Operations Guide

Release 5.5

Model N

THE LEADER IN REVENUE MANAGEMENT

Model N® Revenue Management Application, Release 5.5

Trademarks

Document Revised: March 10, 2011

Contact Model N
Model N
1800 Bridge Parkway
Redwood City, CA 94065

Phone: (650) 610 - 4600
FAX: (650) 610 - 4699

www.modeln.com

Table of Contents

List of Figures

List of Code Examples

List of Tables

1 About This Guide

This administrator's guide provides information on how to monitor and configure the Model N application stack - the application server, the web server, the Model N system, the database, and the Cognos server - to ensure optimal performance.

1.1 Audience

This guide is intended for the following audiences:

- System Administrators
- Those in IT Operations who are responsible for the ongoing maintenance of your Model N system

1.2 Using This Guide

This guide provides you with an overview of the Model N application stack, and system architecture, and explains how you can monitor and configure them for optimal performance. Read the Introduction chapter to become familiar with the Model N system.

1.3 Document Conventions

The following table describes the conventions used in this guide.

Table 1-1: Conventions

Conventions	Descriptions
Field and dialog box names	Names of fields and dialog boxes are first letter (of each word) capitalized, with no other special formatting. For example: Information in the Customer ID field must be valid. Information in the External Data dialog box are read-only.
User input	User input is in **bold** type, whether the input is typed or selected from a menu, or a button or similar element is clicked on a page. For example: Type **Exit**, and then press **Enter**. On the **File** menu, click **Save**. On the tool bar, select **Edit > Preferences**.
Navigation Paths	Each application page description begins with a navigation path written in **bold**. For example: Navigation Path: **Sales Folders > Indirect Sales > Submission # > Lines**
Line breaks in code	Some code examples may contain lines which are longer than can be displayed on the width of a page. Lines which have been broken for page formatting reasons will be indicated with the pilcrow (¶) symbol. Do not separate these lines during actual configuration because doing so may cause the application to fail to initialize correctly.
File names and code objects	Names of files and code objects are in `Courier New` font style.
System outputs	System outputs, such as confirmation messages or alerts, are first letter capitalized. For example: The system displays the following message: Do you want to save your changes before moving on?
Referenced topics	Referenced topic headings are in blue and underlined. For example: See Introduction for more information.
Referenced guides/documents	Referenced guide or document titles are in *italics*. For example: See the *Medicaid Developer's Guide* for more information.

Table 1-1: Conventions (Continued)

Conventions	Descriptions
Lifecycle statuses	The lifecycle statuses are in `Courier New` font style. For example: Users cannot override the `Ignore` status or the `Fatal` severity level.
Notes	Notes are contained within two horizontal lines. For example: This is a note.
Blank pages	There will be a blank page with header and footer at the end of the chapters ending on odd page numbers. This format is designed to accommodate printing multiple chapters on duplex paper and keeping the collation even.

1.4 Related Documents/Resources

The following related documents are available:

- *Installation and Migration Guide*
- *Application Administration Guide*
- *Release Notes*

1.5 Feedback

Model N welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number (if available). You can send comments to us in the following ways:

- E-mail: techdocs@modeln.com
- FAX: (650) 610 - 4699. Attn: Product Documentation
- Postal service:

  Model N

  Product Documentation

1800 Bridge Parkway,

Redwood City, CA 94065

At Model N, we are dedicated to improving the technical documentation. We are not able to reply to your comments individually, but we use your advice to improve our services.

2 Introduction

This document provides background information on the Model N application deployment architecture, and information on how to monitor the performance of the different components that comprise the Model N application stack (the web server, the application server, the Cognos ReportNet server, and the Oracle database) to ensure that the Model N system is performing at an optimal level. Query performance is especially important when the database size grows significantly. In addition, this document provides metrics you can use to measure the Model N system's performance as well as a chapter on how to promote a new version of the Model N system from one environment, such as testing, to another, such as production.

To understand the information provided, you should be familiar with database administration and hardware performance tuning fundamentals, as well as with the version of the Oracle database that you are using, and XML.

2.1 System Requirements

To run the Model N system most efficiently, you should first determine the amount of data that Model N will be accessing from the database during deployment, the number of application users, the number of requests for data, and the acceptable response time.

2.2 Concepts and Terms

The following table summarizes some key terminology used in this document.

Table 2-1: Terminology

Term	Definition
Configuration Console	The user interface for managing configurations such as business object configurations (FGOs), enumerations, and application switches for the entire Model N system.
Configuration Set	A logical collection of configuration values that can be activated or deactivated as a set. Also called "Configurations".
Initial Configuration	The name given to the out-of-the-box product configuration set. This configuration set is not editable through the user interface.
Configurations	A logical collection of configuration values that can be activated or deactivated as a set. Also called "Configuration Set".

2.3 References to Other Documentation

This *Operations Guide* provides system administrators with the guidelines and background information necessary to successfully monitor the Model N system and to maximize its performance. It does not provide comprehensive information on how to tune or monitor other components that comprise the Model N stack.

For information on a particular third-party component, refer to that party's documentation. The following list provides links to some documentation from third-party providers that may be applicable.

Note: Model N is not responsible for the availability of third-party websites mentioned in this document. Model N does not endorse and is not responsible or liable for any content on or made available though said websites. Model N will not be responsible or liable for any damage or loss, actual or otherwise, caused by or in connection with use of or reliance on any such content.

- Sun

  http://docs.sun.com
- Oracle WebLogic

  http://www.oracle.com/technology/documentation/bea.html
- Oracle 10g Database

  http://www.oracle.com/pls/db10g/homepage
- Cognos ReportNet

See the *Cognos Administration and Security Guide* that is provided along with Model N's documentation.

- IBM WebSphere

  http://publib.boulder.ibm.com/infocenter/wasinfo/v7r0/index.jsp?topic=/com

3 Application Deployment Architecture

The Model N system is divided into the following:

- Application

  The logic residing in the J2EE application server that directly supports requests from the users' browsers.

- Background Processing

  The logic supporting any asynchronous background activity that was scheduled or triggered by user activity.

- Cognos Reporting

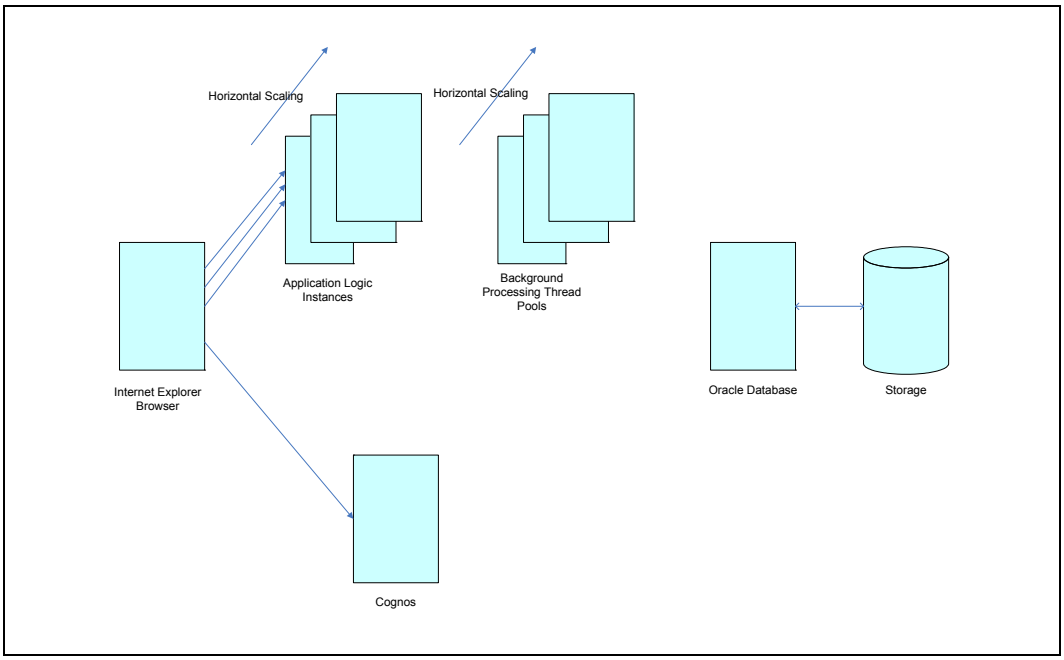
  The Cognos server used to execute and render reports.

- Database

  The Oracle database that supports the data for both the Model N system and Cognos Content Store.

The following diagram shows the deployment system architecture.

Figure 3-1: Deployment Architecture



3.1 Application

The number of application instances required depends on the number of users that are logged into the system concurrently. The capacity that a single application instance can support is based on the hardware capacity, the applications deployed, and the types of user activity. If the application instances appear resource bound (CPU, memory, I/O), then additional application instances can be added (as in horizontal scaling).

3.2 Background Processing

Background processing within the Model N system is executed within thread pools defined in the application instances. These background jobs can cause load on the application instance, the database, or both. Other than integration data flows and sales resubmissions, most jobs typically do not impose a significant load on the application instance since much of the processing is performed in the database.

Each application instance can configure which thread pools are allowed to run in that instance. In addition, specific background jobs can be configured to run only in specific thread pools. The number of threads per thread pool and the number of application instances gives the deployment control over scaling the background processing.

For more information on managing thread pools, see Thread Pool Management.

Because the Model N program logic runs within a J2EE application server, the application instances are always configured to be able to respond to user requests. To prevent an application instance from servicing user requests, do not include the application in the web server or load balancer configuration.

3.3 Cognos Reporting

Cognos ReportNet is a Java-based web application and must run within a Java application server (Tomcat, WebLogic, WebSphere). Cognos ReportNet comes prepackaged with its own embedded version of Tomcat. As of the 5.2 release, Model N only supports a single Cognos application instance which resides on a separate server instance. However, Cognos clustering is possible and is covered in Cognos' documentation.

3.4 Database

The Model N system must run against a single Oracle database instance. The capacity of the database server depends on many factors, such as the number of concurrent users, amount of background activity, as well as the hardware involved. As the needs of the Model N system grows, the capacity of the database server needs to grow to match (as in vertical scaling) which could mean scaling such things as the memory, number of CPUs, or storage capacity.

4 Monitoring

This chapter provides guidelines on areas of the Model N system that should be monitored for performance. Presented here are the metrics that should be measured on a regular basis and information on how to understand how certain metrics scale.

This chapter focuses on performance from a user's point of view including the metrics that can be leading indicators of performance and scalability issues. The metrics are either related to user experience such as page response times and pricing alerts response time, or system performance requirements (for example, inbound data flows must complete within an allocated time window, and the number of concurrent users logged in).

The following sections are covered in this chapter:

In a production system, many activities run concurrently and any single metric is usually influenced by a number of factors. Because of these interdependencies, it is often difficult to accurately determine why a metric is increasing. View the metrics explained in this chapter as the first step in identifying a problem. To obtain full diagnosis, you may need a more detailed analysis.

See the Monitoring Guidelines appendix for additional information.

4.1 Metrics

This section describes aspects of the system that you should measure frequently and regularly to gain an understanding of the health of the system as well as to determine whether the system is moving towards limitations. Included with the measurement definition is a description of how the measurement can be made. When the data already resides in Model N tables, a query is described that can be used to provide the measurement. Otherwise, recommendations are given on how the data can be gathered.

4.1.1 Assumptions

The following assumptions are made regarding the monitoring guidelines:

- These metrics are monitored frequently enough to detect changes in the system before they become a serious problem, but with a large enough period to provide statistical significance to the results.

  As a practical matter, the period should not be too frequent to provide a significant burden, so a period in the range of a week to a month seems reasonable.

  Note: Metrics in this document assume a period of a week is used. In the queries, this is identified by the `task_start_date` column having a restriction between `SYSTIMESTAMP - 7` and `SYSTIMESTAMP`.

- The Performance Monitoring Application (PMA) is available and enabled.
  - PMA Task Purging: Since the PMA purges task data at regular intervals, the PMA must be configured to hold the task data at least as long at the monitoring period. The default PMA purging time is seven days.
  - Resource Keys: For each task, which could be a UI request or background activity, a resource key identifies what that task is doing. For the UI request, the resource key identifies the location within the user interface as well as the specific action taken. For background activities, the resource key identifies the timer or command being run.
- All times are in seconds and rounded to two decimal places unless explicitly stated otherwise.

4.1.2 Application Response Time

The response time for user web requests is a good indicator of system scalability. Because the Model N applications generally manage and process significant amounts of data, a common cause of increased response time is that either the number of queries is increasing or the query response times are increasing.

Note: You should measure the response times for all pages in the application and review them as a function of time of day and number of session on the system.

Following is the list of common request types in the application.

Table 4-1: Common Request Types

Request Type	Description
Login	The user clicks the login button after entering a username and password. • Constructs the user access control list (ACL). • Builds the UI component tree.
Navigation	These requests occur when the user navigates around the application without executing a specific action. • Navigation itself is lightweight, but the target page may execute a number of queries to render.
Search Action	These requests occur when the user clicks the **Search** button in any of the search pages. • Search screens typically translate into queries.
Detail Action	These requests occur when the user clicks a task while looking at the details of a particular object. • Standard tasks (for example, save, or validate). • Lifecycle Actions (for example, submit for approval, activate, or implement).
Dialog Action	These requests generally occur when the user clicks **OK** on a dialog box. The more complex dialog actions usually occur for creation dialog boxs (for example, create a new contract) or import dialog boxs (for example, import a membership file). • Create new document (for example, price list, or contract). • Choosers (for example, select products, or select customers).
Download	These requests occur when the user chooses to export some data through the UI. • Export data from tables. • Printing contracts.

Table 4-1: Common Request Types (Continued)

Request Type	Description
Report Action	These requests occur when the user runs a report.

4.1.2.1 User Response Time Levels

To provide a summary of the performance of the application for the users, the response time percentiles should be measured. A percentile indicates the percentage of responses that were lower than the given value. For example, if the 90th percentile value was 1.25 seconds, this means that 90% of the requests had a response time of 1.25 seconds or less.

A useful set of numbers is the 50th (median), 90th, and 100th (maximum) percentiles in the past period.

Query

The following query lists the number of user requests, 50th, 90th, and 100th percentiles over the past week.

Code 4-1: Query for Response Time Levels

```
SELECT
  COUNT(*) AS cnt,
ROUND(PERCENTILE_CONT(0.50)
    WITHIN GROUP (ORDER BY server_time ASC)/1000, 2) AS pct50,
ROUND(PERCENTILE_CONT(0.90)
    WITHIN GROUP (ORDER BY server_time ASC)/1000, 2) AS pct90,
  ROUND(MAX(server_time)/1000, 2) AS pct100
FROM
  mn_hist_task
WHERE
  thread_type = 'UserRequest'
  AND task_start_date BETWEEN SYSTIMESTAMP - 7 AND SYSTIMESTAMP
;
```

4.1.2.2 Top Response Times by User Action

The longest user requests may indicate performance problems in the application. The top user requests over a given threshold time should be measured for the given monitoring period.

Query

The following query lists the longest user requests greater than 10 seconds during the past week.

