A software architecture construct, implemented by technologies found in a category of middleware infrastructure products usually based on standards that provides foundational services for more complex architectures via an event-driven and standards-based messaging engine (bus).

ESB: *see* Enterprise Service Bus

EVT: Event Log file extension (e.g.: sample.evt)

Event: An Event is something that happens to an object. Events are logged by AutoPilot M6 and are available for use by AutoPilot M6 Policies or the user.

Expert: Services that monitor specific applications such as an applications server, web-server or specific components within the applications (example - channels in WebSphere MQ). Experts generate facts.

Facts: Facts are single pieces of data that have a unique name and value. One or more facts are used to determine the health of the object, application or server.

File Monitor: Built-in expert that monitors application error logs and publishes logs to AutoPilot M6 as facts and events.

Graphical User Interface (GUI): A type of environment that represents programs, files, and options by means of icons, menus, and dialog boxes on the screen. The user can select and activate these options by pointing and clicking with a mouse or, often, with the keyboard. Because the graphical user interface provides standard software routines to handle these elements and report the user‘s actions (such as a mouse click a particular icon or at a particular location in text, or a key press); applications call these routines with specific parameters rather than attempting to reproduce them from scratch.

Grids: In AutoPilot M6, a collection of clusters that allows users to define and automate CEP server failover. Defined under domain server\naming\grid folder.

GUI: *see* Graphical User Interface.

IIS: *see* Internet Information Services

Internet Information Services (IIS): Microsoft’s brand of Web server software, utilizing HTTP to deliver World Wide Web documents. It incorporates various functions for security, allows CGI programs, and also provides for Gopher and FTP services.

J2EE: *see* Java Platform, Enterprise Edition.

Java: A platform-independent, object-oriented programming language developed and made available by Sun Microsystems.

Java Developer’s Kit (JDK): A set of software tools developed by Sun Microsystems, Inc., for writing Java applets or applications. The kit, which is distributed free, includes a Java compiler, interpreter, debugger, viewer for applets, and documentation.

JDBC: *see* Java Database Connectivity.

Java Database Connectivity (JDBC): The JDBC API provides universal data access from the Java programming language. Using the JDBC 2.0 API, you can access virtually any data source, from relational databases to spreadsheets and flat files. JDBC technology also provides a common base on which tools and alternate interfaces can be built. The JDBC *Test Tool* that was developed by Merant and Sun Microsystems may be used to test drivers, to demonstrate executing queries and getting results, and to teach programmers about the JDBC API.

Java Management Extensions: Java technology that supplies tools for managing and monitoring applications, system objects, devices (e.g. printers) and service oriented networks. Those resources are represented by objects called MBeans (for *Managed Bean*).

Java Message Service: Java Message-oriented middleware API for sending messages between two or more clients.

Java Platform, Enterprise Edition (J2EE): A widely used platform for server programming in the Java programming language. The Java EE Platform differs from the Standard Edition (SE) of Java in that it adds additional libraries which provide functionality to deploy fault-tolerant, distributed, multi-tier Java software, based largely on modular components running on an application server.

Java Server Pages (JSP): JSP technology enables rapid development of web-based applications that are platform independent. Java Server Pages technology separates the user interface from content generation enabling designers to change the overall page layout without altering the underlying dynamic content. Java Server Pages technology is an extension of the Java Servlet technology.

Java Virtual Machine (JVM): The virtual operating system that JAVA-written programs run. The JVM is a hardware- and operating system-independent abstract computing machine and execution environment. Java programs execute in the JVM where they are protected from malicious programs and have a small compiled footprint.

JDK: *see* Java Developer‘s Kit.

JMS: *see* Java Message Service.

JMX: *see* Java Management Extensions.

JRE: JAVA Run-time Environment. The minimum core JAVA required to run JAVA Programs.

JSP: *see* Java Server Pages

JVM: *see* JAVA Virtual Machine.

LDAP: *see* Lightweight Directory Access Protocol.

Lightweight Directory Access Protocol: A networking protocol for querying and modifying directory services running over TCP/IP.

Managed Node: A container that can host any number of AutoPilot services such as experts, managers, policies, etc. Redesignated as CEP Server after AutoPilot M6 with Service Update 4.