Code 4-2: Query for Top Response Times by User Action with product_area

```
SELECT
  *
FROM
```

Code 4-2: Query for Top Response Times by User Action with product_area (Continued)

```
  (SELECT
     REGEXP_REPLACE(
       REGEXP_REPLACE(
         REGEXP_REPLACE(
           resource_key,
           '(root.app.)?root.main.bodyComp(.documentFrame)?.',''),
         '^([^\.]+)\..*', '\1'),
       '(.*)\[.*', '\1', 1, 1, 'n')     AS product_area,
     resource_key                       AS resource_key,
     COUNT(*)                           AS cnt,
     ROUND(MIN(server_time)/1000, 2)    AS min_server_time,
     ROUND(MAX(server_time)/1000, 2)    AS max_server_time,
     ROUND(AVG(server_time)/1000, 2)    AS avg_server_time,
     ROUND(STDDEV(server_time)/1000, 2) AS stddev_server_time
   FROM
     mn_hist_task
   WHERE
     thread_type = 'UserRequest'
     AND task_start_date BETWEEN SYSTIMESTAMP - 7 AND SYSTIMESTAMP
   GROUP BY
     resource_key
   ORDER BY
     ROUND(MAX(server_time)/1000, 2) DESC
   ) temp
WHERE
   max_server_time > 10
;
```

The `product_area` column provides an indication of which area of the application the request comes from (such as contracts, rebates, or admin). The `resource_key` provides the specific detail on exactly where in that application the request originates.

4.1.3 Application Server Memory Usage

Each session in the Model N system caches a number of objects to support the user experience. The more concurrent sessions logged into the application, the more memory is consumed. Once the memory reaches close to the limits, performance can degrade significantly due to issues such as memory fragmentation.

You should monitor the memory usage within the JVM heap, which you can do by using JVM garbage collection logging. You should note the actual used memory and not include memory from objects that simply have not yet been garbage collected. Both the Sun and IBM JVMs have the following for reporting this information.

IBM JVM
The GCCollector Diagnostic Tool for Java Garbage Collector displays garbage collection activity by analyzing GC log files.

The IBM Pattern Modeling and Analysis Tool for Java Garbage Collector displays patterns in garbage collection activity by analyzing GC log files.
Sun JVM
HPJTune by Hewlett-Packard
HPJMeter by Hewlett-Packard

Because memory usage is a very coarse measurement of the system, it is important to correlate the memory usage with other activity on the system. Model N recommends monitoring the number of session and background activity with enough precision to correlate the memory measurements in time.

4.1.4 Background Processing Response Times

The background processing load is the load on the system that has either been triggered from the scheduler (for example, periodically scheduled timers) or triggered from some user action to run in the background. Examples of these are scheduled data flows, Bid Award generation, membership publishing, price monitoring, and materialized view refreshes. Typically, these activities process larger amounts of data and, thus, they tend to be more sensitive as the system scale increases.

Similar to user response time, the response time for each background activity is a good indicator of system scalability. In many areas, specific logging is performed so that the response time can be evaluated not only for the entire activity, but down to each query as well (for example, Bid Awards, Price Monitoring, and Government Pricing calculations).

The response times should be measured for all background activities in the system. In many cases, these can be determined via specific logging configurations. The `mn_command_stats` table records background activities and provides the following information:

- average database elapsed time
- average application elapsed time
- average total time
- execution count.

For data flows, the response time should be reviewed as a function of number of lines input or output. Ideally, data flow monitoring would measure number of lines per hour.

For other heavy processing (for example, materialized view refreshes or bucketing queries), correlating the results with the scope of the change (for example, number of rows inserted, updated, or deleted) is ideal.

4.1.4.1 Monitoring Background Activity

Background activity is executed through timers and commands. These activities can be scheduled to be run periodically (such as nightly dataf lows) or executed immediately as the result of a user action (such as calculate a Government Pricing workbook).

4.1.4.2 Response Time Levels by Activity

As with the user requests, the background activity response time levels should be monitored. Since each background activity can have significantly different durations, the response times should be segmented by the activity type (such as GP calculation versus membership publishing).

Query

The following query returns the number of runs, 50th, 90th, and 100th percentiles for each background activity over the past week. The results are ordered such that the longest duration activity is at the top.

Code 4-3: Query for Response Time Levels by Activity

```
SELECT
  resource_key AS resource_key,
  class        AS class,
  COUNT(*)     AS cnt,
ROUND(PERCENTILE_CONT(0.50)
    WITHIN GROUP (ORDER BY server_time ASC)/1000, 2) AS pct50,
ROUND(PERCENTILE_CONT(0.90)
    WITHIN GROUP (ORDER BY server_time ASC)/1000, 2) AS pct90,
  ROUND(MAX(server_time)/1000, 2) AS pct100
FROM
  mn_hist_task
WHERE
  thread_type IN ('TimerService','Command')
  AND task_start_date BETWEEN SYSTIMESTAMP - 7 AND SYSTIMESTAMP
GROUP BY
  resource_key,
  class
ORDER BY
  ROUND(MAX(server_time)/1000, 2) DESC
;
```

Note: The structure of the `resource_key` column is different than for UI requests, so the `product_area` column should not be used here.

4.1.4.3 Top Response Times by Activity

The longest running background activities may provide insight into where the performance bottlenecks in the system reside. This metric is similar to that for user requests except that the results are segmented by activity.

Query

The following query lists the background activities during the past week grouped by activity. Since background activities are generally longer running than user requests, we look only for activities that have a maximum duration of at least 60 seconds. The results are ordered such that the longest duration activity returns first.

Code 4-4: Query for Top Response Times by Activity

```
SELECT
  *
FROM
 (SELECT
    resource_key                       AS resource_key,
    class                              AS class,
    COUNT(*)                           AS cnt,
    ROUND(MIN(server_time)/1000, 2)    AS min_server_time,
    ROUND(MAX(server_time)/1000, 2)    AS max_server_time,
    ROUND(AVG(server_time)/1000, 2)    AS avg_server_time,
    ROUND(STDDEV(server_time)/1000, 2) AS stddev_server_time
  FROM
    mn_hist_task
  WHERE
    thread_type IN ('TimerService','Command')
    AND task_start_date BETWEEN SYSTIMESTAMP - 7 AND SYSTIMESTAMP
  GROUP BY
    resource_key,
    class
  ORDER BY
    ROUND(MAX(server_time)/1000, 2) DESC
  ) temp
WHERE
  max_server_time > 60
;
```

Response Time Trend

To understand if there are activities in the system that have degraded significantly since the system went live, the response time trend for each background activity is important to monitor. For example, a large difference between the Go-Live and current response times could indicate queries that have slowed down as the volume of data in the system has increased.

Since the PMA does not currently support persisting this information for longer than the purging time period, the Top Response Times by Activity data should be stored per monitoring period (for example, a week) and reviewed for increasing response times.

4.1.5 Real-Time Pricing Engine Call Response Times

The real-time pricing engine resides in the database as a Java stored procedure. The Java stored procedure constructs the appropriate price resolution query that is executed. The query is a relatively expensive call in terms of database resources. Therefore, if the call volume grows, it may impact the response time of the call itself as well as slow down other aspects of the system.

The response time for the real time pricing engine call should be monitored and reviewed as a function of number of concurrent calls as well as by database statistics (for example, CPU, memory, and I/O utilization). Since the real-time pricing call can price multiple lines, recording the number of lines priced is useful for understanding the performance characteristics of the line batching.

4.2 Analyzing the System Using Statistics

To understand the performance characteristics of an application, it is often useful to collect various key statistics over a period of time. When analyzed together, you can use these individual components to understand the resource requirements of the Model N application. The following figures show how the number of user sessions influences JVM memory allocation (garbage collection) and processor load (CPU activity). You can use these types of performance characteristics to determine whether system performance has degraded and what factors have contributed to the degradation.

Figure 4-1: JVM Memory Allocation

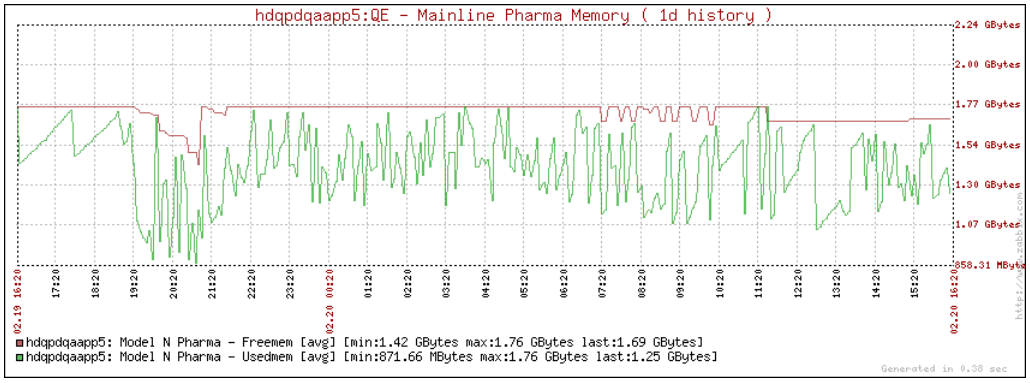


Figure 4-2: Number of Sessions

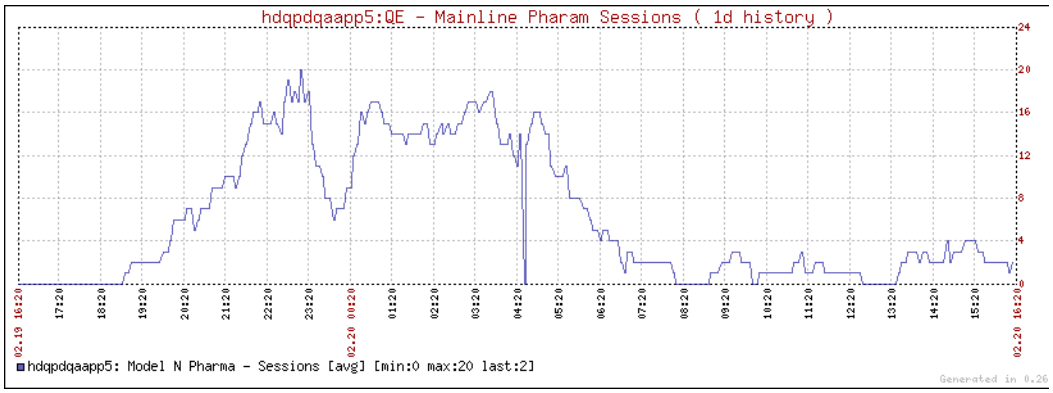
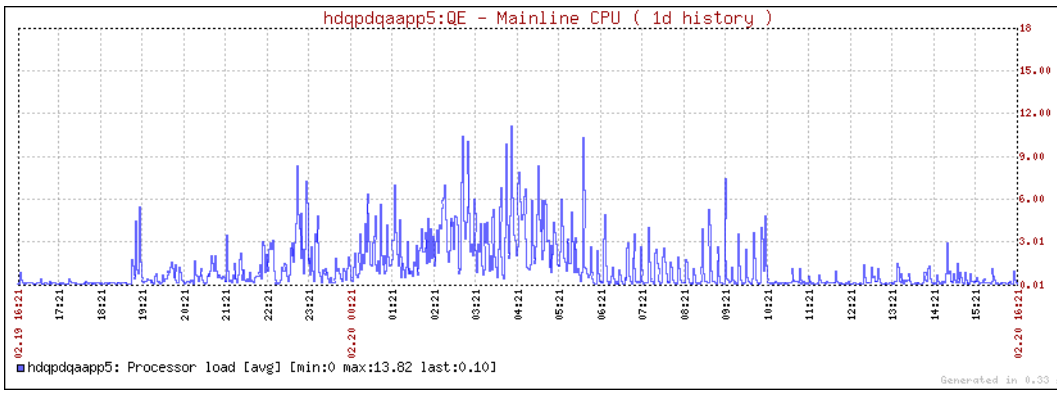


Figure 4-3: Processor Load

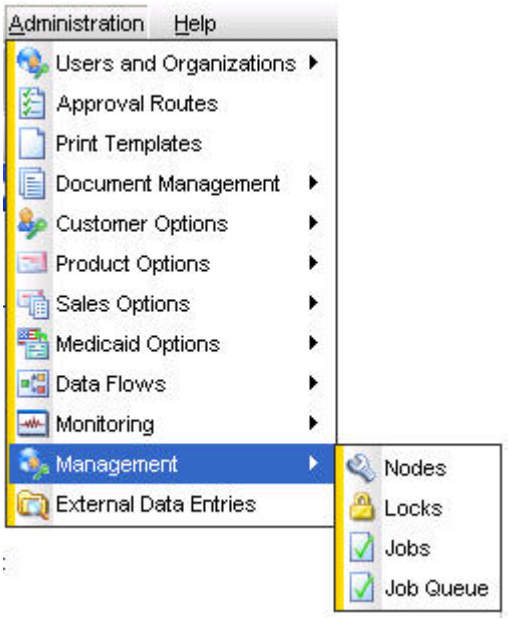


4.3 Management Console

The Management Console provides the tools necessary to perform the following daily operational tasks on Model N applications:

- gain visibility into background jobs being carried out in the application
- obtain logs for debugging operational issues
- gain visibility into the details of job executions of jobs

The Management Console is accessible to system administrators in the Model N community. To access the Management Console, select one of the options available under the **Administration > Management** menu as shown in the following screenshot.



Model N applications can be hosted by deploying one or more servers as part of a deployment. Each instance of the server is referred to as a node. A cluster is the collection of all nodes in a deployed configuration.

The Management Console provides functionality that applies to components associated with the nodes as well as those that apply to the cluster as a whole.

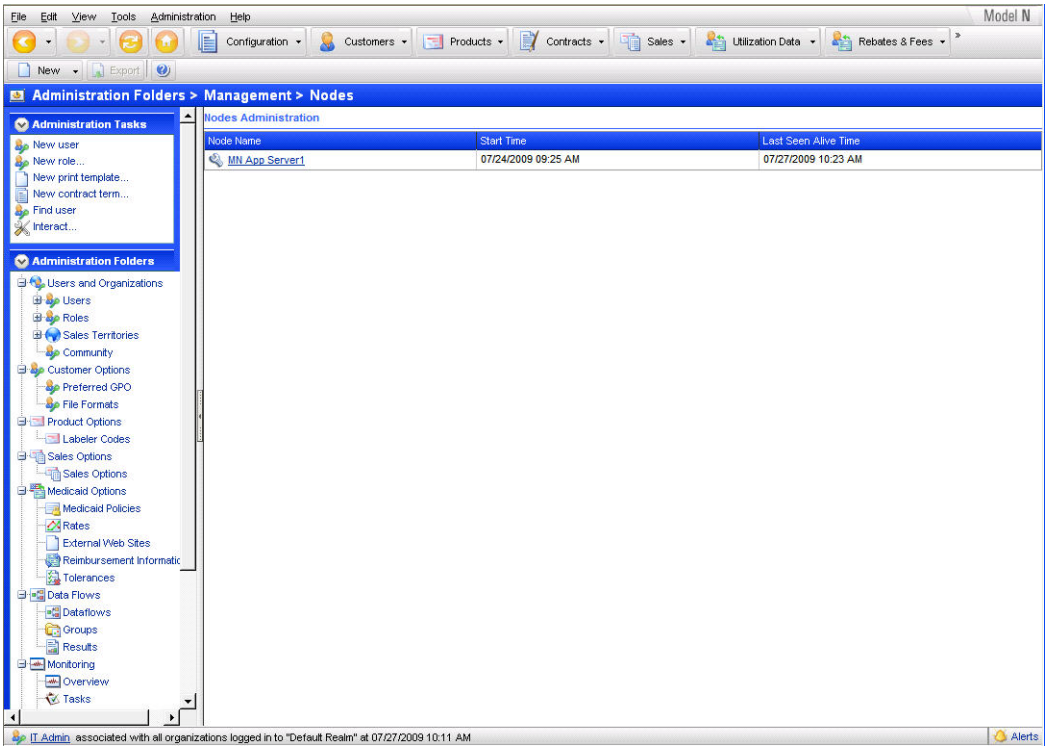
4.4 Node-Based Management

Node-based management lets you identify individual nodes in a Model N application cluster and operate on them. To see the list of nodes that form the Model N cluster, select the Nodes option under the Administration menu. The list of nodes also shows the time when the node was started and the last time when it was known to be functional within the cluster. The Model N architecture lets you dynamically add or remove nodes in the application cluster based on the load demands on the application.

4.4.1 Nodes Page

Navigation Path: **Administration > Management > Nodes**

Figure 4-4: Nodes Page



The Nodes page shows all nodes that are either part of the cluster at that instant or have been in the past. To manage an individual node, click the name of the node in the list.

Table 4-2: Nodes Page

Name	Type	Description
Nodes Administration		
Node Name	Link	Name of an individual node in a Model N clustered environment.
Start Time	Date	The time when the node was started.
Last Seen Alive Time	Date	The last time when the node was known to be functional within the cluster.

4.4.2 Configuring an Application Server to be a Front-End Server

To configure an application server instance to be a front-end server only, indicate that there are no timers or commands that should run in that JVM by setting the following properties in the **Administration > Nodes > Node > Job Designation** page:

- `com.modeln.CommandSvc.filter.mode=INCLUDE`
- `com.modeln.CommandSvc.filter.list=`

4.4.3 Configuring an Application Server for Back-End Processing

To configure an application server instance to handle only back-end processing, you do not need to explicitly tell it anything. This JVM is identical to the JVMs that handle user interface (UI) requests (it still runs within the Servlet Container). To prevent any UI requests from reaching the application server instance, configure the front-end web server so that it includes this JVM in the set of application servers that will handle UI requests.

Note: Even if a single UI request reaches this JVM instance, the ACF definition tree may consume hundreds of megabytes and there will be less memory available for the back-end processing.

4.5 Log Management

Logging lets you direct selective sub-systems of the Model N system to produce output that can help you identify causes of observed erratic behavior in the application. You can direct the output to one of several destinations allowed by the logging sub-system. The Log Management Console lets you view and make changes to the log settings on a selected node. Levels associated with logging let you identify the severity level for which log messages are produced. Category-based logging lets you narrow the logging to specific packages where logging output is desired.

You can modify log settings by:

- changing the existing settings
- adding new settings
- removing existing settings

To change the logging levels for a node, select the **Save Node** task from the left navigation. Any changes made to log settings are applied to the node immediately.

If any output setting is selected to be of type File, the File Log Size specification is used to identify the maximum size of the log file. The logging sub-system creates new files to capture log output once the maximum file size has been reached.

In some cases, capturing log output in the file can alter the behavior of the application. This can happen because the time taken to capture the log can alter the relative timing between two concurrent events in the application, resulting in a failure to capture the execution details from the logging sub system. For such occurrences, the Management Console lets you select the log output channel to be of type, Memory, which results in the log statements being collected in an in-memory ring buffer which has minimal latency. Since memory is a scarce resource for the application, the log is captured in a ring buffer. This lets the application restrict the amount of memory that can be consumed by the logging sub system. The Buffer Log Size setting lets you specify the number of lines present in the in-memory ring buffer in which the log is captured.

The following describes what occurs in a typical use case scenario:

1. Go to the Logging page: **Administration > Management > Nodes > Node Name** link.
2. Click **Clear Buffer Log** to clear any previous logging statements in the in-memory log.
3. Enable the desired logging with **Output** set to `Memory`.

   The Save Node task is invoked, which applies the settings to the node.

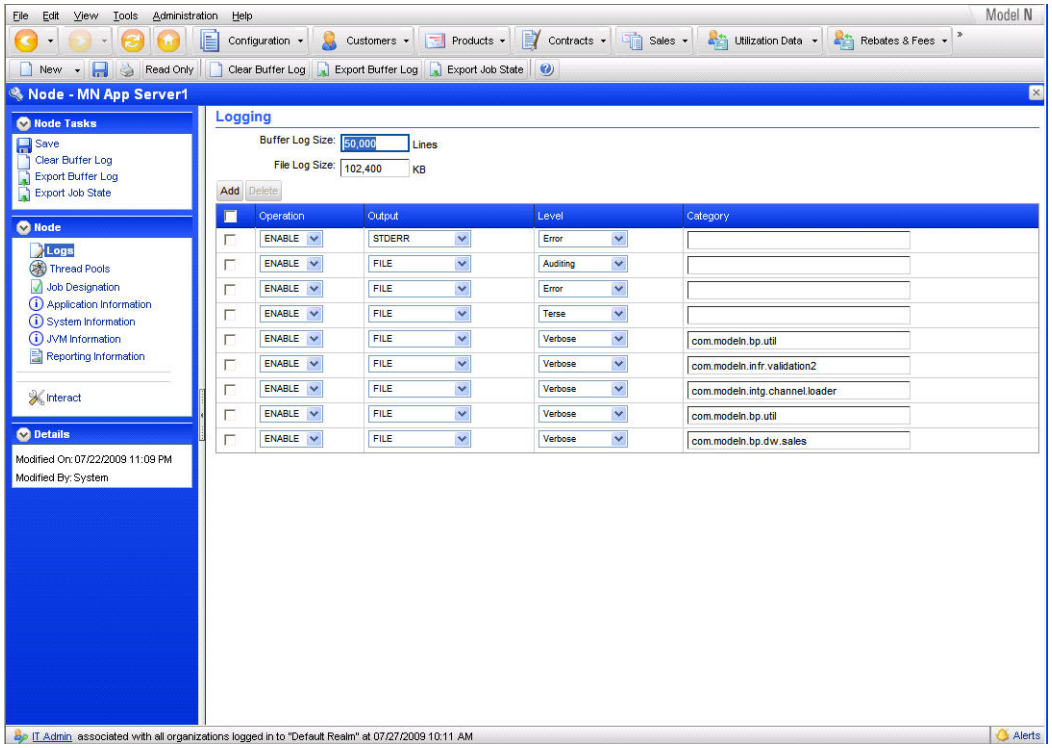
   The server starts capturing log output in the in-memory buffer.
4. Once the desired event has occurred in the application, click **Export Buffer Log**.

   The log output is returned to you as a .csv file which you can save on your machine and analyze.

4.5.1 Logging Page

Navigation Path: **Administration > Management > Nodes > Node Name** link

Figure 4-5: Logging Page



The Logging page lets you view and make changes to the log settings on a selected node.

Table 4-3: Logging Page

Name	Type	Description
Buffer Log Size	Text box	Lets you specify the number of lines present in the in-memory buffer in which the log is captured.
File Log Size	Text box	Used to identify the maximum size of the log file if any output setting is selected to be of type *File*. The logging sub-system creates new files to capture log output once the maximum file size had been reached.
Add	Button	Lets you add a log setting.
Delete	Button	Lets you delete a selected log setting.
{check box}	Check box	Enable the Delete button to delete the selected item.

Table 4-3: Logging Page (Continued)

Name	Type	Description
Operation	Drop-down list	Sets whether to enable or disable the log setting.
Output	Drop-down list	Sets the output type for the log.
Level	Drop-down list	Sets the logging level.
Category	Text box	The class name you want to log.

4.5.1.1 Recommended Logging Levels

Following are the recommended logging levels to use in production and for testing. The following logging levels send all of the logs to Standard Error. If you want the logging output to go somewhere else, you can use something other than `STDERR`. The following levels contain `ADD` lines which let you add to your existing settings. If you want to override your previous settings, use `SET` instead of `ADD`.

Note: Log levels set by different parts of the system interact, so the order in which they are applied is important. For example, Sales logs a component at the `WARNING` level but Chargebacks logs it at the `VERBOSE` level. If you apply the Sales setting on top of the Chargebacks setting, you get `WARNING`. If you apply the Chargebacks setting on top of the Sales setting, you get `VERBOSE`. Therefore, you should generally put the more verbose logging levels towards the end.

Table 4-4: Recommended Logging Levels

Purpose	Level	Category
Production	ERROR	com.modeln
Batch Validation Framework Testing	VERBOSE	com.modeln.infr.validation2
Bid Awards Testing	VERBOSE	com.modeln.bp.ba (to debug back-end logic)
Catalog Testing	VERBOSE	com.modeln.bp.catalog (to debug back-end logic)
Chargebacks Testing	VERBOSE	com.modeln.bp.distrebates (to debug back-end logic)
	VERBOSE	com.modeln.ac.distrebates (to debug the user interface)

Table 4-4: Recommended Logging Levels (Continued)

Purpose	Level	Category
Community Testing	VERBOSE	com.modeln.bp.cmty (to debug back-end logic)
	VERBOSE	com.modeln.infr.cmty
Direct Loader Testing	VERBOSE	com.modeln.intg.channel.loader
FSS Compliance Testing	VERBOSE	com.modeln.ac.fss (to debug the user interface)
	VERBOSE	com.modeln.bp.fss (to debug back-end logic)
	VERBOSE	com.modeln.bp.pmon (to debug back-end logic)
Government Pricing Testing	VERBOSE	com.modeln.ac.gp (to debug the user interface)
	VERBOSE	com.modeln.bp.gp (to debug back-end logic)
Mass Updates Testing	VERBOSE	com.modeln.ac.contract.massupdate (to debug the user interface)
	VERBOSE	com.modeln.bp.massupdate (to debug the back-end logic)
	To debug the batch validation framework used for mass updates validations, see Batch Validation Framework Testing.	
Membership Testing	VERBOSE	com.modeln.bp.cmty.membership (to debug the back-end logic)

Table 4-4: Recommended Logging Levels (Continued)

Purpose	Level	Category
Managed Care Testing	ERROR	com.modeln (the default to log only ERROR messages)
	VERBOSE	com.modeln.bp.mco.claim (for Managed Care claims back-end logic)
	VERBOSE	com.modeln.bp.mco.utilization (for Managed Care utilization back-end logic)
	VERBOSE	com.modeln.bp.mco.invoice (for Managed Care invoices back-end logic)
	VERBOSE	com.modeln.bp.mco.nms (for Managed Care National Market Share back-end logic)
	To debug the batch validation framework used for mass updates validations, see Batch Validation Framework Testing.	
Medicaid Testing	VERBOSE	com.modeln.ac.medicaid (to debug the user interface)
	VERBOSE	com.modeln.bp.medicaid (to debug the back-end logic)
Price Master Testing	VERBOSE	com.modeln.bp.pm (to debug the back-end logic)
Price Monitoring Testing	VERBOSE	com.modeln.bp.pmon (to debug the back-end logic)

Table 4-4: Recommended Logging Levels (Continued)

Purpose	Level	Category
Rebates Testing	ERROR	com.modeln (the default to log only ERROR messages)
	VERBOSE	com.modeln.infr.validation2 (to test the Validation Framework)
	VERBOSE	com.modeln.infr.validation (to test the Validation Framework used for market share and trace sales only)
	VERBOSE	com.modeln.bp.rebates (to debug the back-end logic)
	ERROR	com.modeln.bp.rebates (to debug the back-end logic)
	VERBOSE	com.modeln.bp.tracesales (to debug the back-end logic for Trace Sales)
	VERBOSE	com.modeln.bp.mktsales (to debug the back-end logic for Market Share Sales)
Sales and Chargebacks Testing	VERBOSE	com.modeln.infr.validation2
Sales Testing	VERBOSE	com.modeln.bp.dw.sales (to debug the back-end logic)
	VERBOSE	com.modeln.ac.sales (to debug the user interface)
	ERROR	com.modeln.ac.sales.line.CMnSalesLineDetailComp (to debug the user interface)
Platform Testing	LOCKING	com.modeln.infr.locking
	TIMING	
	INTG	

4.6 Job Designation

Job Designation lets you direct background jobs to specific nodes. You can identify jobs by identifying the filter associated with a job. You can choose to exclude a job from being executed on a specific node. By default, background jobs can be executed on any node in the cluster.

The job filter is a list of command types, separated by commas. For example, `root.command.intg.*,root.command.audit. root.command.intg.*` means all command types with a type that begins with `root.command.intg`. The command types are saved in `mn_resource` table with `rsrc_name` starts with "root.command.".

If you select Exclude as the filter mode, the list of jobs that have been identified to be excluded from being executed on the node is displayed. Similarly, if you select Include as the filter mode, the list of jobs that have been identified to appear in the Include list for a node is displayed.

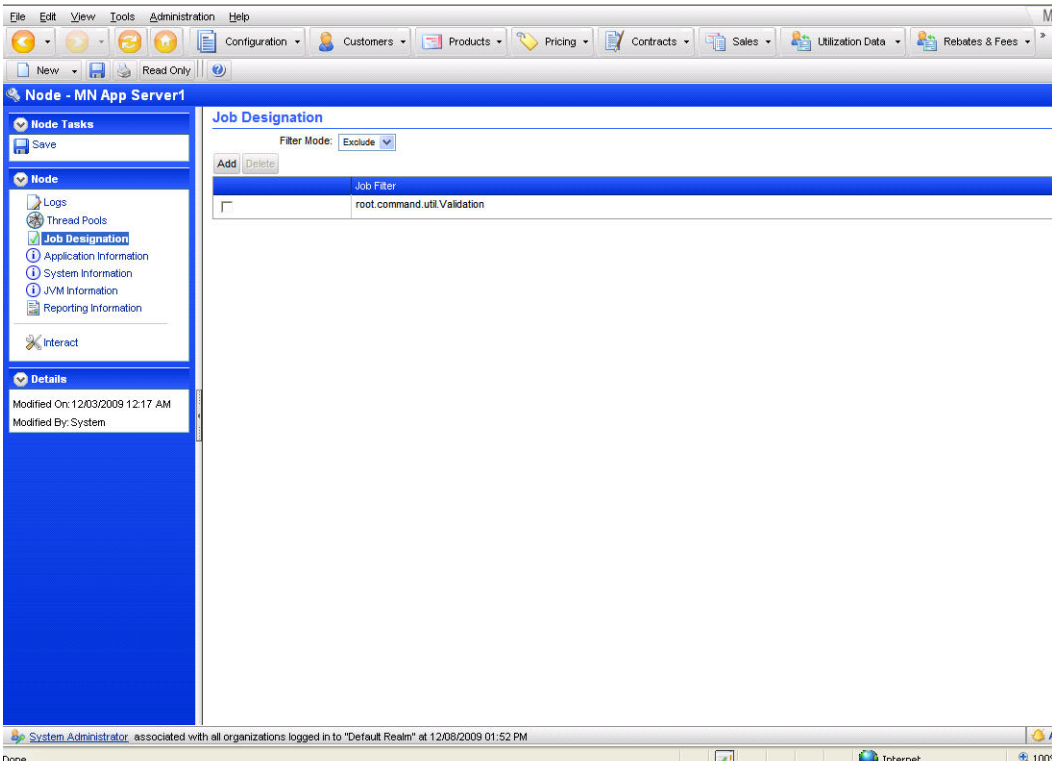
Note: The default value of the filter mode is Exclude. By default, there are no job types in the Job Filter list.