Manager: Managers are the home or container for policies. All business views must reside on managers, and managers must be deployed prior to deploying a business view or policy.

Message-Oriented Middleware (MOM): Message-oriented middleware (MOM) is a client/server infrastructure that increases the interoperability, portability, and flexibility of an application by allowing the application to be distributed over multiple heterogeneous platforms.

Message Queue Interface: The Message Queue Interface (MQI) is part of IBM‘s Networking Blueprint. It is a method of program-to-program communication suitable for connecting independent and potentially non-concurrent distributed applications.

MOM: *see* Message-Oriented Middleware.

MQI: *see* Message Queue Interface

Naming Service: A common server records names of objects and associates them with references, locations and properties.

ORB: Object Request Broker.

Orbix: CORBA product distributed by IONA Technologies.

Package Manager: The command line utility that allows users to list, install, uninstall, verify and update AutoPilot M6 installation on any CEP Server.

PKGMAN: *see* Package Manager

Policy/Business Views: Business views are a collection of one or more sensors. Business views are used to visually present the health and status of the different systems as well as automatically issue remedial actions.

Process Wrapper: Built-in wrapper that monitors a process or script started by AutoPilot M6.

Sensor: A rule that is used to determine the health of an object or application based on one or more facts. Actions can then be issued, based on the health.

Service-Oriented Architecture (SOA): An evolution of distributed computing and modular programming. SOAs build applications out of software services. Services are relatively large, intrinsically unassociated units of functionality, which have no calls to each other embedded in them. They typically implement functionalities most humans would recognize as a service, such as filling out an online application for an account, viewing an online bank statement, or placing an online book or airline ticket order. Instead of services embedding calls to each other in their source code, protocols are defined which describe how one or more services can talk to each other. This architecture then relies on a business process expert to link and sequence services, in a process known as orchestration, to meet a new or existing business system requirement.

Simple Mail Transfer Protocol (SMTP): A TCP/IP protocol for sending messages from one computer to another on a network. This protocol is used on the Internet to route e-mail. *See also* communications protocol, TCP/IP. *Compare* CCITT X series, Post Office Protocol.

SMTP: *see* Simple Mail Transfer Protocol

SOA: *see* Service-Oriented Architecture

Synchronous: communication within a computer that occurs at regular intervals. Usually governed by the microprocessor clock.

TCP/IP: *see* Transmission Control Protocol/Internet Protocol.

Transmission Control Protocol/Internet Protocol (TCP/IP): A protocol developed by the Department of Defense for communications between computers. It is built into the UNIX system and has become the de facto standard for data transmission over networks, including the Internet.

UDP: *see* Universal Datagram Protocol.

Universal Datagram Protocol (UDP): A connectionless protocol that runs on top of IP networks. Provides very few error recovery services, offering instead a direct way to send and receive datagrams over an IP network. It is used primarily for broadcasting messages over a network.

Virtual Machine (VM): Software that mimics the performance of a hardware device, such as a program that allows applications written for an Intel processor to be run on a Motorola chip. *Also See* Java Virtual Machine.

Visual Source Safe (VSS): Microsoft VSS helps you manage your projects by saving them to a database. When you need to share files between two or more projects, you can share them quickly and efficiently. When you add a file to VSS, the file is backed up on the database, made available to other people, and changes that have been made to the file are saved so you can recover an old version at any time. Members of your team can see the latest version of any file, make changes, and save a new version in the database.

VM: *see* Virtual Machine

VSS: *see* Visual Source Safe.

WebSphere MQ: IBM's message queuing product.

Websphere_MQ_Manager: A specialized manager capable of hosting one or more MQSeries specific policies, apart from the regular policies.

Wireless Application Protocol (WAP): An open global specification that is used by most mobile telephone manufacturers. WAP determines how wireless devices utilize Internet content and other services. WAP enables devices to link diverse systems contents and controls.

XEN: A free software virtual machine monitor for IA-32, x86-64, IA-64 and PowerPC architectures. It is software that runs on a host operating system and allows several guest operating systems to be run on top of the host on the same computer hardware at the same time. Modified versions of Linux and NetBSD can be used as hosts.