If you need to have a dedicated server node configured to run data flow jobs only, the filter mode should be set to INCLUDE, and filter list should be set to `root.command.intg.dataflow.*` . For the other nodes that can run everything else, the filter mode should set to EXCLUDE, and the filter list also set to `root.command.intg.dataflow.*`

4.6.1 Job Designation Page

Navigation Path: **Administration > Management > Nodes > Node Name** link **> Job Designation**

Figure 4-6: Job Designation Page



The Job Designation page lets you direct background jobs to specific nodes.

Table 4-5: Job Designation Page

Name	Type	Description
Filter Mode	Drop-down list	Lets you filter the jobs displayed by whether they are excluded from execution or not.
Add	Button	Lets you add job filters.
Delete	Button	Lets you delete job filters.
{check box}	Check box	Select the check box to enable the Delete button.
Job Filter	String	Lists the job filters found.

4.7 State Management

The Job Service in the Model N platform maintains in-memory state as the jobs are executed on each node in the application. To observe the internal state of in-memory data structures, you can execute the Export Job State task which is available on the node when the log section of node has been selected. If you execute this task, the internal state is delivered as an XML file to the browser. This information can be useful for debugging job behavior.

4.7.1 Thread Pool Management

Model N applications are configured with multiple thread pools. Background jobs executed as part of the application operation are executed by one of the threads in a thread pool. Each kind of job is tied to a specific thread pool. The number of threads in a thread pool defines the maximum number of concurrent jobs of a specific type that can be executed on a node in the cluster. It is possible for more than one job type to be associated with a thread pool.

For each thread pool, the application lets you define the minimum and maximum pool size. You should manage the number of threads in a thread pool with respect to the hardware on which a node is hosted. Any changes to the pool size are applied when the node is restarted. Knowing the average percentage of busy time helps you identify the usage of threads in the thread pool which can help you decide the size of the thread pool.

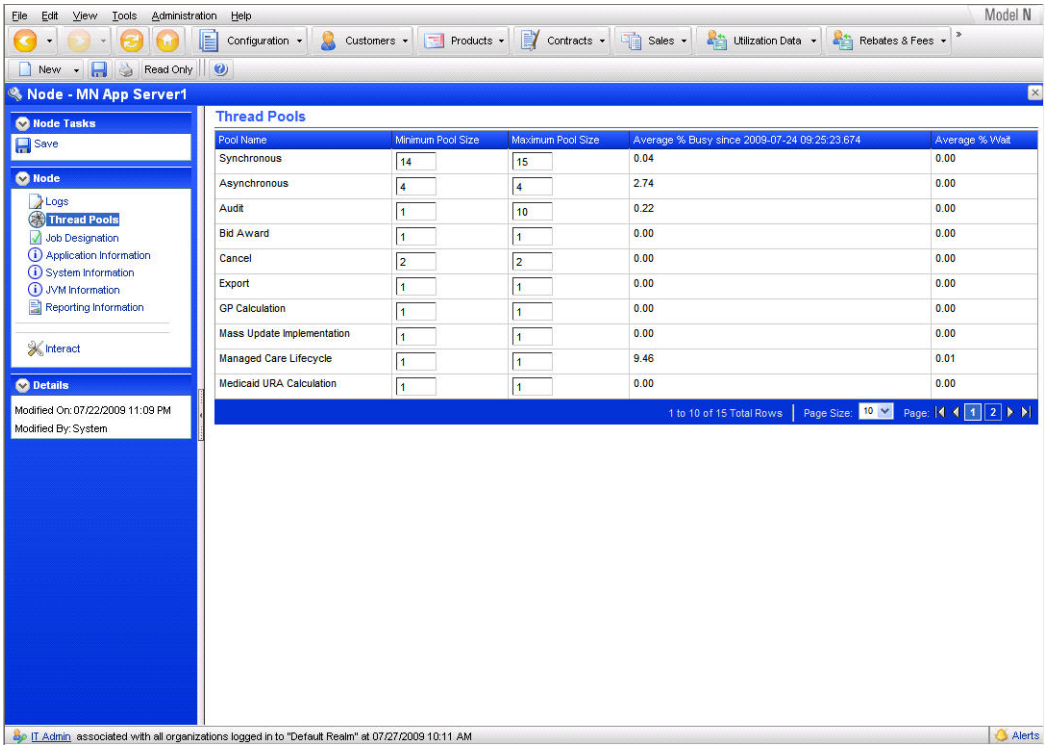
When you modify thread pool sizes, consider factors such as the number of processors, type of processor, or utilization of the processor.

For more information about Model N thread pools, see the Model N *Installation & Migration Guide*.

4.7.2 Thread Pools Page

Navigation Path: **Administration > Management > Nodes > Node Name** link **> Thread Pools**

Figure 4-7: Thread Pools Page



The Thread Pools page lets you define the minimum and maximum pool size.

Table 4-6: Thread Pools Page

Name	Type	Description
Pool Name	String	Name of the thread pool.
Minimum Pool Size	Text box	Minimum thread pool size.
Maximum Pool Size	Text box	Maximum thread pool size.
Average % Busy Since	String	Average percent use of the thread pool.
Average % Wait	String	Average percent wait of the thread pool.

4.8 System Information

The System Information includes the following set of pages that provide information about the application, node instance, JVM and reporting:

- Application Information Page

  This page lists the major and minor release versions.

- System Information Page

  This page shows information about the host IP address, the host name, the operating system name, the operating system architecture, and the number of processors.

- JVM Information Page

  This section contains information about the Java Virtual Machine (JVM) on which this node is running. It contains information about the JVM vendor, the JVM name, the JVM version, the JVM arguments, the JVM class path, minimum and maximum heap size, the JDBC driver name, and the JDBC driver version.

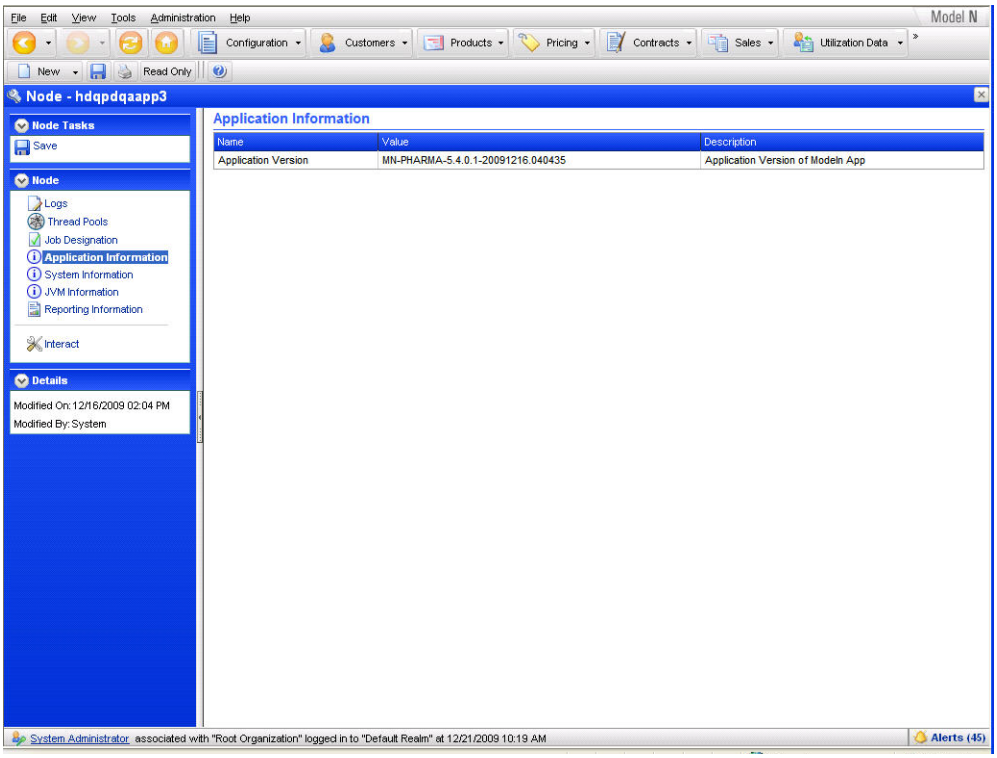
- Reporting Information Page

  This section contains information about the Cognos report integration and lists some configuration values such as Cognos enabled/disabled, dispatcher URL, connection timeout, request timeout, Cognos domain, Cognos path, root template path, wait time, and connection test period. It also contains a Cognos connection test button that synchronously tests the Cognos connection from that particular node.

4.8.1 Application Information Page

Navigation Path: **Administration > Management > Nodes > Node Name** link **> Application Information**

Figure 4-8: Application Information Page

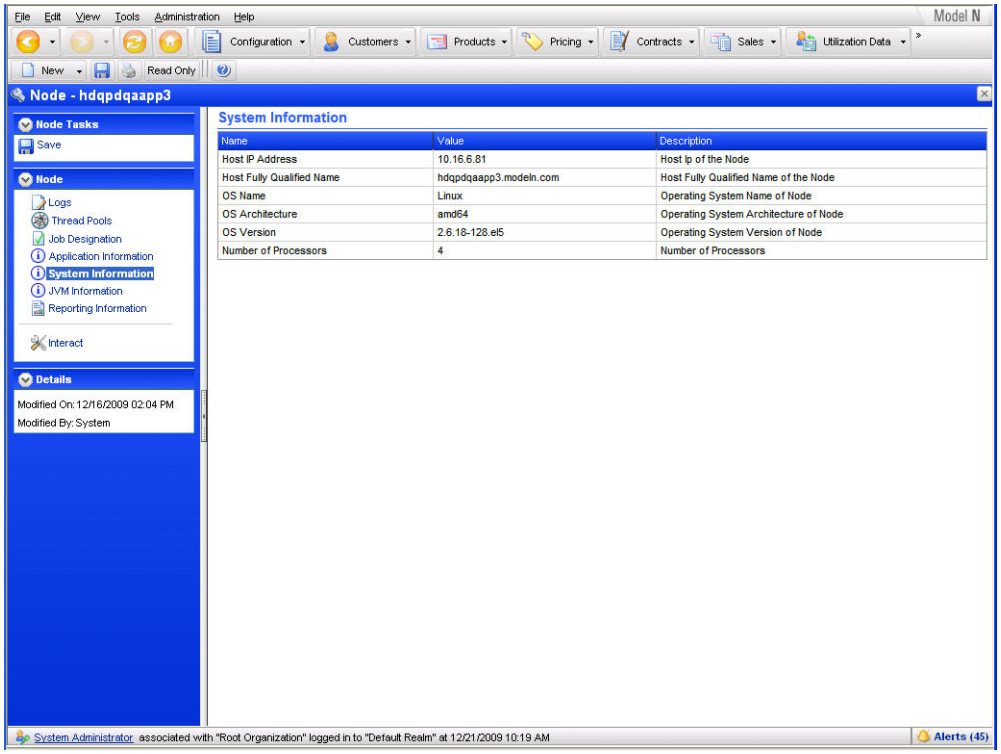


The Application Information Page provides information about Model N application build version.

4.8.2 System Information Page

Navigation Path: **Administration > Management > Nodes > Node Name** link **> System Information**

Figure 4-9: System Information Page



The System Information page provides information about the system hosting your Model N application.

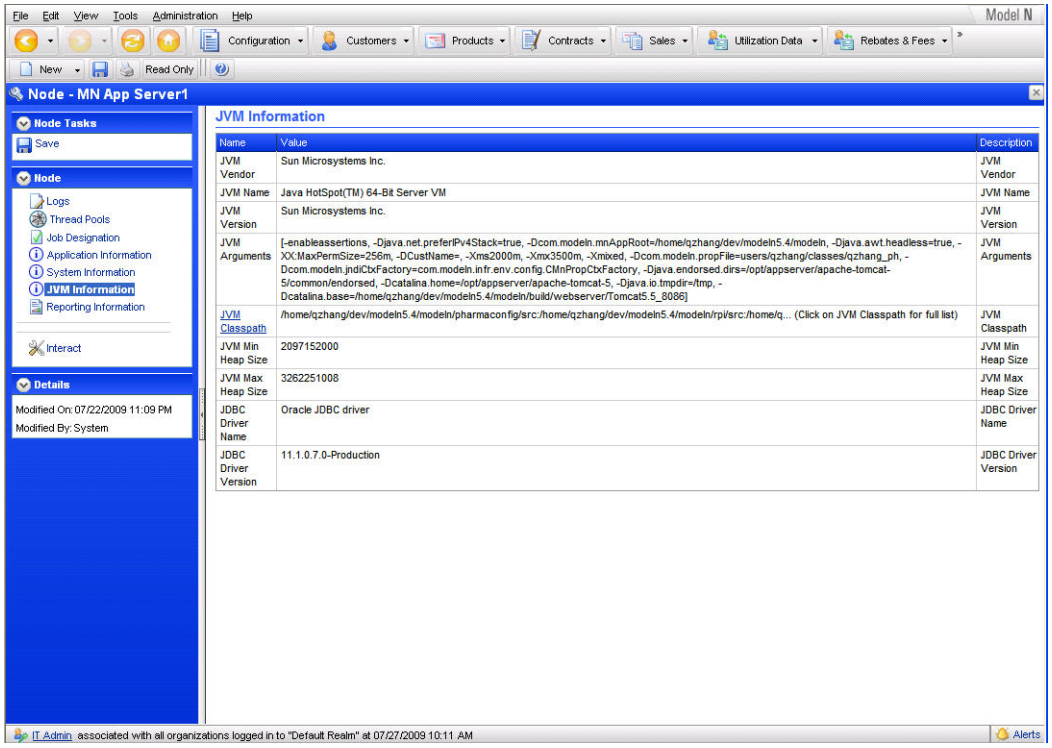
Table 4-7: System Information Page

Name	Type	Description
Name	String	The types of information presented.
Value	String	The values for the specified information types.
Description	String	A description of the information presented.

4.8.3 JVM Information Page

Navigation Path: **Administration > Management > Nodes > Node Name** link **> JVM Information**

Figure 4-10: JVM Information Page

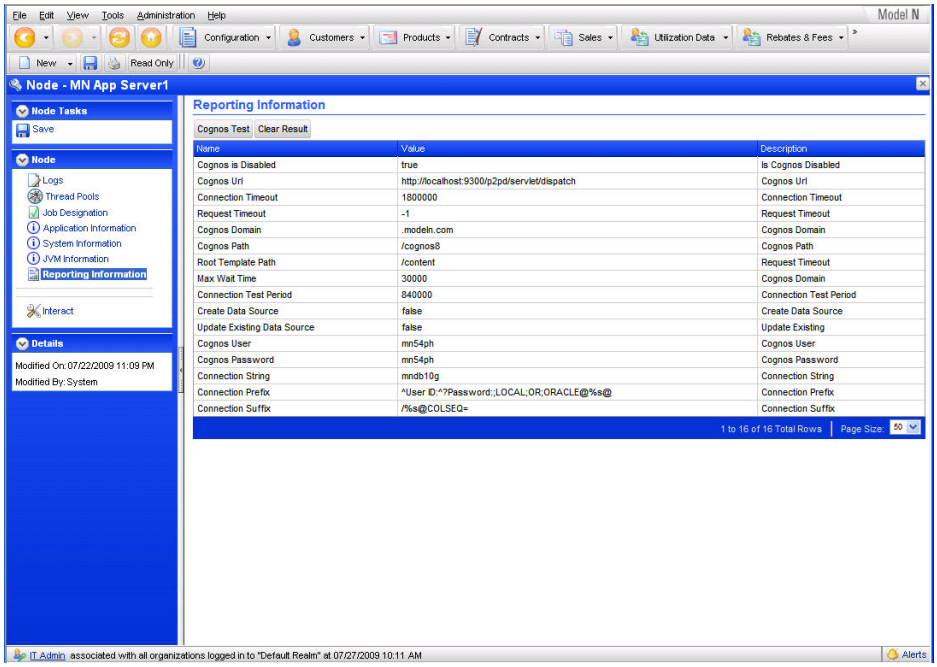


This page contains information about the Java Virtual Machine (JVM) on which this node is running. The JVM Classpath link opens the JVM Class Path dialog box, which contains the full list of JVM classpaths.

4.8.4 Reporting Information Page

Navigation Path: **Administration > Management > Nodes > Node Name** link **> Reporting Information**

Figure 4-11: Reporting Information Page



This page contains information about the Cognos report integration and its current status.

Table 4-8: Reporting Information Page

Name	Type	Description
Cognos Test	Button	Tests the connection to the Cognos reporting server.
Clear Result	Button	Clears the result of the Cognos Test.

4.9 Cluster-Based Management

Cluster-based management lets you manage tasks that are spread over the cluster. This section includes information on the following:

- Job Management

  Jobs in the Model N framework are used to execute background jobs. The sections that follow describe various management tools available to manage jobs.

- Lock Management

  Lock Management provides visibility into persistent locks being held by the application.

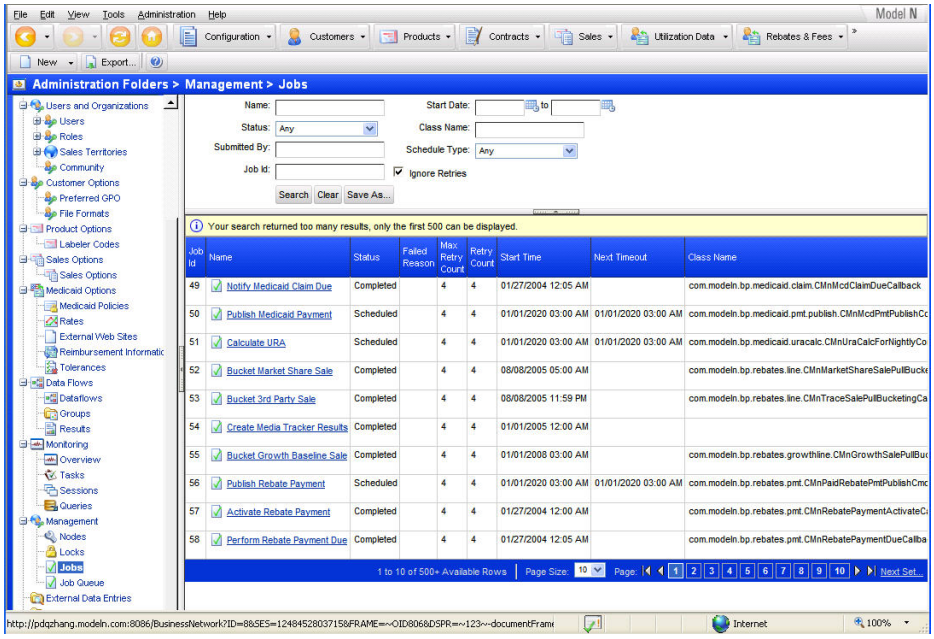
4.9.1 How to Search for Jobs

To locate a job based on various search fields, go to **Administration > Management > Jobs**.

4.9.2 Jobs Page

Navigation Path: **Administration > Management > Jobs**

Figure 4-12: Job Page



The Jobs page lets you view jobs in the Model N framework.

Table 4-9: Job Page

Name	Type	Description
Search		
Name	String	Name of the job being executed.

Table 4-9: Job Page (Continued)

Name	Type	Description
Status	Drop-down list	Helps to identify the status of the job. Possible values are: • **Cancelled:** jobs that had begun execution but were cancelled by the user. • **Completed:** jobs that have completed execution. • **Failed:** jobs that have failed execution upon termination of the job. • **On Hold:** jobs that are placed on hold using the Job Queue section of the Management Console. • **Pending Cancel:** jobs that are in the process of being cancelled. • **Recovered and Rescheduled:** jobs that have been automatically recovered and rescheduled for execution. • **Recovered and Terminated:** jobs that have automatically recovered and terminated. Some jobs are automatically terminated because the job is structured to be that way. • **Running:** jobs that are currently executing. • **Scheduled:** jobs that are scheduled to run in the future. • **Waiting:** jobs that should be executing as per their scheduling time, but are waiting because of lack of availability of threads in the cluster.
Submitted By	String	User who submitted the job.
Job Id	String	The ID number of the job.
Start Date	Date selector	Date range over which the job was or is scheduled to execute.
Class Name	String	Name of the class implementing the job.
Schedule Type	Drop-down list	Search by the various schedule types allowed in the application. Possible vales are: • **Single:** scheduled for a single run. • **Periodic:** scheduled to run after a periodic interval, specified in milliseconds. • **Daily:** scheduled to be run on a daily basis. • **Monthly:** scheduled to be run once a month.

Table 4-9: Job Page (Continued)

Name	Type	Description
Ignore Retries	Check box	Select this option to ignore jobs that are retries of other jobs.
Search	Button	Searches for jobs with the specified criteria.
Clear	Button	Clears your search criteria, letting you perform another search.
Save As	Button	Saves your search criteria so you many repeat the search.
Results		
Job Id	String	The ID number of the job.
Name	Link	Name of the job being executed.
Status	String	Status of the job.
Failed Reason	String	Reason the job failed.
Max Retry Count	String	Maximum retries the job can attempt.
Retry Count	String	Current retry count for the job.
Start Time	String	When the job was last started.
Next Timeout	String	When the next timeout is set to occur.
Class Name	String	Class name for the job.
Schedule Type	String	The schedule type for the job.
Schedule Interval	Integer	The interval of when the job can be scheduled.
Submitted By	String	User that submitted the job
Node Name	String	Name of the server node from which the job applies.

4.9.3 Job Detail

The Job General page provides details about the configuration associated with a job as well as details about previous runs associated with that job.

The General page provides general details on the job such as a short description of the job, its current state, and its average running time as observed by the application. It also provides details on the schedule setup for the job which includes details about the next start time of the job, the time when the schedule ends, the time when the job is scheduled to start, and its recurrence frequency. You can configure all jobs to be re-tried in case the job fails to execute using the following two parameters:

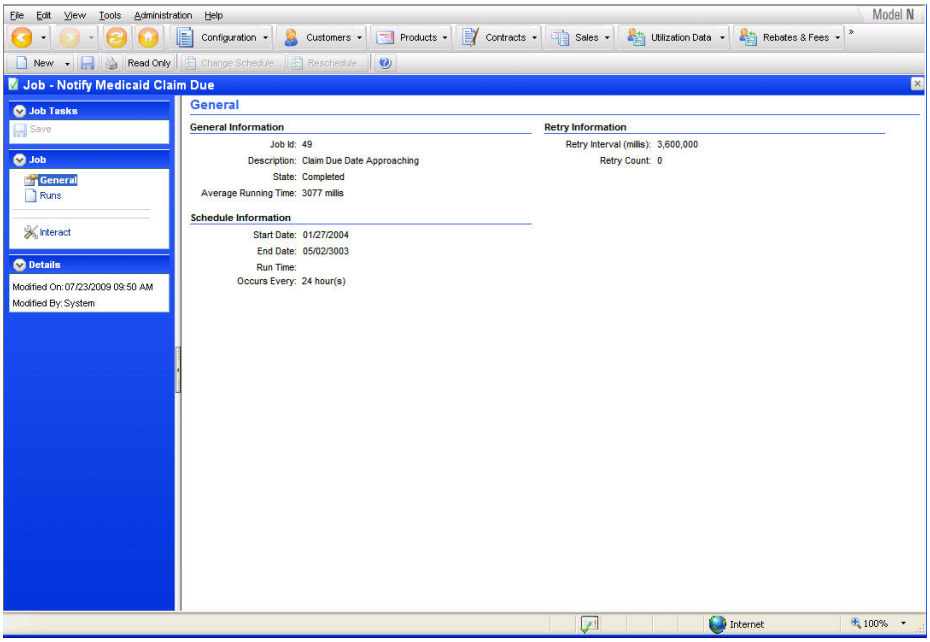
- the re-try interval after which the application tries to execute the job again
- the number of re-try attempts associated with the job

Any changes to the re-try configuration are applicable on the next run associated with the job.

4.9.4 Job General Page

Navigation Path: **Administration > Management > Jobs > Name** link

Figure 4-13: Job General Page



The Job General page provides general details on the job such as a short description of the job, its current state, and its average running time as observed by the application. It also provides details on the schedule setup for the job which includes details about the next start time of the job, the time when the schedule ends, the time when the job is scheduled to start, and its recurrence frequency.

4.9.5 Job Auditing

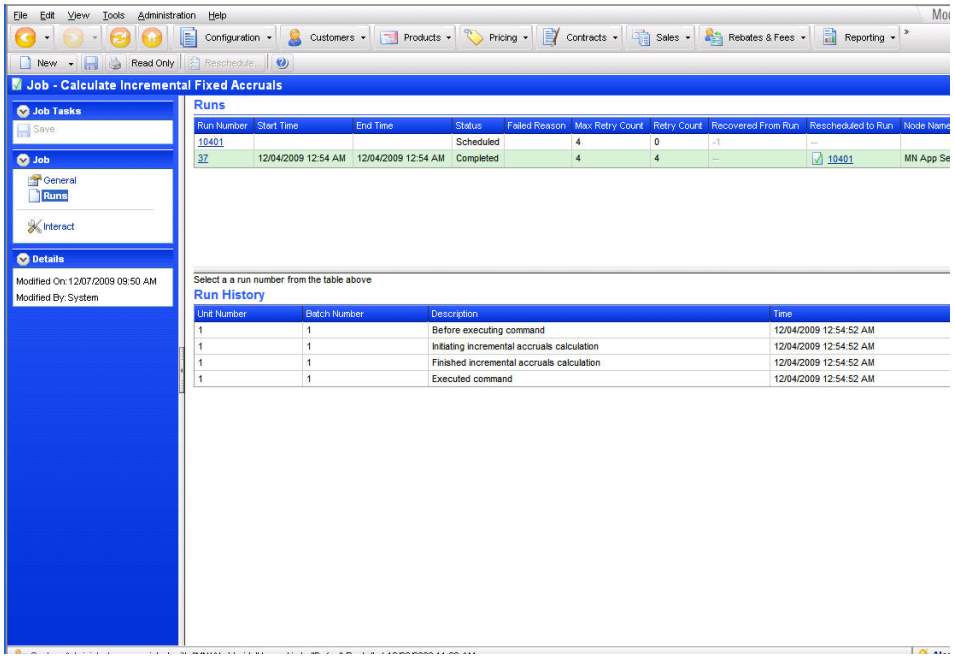
The **Runs** link on a job detail provides details of historical runs associated with the job. The upper table on the right pane shows this list. Every job has a run number to identify it uniquely in the application. Click on the run number to display the run log associated with the job. For each run, the list of historical runs also shows the time when the job was started and the time when the job completed execution.

In case of system failure, the job may be forced to terminate execution. In such cases, the application recovers the job automatically. The job might or might not re-execute on the node on which it was running in its prior run. In some cases, the job re-starts execution from the start. In cases when the job is processing a large amount of data, the execution could re-start from the last recoverable point known to the job in which case it divides its execution into parts and commits the changes for each separately before moving on to process more data. The **Recovered from Run** column in the list of runs identifies the job from which this job might have recovered. If the job did not recover from any prior run, the field is left blank.

4.9.6 Job Runs Page

Navigation Path: **Administration > Management > Jobs > Name** link **> Runs**

Figure 4-14: Job Runs Page



The Runs page provides details of historical runs associated with the job.

Table 4-10: Job Runs Page

Name	Type	Description
Runs		

Table 4-10: Job Runs Page (Continued)

Name	Type	Description
Run Number	Link	Unique system identifier for the run.
Start Time	String	When the job was started.
End Time	String	When the job completed.
Status	String	Status of the job run.
Failed Reason	String	Reason the job failed, if applicable.
Max Retry Count	String	Maximum retries the job can attempt.
Retry Count	String	Current retry count for the job.
Recovered From Run	String	Identifies the job from which this job might have recovered. If the job did not recover from any prior run, the field is left blank.
Rescheduled To Run	Link	Run number for a job rescheduled to run.
Node Name	String	Name of the node on which the job is to run.
Run History		
Unit Number	String	
Batch Number	String	
Description	String	
Time	String	

4.9.7 Job Queue

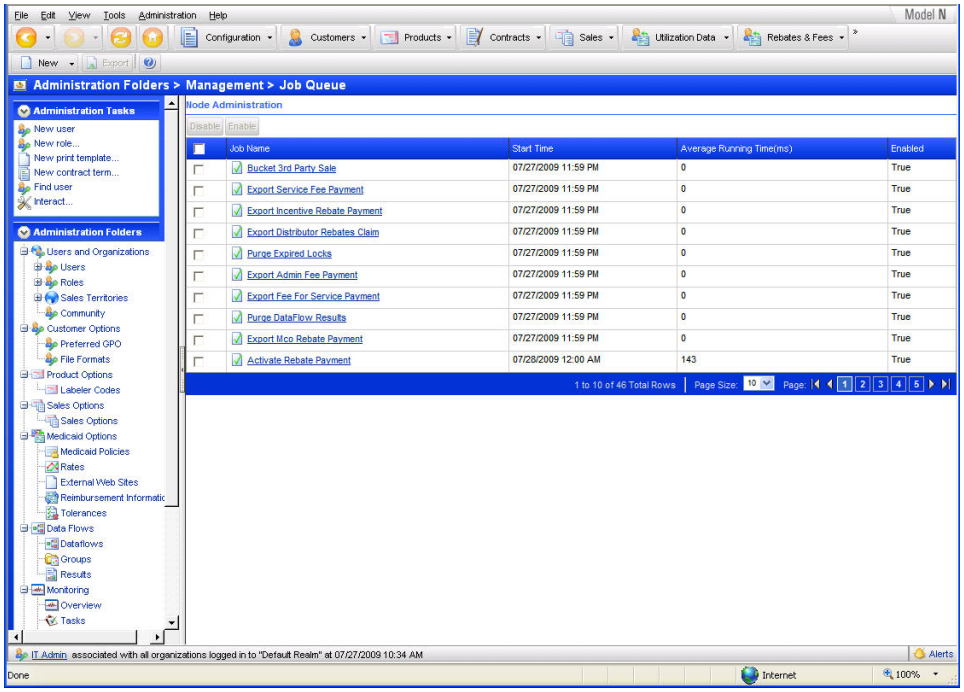
The **Job Queue** link in the left pane of the Management Console shows the active job queue present in the cluster. The jobs listed in the right pane are ordered by scheduled time with the earliest job listed first.

The Job Queue also lets you manage the scheduling of the jobs. The Enable and Disable buttons available on top of the job queue list let you to select one or more jobs and either enable or disable them. You may want to disable a job that has on average taken a long time and has been spawned by the application. Once you have disabled a job, you can click the job and navigate to the detail view of the job to schedule it for a time when dedicating computing resources are available within the cluster. Later, you can come back to the job queue and enable the job, thereby making it eligible to be executed at the specified time.

4.9.8 Jobs Queue Page

Navigation Path: **Administration > Management > Jobs Queue**

Figure 4-15: Jobs Queue Page



The Jobs Queue page shows the active job queue present in the cluster.

Table 4-11: Jobs Queue Page

Name	Type	Description
[check box]	Check box	Lets you select one or more jobs , then enables the **Enable** and **Disable** buttons.
Job Name	Link	Name of the scheduled job.
Start Time	String	Time when the job is scheduled to run next.
Average Running Time (ms)	String	Represents the average time taken by this job in the past runs.
Enabled	String	Indicates whether a particular job is currently enabled or not.
Disable	Button	Lets you disable the selected job.
Enable	Button	Lets you enable the selected job.

4.9.9 Lock Management

The Lock Management section (click the Locks link in the left pane) of the Management Console provides visibility into persistent locks being held by the application.

You can search for locks by identifying either the Object Type or the Lock Owner. The search results are grouped by the following object types.

Table 4-12: Search Objects Types

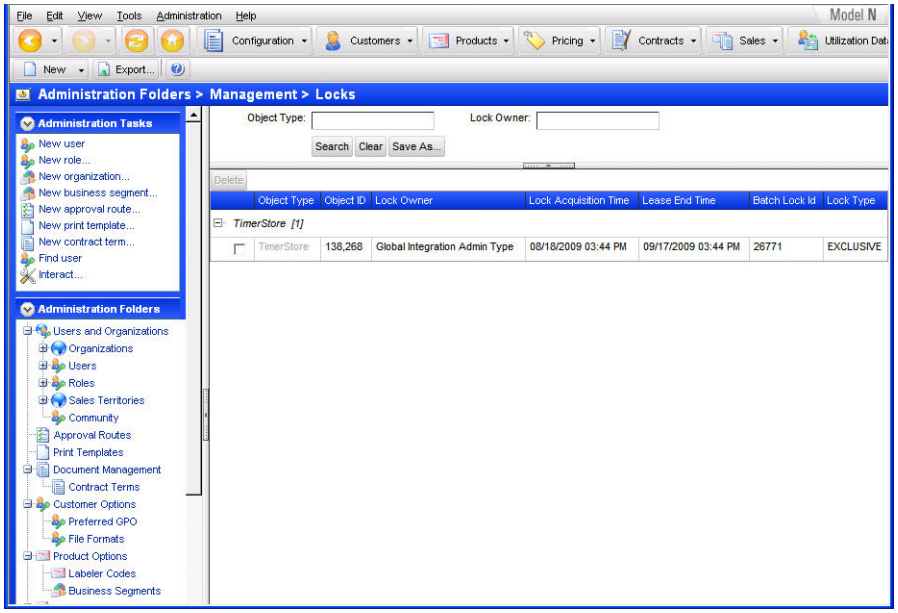
Object Type	This identifies the identity name of the locked object.
Object ID	This identifies the identifying information of the locked object.
Lock Owner	This identifies the owner of the lock.
Lock Acquisition Time	This identifies the time when the lock was acquired on the object.
Lease End Time	Locks are valid for pre-allocated time intervals known as lease time. The Lease End Time identifies the time when the object is set to lose the lease on the lock.
Batch Lock ID	In some cases the application might obtain locks for a collection of objects as a batch operation. The operation to obtain a batch lock on a collection of objects associates a Batch Lock ID with all of the locks involved. This field identifies the Batch Lock ID associated with the lock.
Lock Type	Locks assigned to objects can either be shared or exclusive. Objects in the Model N application can belong to locking hierarchies. Obtaining a lock on an object results in an exclusive lock being granted to the object being locked and a shared lock being granted with the locking operation to all of the parent objects in the hierarchy of the object being locked. The Lock Type identifies the lock as either a shared or an exclusive lock.

Locks are closely tied to the implementation in progress and are held to provide exclusive operation rights on objects. Deleting locks is not a common administrative operation and caution should be exercised when deleting locks.

4.9.10 Locks Page

Navigation Path: **Administration > Management > Locks**

Figure 4-16: Locks Page



Lock Management provides visibility into persistent locks being held by the application.

Table 4-13: Locks Page

Name	Type	Description
Search		
Object Type	Text box	Identity name of the locked object.
Lock Owner	Text box	Owner of the lock.
Search	Button	Searches for locks with the specified criteria.
Clear	Button	Clears your search criteria, letting you perform another search.
Save As	Button	Saves your search criteria so you many repeat the search.
Results		
[check box]	Check box	Lets you select a persistent lock and enables the Delete button.

Table 4-13: Locks Page (Continued)

Name	Type	Description
Object Type	String	Identity name of the locked object.
Object ID	String	Identifies the identifying information of the locked object.
Lock Owner	String	Owner of the lock.
Lock Acquisition Time	String	Time when the lock was acquired on the object.
Lease End Time	String	Time when the object is set to lose the lease on the lock. Locks are valid for pre-allocated time intervals known as lease time.
Batch Lock ID	String	Batch Lock ID associated with the lock. In some cases the application might obtain locks for a collection of objects as a batch operation. The operation to obtain a batch lock on a collection of objects associates a Batch Lock ID with all of the locks involved.
Lock Type	String	Identifies the lock as either a shared or an exclusive lock. Locks assigned to objects can either be shared or exclusive. Objects in the Model N application can belong to locking hierarchies. Obtaining a lock on an object results in an exclusive lock being granted to the object being locked and a shared lock being granted with the locking operation to all of the parent objects in the hierarchy of the object being locked.
Delete	Button	Lets you delete the selected locks.

4.10 Oracle Statistics

This section covers information on how to gather statistics used to monitor the Oracle database.

You should gather table and index statistics regularly (at least daily) in order to provide the Oracle optimizer with current statistics for query optimization. You can automate this task through the `DBMS_JOB` and `DBMS_STATS` Oracle packages.

4.10.1 Oracle STATSPACK

If you suspect performance bottlenecks, use Oracle STATSPACK to monitor the instance efficiency or to identify problem queries. STATSPACK produces comprehensive reports on database statistics. You can define the granularity of a STATSPACK report by specifying a level value in the `i_snap_level` parameter. The following table lists these parameters and provides information about them.

Parameter Name	Range of Valid Values	Default Value	Meaning
`i_snap_level`	0, 5, 6, 7, 10	5	Snapshot Level
`i_ucomment`	Text	Blank	Comment to be stored with Snapshot
`i_executions`	Integer >= 0	100	SQL Threshold: number of times the statement was executed
`i_disk_reads_th`	Integer >= 0	1000	SQL Threshold: number of disk reads the statement made
`i_parse_calls_th`	Integer >= 0	10000	SQL Threshold: number of parse calls the statement made
`i_buffer_gets_th`	Integer >= 0	10000	SQL Threshold: number of buffer gets the statement made
`i_sharable_mem_th`	Integer >= 0	1048576	SQL Threshold: amount of sharable memory
`i_version_count_th`	Integer >= 0	20	SQL Threshold: number of versions of a SQL statement
`i_session_id`	Valid SID from v$session	0 (no session)	Session Id of the Oracle Session for which to capture session granular statistics
`i_modify_parameter`	True, False	False	Save the parameters specified for future snapshots?

4.10.2 Snapshot Levels

To change the amount of information gathered by the package, specify a different snapshot level. The level that you choose (or the default level) determines the amount of data collected.

Following are descriptions of different snapshot levels.

Level = 0

General performance statistics

This level and any level greater than 0 collects general performance statistics, such as:

- wait statistics
- system events
- system statistics
- rollback segment data
- row cache
- SGA
- background events
- session events
- lock statistics
- buffer pool statistics
- parent latch statistics

Level = 5

Additional data: SQL Statements

This level includes all statistics gathered in the lower levels, and additionally gathers the performance data on high resource usage SQL statements.

SQL 'Thresholds'

The SQL statements gathered by Oracle STATSPACK are those that exceed one of the following predefined threshold parameters:

- number of executions of the SQL statement (default 100)
- number of disk reads performed by the SQL statement (default 1,000)
- number of parse calls performed by the SQL statement (default 1,000)
- number of buffer gets performed by the SQL statement (default 10,000)

The values of each of these threshold parameters are used when to determine which SQL statements to collect. If an SQL statement's resource usage exceeds any of the listed threshold values, it is captured during the snapshot.

The SQL threshold levels used are either those stored in the table `stats$statspack_parameter` or by the thresholds specified when the snapshot is taken.

Level = 6

This level includes all statistics gathered in the lower level(s). Additionally, it gathers SQL plans and plan usage data for each of the high resource usage SQL statements captured. Therefore, you should use level 6 snapshots whenever there is the possibility that a plan may change. To gather the plan for an SQL statement, the statement must be in the shared pool at the time the snapshot is taken, and it must exceed one of the SQL thresholds. To gather plans for all statements in the shared pool, specify the executions threshold to be zero (0) for those snapshots.

Level = 7

Additional statistics: Segment Reads

This level was added in Oracle 9.2 and by using the new segment statistics data from view `V$SEGMENT_STATISTICS` shows which segments (tables or indexes) have the most physical reads being performed against them. The details of such segments provide information about whether indexes should be rebuilt or partitioning could be used to reduce I/O on those segments. Oracle STATSPACK also generates a Segment Statistics report starting at level 7.

Level = 10

Additional statistics: Child latches

This level includes all statistics gathered in the lower levels and additionally gathers high Child Latch information. Data gathered at this level can sometimes cause the snapshot to take longer to complete. For example, this level can be resource intensive and should only be used when advised by Oracle personnel.

4.10.3 Rebuilding Indexes

Rebuild or coalesce indexes regularly (every night) with the `ALTER INDEX index_name REBUILD ONLINE` and the `ALTER INDEX index_name COALESCE` commands, respectively.

The `COALESCE` option is a less resource-intensive technique to regain free space in an index without having to rebuild it completely. For a rebuild, the index has to be scanned, the results sorted, and new extents built to hold the newly constructed index. These extents must co-exist with the original index until the process is complete and the original can be dropped. Thus, the rebuild requires a sort and sufficient free space to hold two versions of the index temporarily.

4.10.4 Monitoring Index Usage

Oracle provides a means of monitoring indexes to determine whether they are being used. If an index is not being used, you can drop it to eliminate unnecessary DML statement overhead.

To start monitoring the usage of an index, issue the following statement in SQL*Plus:

```
SQL> ALTER INDEX index MONITORING USAGE;
```

Once the Model N application has been in use for some time or, after you have completed a full test cycle of the application, issue the following statement to stop index monitoring:

```
SQL> ALTER INDEX index NOMONITORING USAGE;
```

You can query the Oracle view V$OBJECT_USAGE for the index being monitored to see if the index has been used. The view contains a USED column whose value is either YES or NO, depending on whether the index has been used within the time period being monitored. The view also contains the start and stop times of the monitoring period and a MONITORING column (YES/NO) to indicate if usage monitoring is currently active.

Each time you specify the MONITORING USAGE clause, the V$OBJECT_USAGE view is reset for the specified index. The previous usage information is cleared and a new start time is recorded. When you specify the NOMONITORING USAGE clause, no further monitoring is performed and the end time is recorded for the monitoring period. Until you issue the next ALTER INDEX...MONITORING USAGE statement, the view information is left unchanged.

4.10.5 Monitoring the Database Server

4.10.5.1 CPU Utilization Trend

The CPU utilization for the database server should be measured over time. The utilization is the load on the CPU. The common measurements are either the CPU load or the % utilization.

In most production systems, we usually see the CPU utilization around 50% for most periods, but can climb to higher values resulting from background activities, such as rebate bucketing, price monitoring, membership publishing, GP calculations, or Medicaid URA calculations.

Tools

Most monitoring tools provide some CPU monitoring capability. At Model N, we use an open source tool called internally to monitor our servers. For more information, see http://www.zabbix.com.

4.10.5.2 Slowest Queries

The response time for queries provides a good indicator on the overall application performance, since the Model N system relies heavily on the database. Queries can slow down for a variety of reasons. The most common reason is data growth.

The following query lists the slowest queries by average response time. The full SQL string of the query, average response time, and number of executions is provided. The task type indicates whether the query is associated with a UI request (UI) or background activity (BE). A threshold of 10 seconds is given for UI queries and 60 seconds for background activity queries.

Code 4-5: Query for Slowest Queries

```
SELECT
  usage.thread_type AS thread_type,
  usage.avg_time    AS avg_time,
  usage.total_time  AS total_time,
  usage.query_count AS query_count,
  usage.task_count  AS task_count,
  query.sql         AS sql_string
FROM
  (SELECT
```

Code 4-5: Query for Slowest Queries (Continued)

```
    ROUND(SUM(u.time) / 1000, 2)                AS total_time,
    SUM(u.uses)                                 AS query_count,
    ROUND(SUM(u.time) / (1000*SUM(u.uses)), 2)  AS avg_time,
    COUNT(*)                                    AS task_count,
    u.query_handle_id                           AS query_handle_id,
    t.thread_type                               AS thread_type
  FROM
    mn_hist_query_usage u
    INNER JOIN mn_hist_task t ON (
      u.task_id = t.task_id
    )
  WHERE
   t.task_start_date BETWEEN SYSTIMESTAMP - 7 AND SYSTIMESTAMP
  GROUP BY
    u.query_handle_id,
    t.thread_type
 ) usage
 INNER JOIN mn_hist_query_handle handle ON (
   usage.query_handle_id = handle.handle_id
 )
 INNER JOIN mn_hist_query query ON (
   query.query_id = handle.query_id
 )
WHERE
 (usage.thread_type = 'UserRequest'
    AND usage.avg_time > 10)
 OR
 (usage.thread_type IN ('TimerService','Command')
    AND usage.avg_time > 60)
ORDER BY
 usage.thread_type ASC,
 usage.avg_time DESC
;
```

4.10.5.3 Top Tables by Size

There are hundreds of tables in the Model N system with a variety of growth patterns. In the past, we have seen unusual growth patterns in customer deployments based on differing usages of the product as well as from customizations separate from the out of the box product. This may also provide information whether some areas may need to adjust the purging frequency (such as timers).

The following query takes a snapshot of all the tables within the Model N schema and includes sizing information (such as average row length) and statistics information (such as when the table was last analyzed). The result set is sorted such that the largest table (defined by the number of rows) is returned first.

Code 4-6: Query for Top Tables by Size

```
SELECT
  table_name,
```

Code 4-6: Query for Top Tables by Size (Continued)

```
  tablespace_name,
  temporary,
  num_rows,
  blocks,
  avg_row_len,
  sample_size,
  last_analyzed
FROM
  user_tables
ORDER BY
   num_rows DESC NULLS LAST
;
```

A similar query can be used to rank the tables by storage size:

Code 4-7: Query for Tables by Storage Size

```
SELECT
  table_name,
  tablespace_name,
  temporary,
  num_rows,
  blocks,
  avg_row_len,
  sample_size,
  last_analyzed
FROM
  user_tables
ORDER BY
   blocks DESC NULLS LAST
;
```

These queries assume that they are being run within the Model N user (that is, you logged into the database as the same user as the application would use).

These queries should be run periodically and tracked so that a trend can a growth trend can be determined.

4.10.5.4 Top Indexes by # Rows

As with the Top Tables by # Rows, the indexes should also be monitored for growth.

The following query takes a snapshot of all the indexes within the Model N schema and includes structural information (such as B-Tree Level, # Distinct Keys, Clustering Factor) and statistics information (such as when the index was last analyzed). The result set is sorted such that the largest index (defined by the number of rows) is returned first.

Code 4-8: Query for Top Indexes by # Rows

```
SELECT
   index_name,
   table_name,
```

Code 4-8: Query for Top Indexes by # Rows (Continued)

```
    tablespace_name,
    blevel,
    distinct_keys,
    clustering_factor,
    leaf_blocks,
    avg_data_blocks_per_key,
    avg_leaf_blocks_per_key,
    num_rows,
    sample_size,
    last_analyzed
FROM
    user_indexes
ORDER BY
    num_rows DESC NULLS LAST
;
```

A similar query can be used to rank the indexes by storage size as determined by the number of leaf blocks that are used:

Code 4-9: Query for Top Indexes by Storage Size

```
SELECT
    index_name,
    table_name,
    tablespace_name,
    blevel,
    distinct_keys,
    clustering_factor,
    leaf_blocks,
    avg_data_blocks_per_key,
    avg_leaf_blocks_per_key,
    num_rows,
    sample_size,
    last_analyzed
FROM
    user_indexes
ORDER BY
    leaf_blocks DESC NULLS LAST
;
```

These queries assume that they are being run within the Model N user.

Note that if the `blevel` for any index grows beyond a value of three, this may indicate that the index is becoming imbalanced and should be rebuilt.

4.10.6 Index Monitoring

There are many indexes within the Model N schema that are used for a variety of purposes. The way the Model N system is used may vary from deployment to deployment. Therefore, some indexes may become more or less important based on the specific usage. We recommend monitoring the indexes on a weekly basis.

Note that Oracle always recommends that foreign keys be indexed except in the case when the matching unique or primary key is never updated or deleted.

4.10.6.1 Oracle 10g

In Oracle 10g, the Automatic Workload Repository (AWR) mechanism provides statistics that include index usage. By default, the AWR collects statistics hourly and preserved for one week. As such, there is no other setup that is required to extract index statistics.

The following query returns the usage of indexes in a given schema:

Code 4-10: Query for Usage of Indexes in a Given Schema

```
SELECT
  i.table_name  AS table_name,
p.object_name AS index_name,
p.operation   AS operation,
p.options     AS options,
COUNT(1)      AS usage_count
FROM
dba_hist_sql_plan p
INNER JOIN dba_hist_sqlstat s ON (
  p.sql_id = s.sql_id
)
INNER JOIN dba_indexes I ON (
  p.object_owner = i.owner
  AND p.object_name = i.index_name
)
WHERE
p.object_owner = '<SCHEMA>'
AND p.operation LIKE '%INDEX%'
GROUP BY
  i.table_name,
p.object_name,
p.operation,
p.options
ORDER BY 1,2,3,4
;
```

Replace `<SCHEMA>` with the schema owner that contains the indexes to monitor. The options column provides information on how the index was used (such as `RANGE SCAN`, `UNIQUE SCAN`). The `usage_count` column indicates how many times a particular index was used. If an index is used multiple times within a single query, then this count would include the number of times it was referenced.

The following query returns the indexes that were not used in the given monitoring period:

Code 4-11: Query for Indexes Not Used in the Given Monitoring Period

```
SELECT
  i.table_name,
  i.index_name
FROM
  dba_indexes i
WHERE
  i.owner = '<SCHEMA>'
  AND NOT EXISTS (
    SELECT
      1
    FROM
      dba_hist_sql_plan p
      INNER JOIN dba_hist_sqlstat s ON (p.sql_id = s.sql_id)
    WHERE
      p.object_owner = i.owner
      AND p.object_name = i.index_name
      AND p.operation LIKE '%INDEX%'
  )
ORDER BY 1,2
;
```

4.11 Model N Application Status

This section describes how to log on to the Model N system as well as how to monitor the status of the application including determining whether the application is functioning normally, the total number of active user sessions, the total number of HTTP requests that the application has received since it was first started, the amount of used memory, and the current size of the JVM heap.

4.11.1 Logging On to the Application

Once the application is installed and running, you can access it using the appropriate URL. The specific URL you use depends on the application server and web server configuration. If you have configured the web server as described in the "Configuring the HTTP" section of the *Installation & Migration Guide*, the application URL generally looks as follows:

```
http://<HttpServerHost>/<ContextRoot>/BusinessNetwork
```

The host and port information may be different depending on the server configuration settings. The web server is responsible for accepting any incoming connections. Any requests containing `<ContextRoot>` are forwarded to the application server by the proxy. Any static content referenced by the application (such as HTML or image files) will not contain `<ContextRoot>` in the name and therefore is served directly by the web server.

4.11.2 Determining Application Status

The Model N application "heartbeat" is an HTTP response that indicates the status of the Model N application. The heartbeat response returns a small, fast response with minimal application overhead. Accessing the application heartbeat response does not generate persistent objects or sessions within the application.

4.11.2.1 Initiating the Application Status Response

You can initiate the application heartbeat response by submitting a request to the Model N application as follows:

```
http://<HttpServerHost>/<ContextRoot>/BusinessNetwork?heartbeat
```

If the application is available and functioning normally, the following HTTP response is returned:

```
Model N Application is alive...
```

4.11.2.2 Retrieving Diagnostic Information

You can also use the heartbeat response to retrieve diagnostic information about the application by submitting the following request:

Code 4-12: Request for Diagnostic Information

```
http://<HttpServerHost>/<ContextRoot>/BusinessNetwork?
  heartbeat=true&diagnostics=true
```

For security purposes, you can configure the application to ignore the diagnostic request See Configuring the Heartbeat. When diagnostic information is disabled, only the standard heartbeat status response is returned.

The diagnostic information displayed within the browser is refreshed at regular intervals. The default refresh interval is typically configured to be 30 seconds. You can specify a different refresh interval by using the `refresh` parameter:

Code 4-13: Refreshing Diagnostic Information

```
http://<HttpServerHost>/<ContextRoot>/BusinessNetwork?
  heartbeat=true&diagnostics=true&refresh=30
```

The refresh interval is specified in *seconds* as in the following example of a heartbeat diagnostic response.

Code 4-14: Heartbeat Diagnostic Response

```
Model N Heartbeat Information

Product Version: MN-REVEX-4.6-20040227.000200
Start Date: Fri Feb 27 10:55:24 PST 2004
Up Time: 5.49 hr Active Sessions: 1 (1)

Total Request Count: 84 (2)

Last Request Time: Fri Feb 27 16:19:23 PST 2004 Used Memory:
420.00 M (3)

Total Memory: 680.00 M (4)
```

Code Section Notes:

(1) The number of active sessions indicates the total number of user sessions that are currently active on the server. Although this number generally indicates the total number of users that are currently logged into the application, it also includes inactive sessions that were not terminated properly and have not yet timed out. These inactive sessions are typically created when users close their browsers rather than log out through the application.

(2) The total request count indicates the total number of HTTP requests that have been received by the application server since the date when the application was first started. The total number of requests will continue to grow larger with time, but does not indicate that resources are being consumed. *A large request count does not indicate a problem, but a rapidly growing request count may indicate abnormally high traffic.*

(3) The amount of used memory indicates the total amount of physical memory utilized by the application. This number is expected to grow in relation to application usage and the number of active sessions. The amount of used memory shrinks as the JVM performs garbage collections, users log out, or as inactive sessions expire.

(4) The total memory indicates the current size of the JVM heap. This is the amount of memory that the application has available to use. As the amount of used memory approaches the amount of total memory, the JVM attempts to increase the total memory by expanding the heap. Application performance may degrade as the amount of used memory approaches the total available memory.

4.11.2.3 Configuring the Heartbeat

Specify the heartbeat configuration values in the application property files. You can override the default properties by placing entries in the base application property file. See `<BaseProps>` under "Deployment Values" in the *Installation & Migration Guide*.

The heartbeat configuration properties are:

- com.modeln.App.heartbeat.diagnostics
- com.modeln.App.heartbeat.refresh

com.modeln.App.heartbeat.diagnostics

Using the diagnostics parameter you can enable or disable heartbeat diagnostic information. When disabled, any request for diagnostic information is ignored and access is only provided to the heartbeat status message. The possible values for this property are `true` and `false`.

com.modeln.App.heartbeat.refresh

The default interval at which the heartbeat diagnostics are refreshed in the browser window is determined by this refresh interval. If the request does not specify a refresh interval, this value is used as the default interval. The interval is specified in seconds.

Note: For information on how to use statistics to analyze the system, see Analyzing the System Using Statistics.

4.11.3 Monitoring Application Instances

The application JVM instances should be monitored to identify general issues that may significantly degrade an application node. Potential issues include running out of memory or uneven workload across the cluster.

4.11.3.1 CPU Utilization Trend

The CPU utilization for each application server should be measured over time. The utilization is the load on the CPU. The common measurements are either the CPU load or the % utilization.

In most production systems, we usually see the CPU utilization under 50% for most periods, but can climb to higher values during short periods of time (for example, when data flows are running).

Tools

Most monitoring tools provide some CPU monitoring capability. At Model N, we use an open source tool called internally to monitor our servers. For more information, see http://www.zabbix.com.

4.11.3.2 JVM Heap Usage Trend

Java applications allocate memory in the JVM heap and not directly from the operating system. Therefore, we should monitor the JVM heap usage and not the memory allocated by the operating system to the JVM itself.

Periods of Excessive Garbage Collection Activity

A well-behaved system should show a regular "saw tooth" pattern in terms of memory usage. The "saw tooth" shape indicates that memory is being used up, but periodically being freed and reclaimed by the garbage collector (GC). The peaks of the "saw tooth" pattern are typically near the total heap size, but you should see a significant drop during each GC cycle.

If you get into a low-memory condition, then the valleys in the "saw tooth" will increase as less memory is available to be freed. If this continues for too long, then you will see more major GC cycles. Once you get low enough in memory, the JVM will get to the point where it can't free up enough memory and will start continuously issuing major GC cycles.

Average and Maximum Memory Usage

One question that is frequently asked of Model N is how many JVMs may be required in any particular deployment. The required JVM heap memory is one of the primary considerations to answer this question. By monitoring the GC activity, over time the average and total memory consumption can be determined. This can be used to determine if more JVMs are required to support the load or if potentially the JVMs are not being used fully.

Tools

Some monitoring tools provide Java heap monitoring capability. The JVM support for monitoring is evolving rapidly. Java SE 5 introduced significant monitoring and management support. The `jconsole` application provides a straightforward means for monitoring the JVM heap usage as well as other metrics, such as threads. For details on jconsole, see http://java.sun.com/developer/technicalArticles/J2SE/jconsole.html.

Sun JVM Garbage Collection Logging

To turn on garbage collection logging within the Sun JVM, use the JVM arguments:

```
-verbose:gc -XX:+PrintGCTimeStamps -XX:+PrintGCDetails
-Xloggc:<logfile>
```

where `<logfile>` is where the GC log entries will go.

IBM JVM Garbage Collection Logging

To turn on garbage collection logging within the IBM JVM, use the JVM arguments:

```
-verbose:gc
```

and the output goes to either the `native_stderr.log` (Windows/Linux) or `native_stdout.log` (Solaris). This can also be accomplished through the WebSphere management console.

The IBM GC and Memory Visualizer tool supports the IBM GC logging format. This tool provides a good visualization and supports monitoring for more than just garbage collection. For details on the IBM GC and Memory Visualizer, see http://www.ibm.com/developerworks/java/library/j-ibmtools2/index.html.

4.12 Server Monitoring

This section provides information on how to monitor web servers and application servers.

4.12.1 Log Rotation

Log rotation prevents server logs from continuously growing until they exceed the maximum file size allowed by the operating system. Most servers provide log rotation which truncates and moves log files when they reach a configurable size limit.

Ongoing maintenance may be required to remove old log files or to manually rotate logs which are not rotated automatically. Failing to perform this routine maintenance can lead to file system capacity issues and possible server crashes. Take steps to ensure that server logs are routinely archived and removed from the server.

4.12.2 Process Monitoring

Process monitoring is a technique used to determine whether an operating system process or thread is currently running on the system. If the process associated with an application such as Oracle or WebSphere is not running on the system, it is likely that the application has crashed or been shut down. Monitoring the operating system process list is necessary to alert the operations staff when attention is required to restart the application.

To determine if a server process is running, an administrator or monitoring process can query the operating system for a process name or process ID (PID). A process ID is a unique identifier that can be used to refer to a single process or thread within the operating system. Many servers record the process ID of their main process by generating a PID file in the log directory when the server application is started. It is recommended that this process ID file be used to monitor the availability of the server process and alert the operations staff when the process is unavailable.

PID files are typically created as plain text files that contain the process ID of the main server thread. For example, when the IBM IHS web server is started, it writes a PID file in the logging directory. This PID file is then used to determine which process to shut down when instructed to stop the server. Using PID files is an excellent way to determine which process to monitor.

4.12.2.1 Windows Process Monitoring

To monitor Windows processes, go to the **Task Manager > Processes** tab.

To view more detailed information about the processes, modify the columns displayed by navigating to **View > Select** columns.

4.12.2.2 Linux Process Monitoring

To monitor Linux processes by searching for a specific process name, enter:

```
ps -ef | grep <name>
```

To display the parent/child relationship between processes, enter:

```
ps -xf
```

4.12.2.3 Solaris Process Monitoring

To monitor Solaris processes by searching for a specific process name, enter:

```
ps -ef | grep <name>
```

4.12.3 Port Monitoring

Port monitoring is a technique frequently used to determine if a server is responding to client requests on a specific TCP or UDP network port. You can use port monitoring to alert the operations staff when a service is failing to respond to client requests on a network port.

4.12.4 Web Server Monitoring

This section provides information on monitoring the following web servers:

- Sun
- IBM HTTPD
- Microsoft IIS

4.12.4.1 Sun Web Server

Port Monitoring

Port	Protocol	Purpose
80	HTTP	Web Server
8888	HTTP	Administration Server

Process Monitoring

Name	Purpose
webservd	Responsible for responding to incoming HTTP requests.

Log Monitoring

Type	Location
Access	`<installdir>/<instance>/logs/access`
Errors	`<installdir>/<instance>/logs/errors`

4.12.4.2 IBM HTTPD

Port Monitoring

Port	Protocol	Purpose
80	HTTP	Web Server

Process Monitoring

Name	Purpose
HTTPD	Responsible for responding to incoming HTTP requests.

Log Monitoring

Type	Location
PID	`<installdir>/logs/pid`
Access	`<installdir>/logs/access_log`
Errors	`<installdir>/logs/error_log`

4.12.4.3 Microsoft IIS

Port Monitoring

Port	Protocol	Purpose
80	HTTP	Web Server

Process Monitoring

Name	Purpose
IIS Admin Service	This is the control unit that is used to verify that the IIS web server is running.
World Wide Web Publishing Service	This is the engine that powers the IIS Admin Service.

Log Monitoring

Type	Location
General	`C:\WINDOWS\system32\LogFiles\W3SVC1`

4.12.5 Application Server Monitoring

This section provides information on monitoring the following web servers:

- IBM WebSphere
- Oracle WebLogic

4.12.5.1 IBM WebSphere

Port Monitoring

Port	Protocol	Purpose
9060	HTTP	WebSphere Administration Console

Process Monitoring

Name	Purpose
Java	Application server threads (grep by `<server>`). There is a PID file that indicates the root process used to start the server instance.

Log Monitoring

Type	Location
PID	`<installdir>/profiles/<node>/logs/<server>/<server>.pid`
Stdout	`<installdir>/profiles/<node>/logs/<server>/SystemOut.log`
Stderr	`<installdir>/profiles/<node>/logs/<server>/SystemErr.log`
FFDC	`<installdir>/profiles/<node>/logs/fddc/<server>_exception.log`

Note: FFDC logs get generated only when there are exceptions.

4.12.5.2 Oracle WebLogic

Port monitoring

Port	Protocol	Purpose
7001	HTTP	Administration Console

Process Monitoring

Name	Purpose
Java	Application server threads (grep by `<server>`)

Log Monitoring

Type	Location
Access	<installdir>/user_projects/domains/<domain>/servers/<server>/logs/access.log
Stdout	<installdir>/user_projects/domains/<domain>/servers/<server>/logs/<server>.log
Stderr	<installdir>/user_projects/domains/<domain>/servers/<server>/logs/<server>.out

4.12.6 Model N Application

This section provides information on monitoring the Model N application using the Performance Monitoring Application (PMA).

The PMA uses an in-memory and a database strategy to monitor threads, queries, user requests and sessions in the system. The combined in-memory and database strategy achieves the following:

- In-memory to give real-time information about the system by way of JMX and not slow the system when capturing and displaying this information.
- Saving to the database so that in-memory structures do not grow too big.

The main features of the in-memory trackers are:

- Ability to display the last few tracked items.
- Aggregations of events that occur too often.

  For example, the PMA does not track every query that ran, but tracks only the slowest queries and queries that ran too often.
- Write out to the database at a configurable interval.
- An HTTP JMX adapter, so that no special client is needed to access the management beans.
- Once saved to the database, the in-memory information is no longer available.

  For example, only data for the last few minutes is available in memory

Note: The tracking is turned on by default in the application.

Persisting older information back to the database helps you to:

- Analyze older data.
- Write ad-hoc reports to present a user interface for older data such as concurrency report, query report, request report.

To prevent the tracker database tables growing indefinitely, there is a purging mechanism by way of the PurgeTrackers timer that purges records from the tracker tables. This timer is configured to run once daily and the SQL for purging is defined via properties.

For information on how to use SQL to retrieve and analyze data using the PMA, see the appendix, Performance Monitoring Application SQL.

In-Memory Trackers

PMA contains the following in-memory trackers:

- Task Tracker

  The task tracker bean tracks both backend threads, such as commands, and timers, and user request threads. Requests in memory are persisted to the database on a configurable timer.

  The task tracker also keeps track of the total server and query times for tasks that are not persisted, so that you can see system totals. The task trackers also lets you persist tasks that take a long time.

  As of the 5.3 release, task tracker has added the following two methods:

  Code 4-15: Task Tracker

```
IMnTaskTracker {
    /** Return all the user requests in-memory since the last
save
     *  @return CMnTaskInfo[] all the request tasks
     */
    public CMnTaskInfo[] getRequests();

    /** Return all the currently running tasks
     *  @return CMnTaskInfo[] all the running tasks
     */
     public Object[] getRunningTasks();
}
```

  The `CMnTaskInfo` class tracks information for a user request or back-end server thread. Following are the attributes in this class:
  - `jvmTime` - the time the thread took as reported by the JVM
  - `uiBytesWritten` - the size in bytes of the response to this user request

- Session Tracker

  The session tracker bean is used to track user sessions in the application. The currently active sessions are always in memory while sessions that have been inactivated are persisted in the database in the UserSession FGO.

- Health Tracker

  The health tracker bean is new in the 5.0.10 release and tracks the following:

- Unique JVM ID
- Total JVM memory
- Free JVM memory
- Total request tasks started
- Total backend tasks started
- Total sessions active
- Total select queries run
- Total insert queries run
- Total update queries run
- Total other queries run
- Total connection pool uses
- Total connection pool wait time

Model N Monitoring Table Dump

When you run into performance problems, data in the Model N tracking tables will help Model N customer support to understand activity in the system prior to and at the time performance problems were discovered. To enable Model N to analyze performance problems in your application, you can export only the tables used for tracking user and thread activity using the following Oracle command:

```
exp userid=username/password@tnsname
tables=(mn_hist_session,mn_hist_task,mn_hist_query_usage,mn_h
ist_
query_handle,mn_hist_query, mn_hist_health)
file=ModelNMonitoring_mmddyyyy_hhmi.dmp rows=yes indexes=yes
```

The tables will be exported to the dump file specified. Please provide this dump file to Model N customer support.

Using Cognos Reports to Analyze PMA Data

The reports deploy automatically if you have properly set the Cognos deployment properties and the deployment task is in the ant task list. For more information on the report deployment tools, see the section on Operations Tools in the Deployment Overview chapter of the Model N *Installation & Migration Guide*.

You must have installed Cognos ReportNet MR4 so that it is able to view the PMA tables as a data source in order to generate reports. For information on how to install Cognos, see the Cognos ReportNet chapter in the Model N *Installation & Migration Guide*.

The PMA reports are in the Cognos folder `public/PMA`. Move them using the Cognos administration tool to `public/standard` before using them. If the PMA reports are not in the PMA folder, import them from `PlatformModelsAndReports.zip`, which is in `/platform/external/cognos/`deployment. Before you import the reports, make sure that:

- The schema resolver tool has been used on the `PlatformModelsAndReports.zip` file. This happens automatically when the

Cognos deployment properties are set properly.

- The reports have been copied to the Cognos deployment folder.

The following high-level reports are then available using Report Studio and through the Model N Reporting tab:

- PMA – Overview

  A snapshot report containing aggregate information about application performance.

- PMA – Session Report

  A list of all user sessions and related data about them.

- PMA – Usage Centers

  A list of which areas of the application are taking the most resources, with some drill-through functionality enabled.

- PMA - Backend Task List

  A list of all backend tasks sorted by query time and links to the details of the queries run by the task.

- PMA - Queries Overview

  A list of which areas of the database are taking the most resources.

- PMA - Email Notifications

  A list of email notification messages within a given date range and of a delivery status.

All of the preceding top-level reports contain links to more detailed lower-level reports that assist in locating the source of any problems. The following tables lists the lower-level reports:

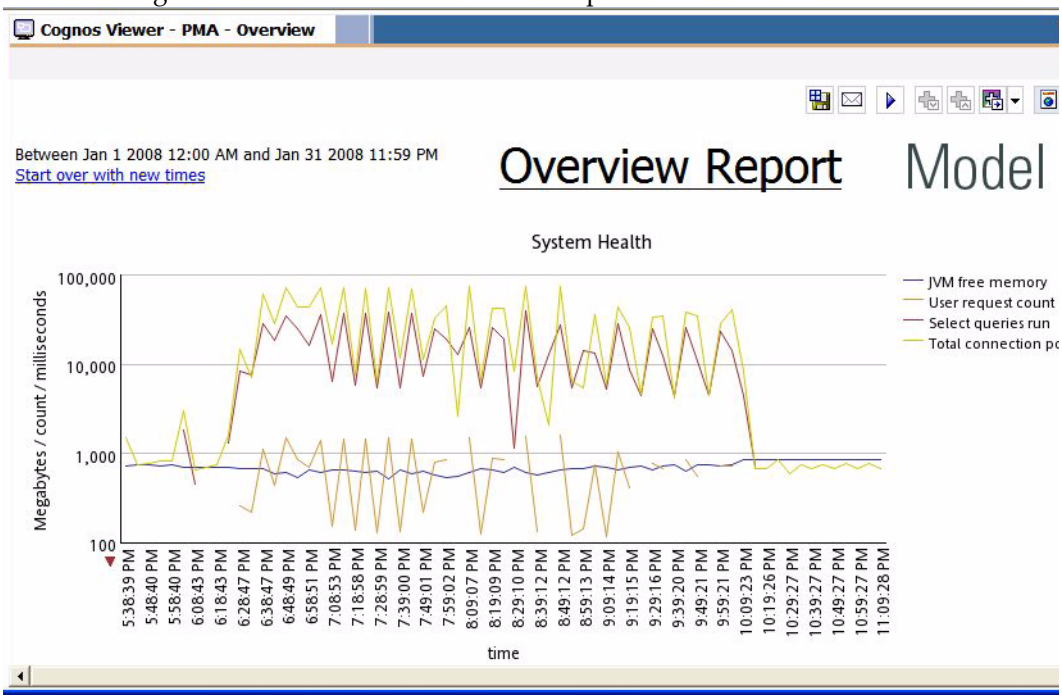
Table 4-14: Low-Level PMA Reports

Report Name	Parameters	Sorted By
Query Report	Query ID	Total query time
Query Tasks	Query ID	Query time
Resource Overview	Resource Key	Server time
Resource Queries	Resource Key	Query time
Resource Tasks	Resource Key	Server time
Session Queries	Session ID	Usage ID
Session Tasks	Session ID	Session Task ID
Task Queries	Task ID	Query time

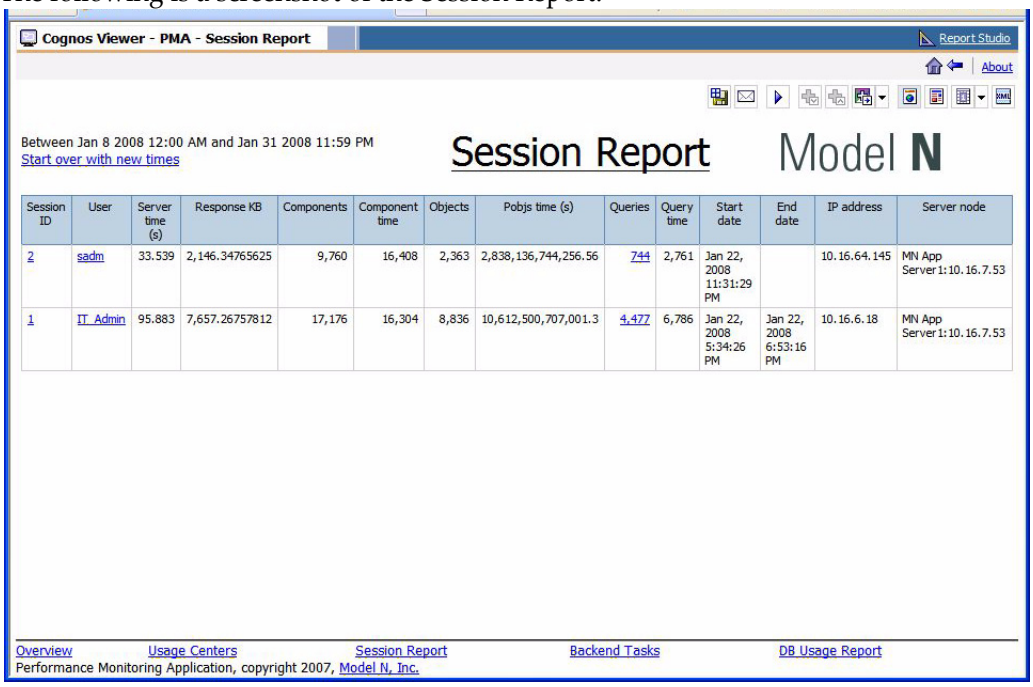
Table 4-14: Low-Level PMA Reports (Continued)

Report Name	Parameters	Sorted By
ThreadType Queries	Thread Type	Query time
ThreadType Tasks	Thread Type	Server time
Usage	Query Usage ID	Query time
User	User name	Server time
User queries	User name	Query time
User tasks	User name	Server time
Cluster Health		
Cluster Health - this week		
Cluster Health - today		
Node Health	Node ID	
Node Health - this week	Node ID	
Node Health - today	Node ID	
Task Query Uses	Task ID, Query ID	Query time
Thread and Resource Overview	Thread Class Name, Resource Key	Server time
User Query Tasks	User name, Query ID	Query time

The following is a screen shot of the Overview Report:

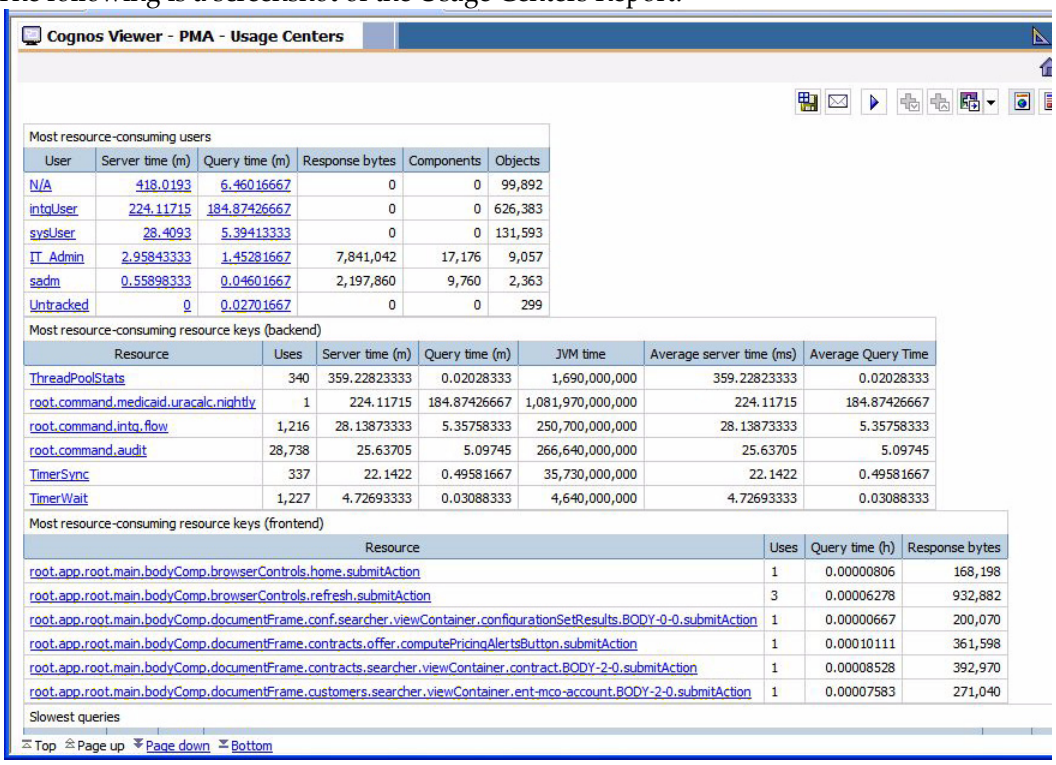


The following is a screenshot of the Session Report:



Session ID	User	Server time (s)	Response KB	Components	Component time	Objects	Pobjs time (s)	Queries	Query time	Start date	End date	IP address	Server node
2	sadm	33.539	2,146.34765625	9,760	16,408	2,363	2,838,136,744,256.56	744	2,761	Jan 22, 2008 11:31:29 PM		10.16.64.145	MN App Server1:10.16.7.53
1	IT Admin	95.883	7,657.26757812	17,176	16,304	8,836	10,612,500,707,001.3	4,477	6,786	Jan 22, 2008 5:34:26 PM	Jan 22, 2008 6:53:16 PM	10.16.6.18	MN App Server1:10.16.7.53

The following is a screenshot of the Usage Centers Report:



Cognos Viewer - PMA - Usage Centers

Most resource-consuming users

User	Server time (m)	Query time (m)	Response bytes	Components	Objects
N/A	418.0193	6.46016667	0	0	99,892
intgUser	224.11715	184.87426667	0	0	626,383
sysUser	28.4093	5.39413333	0	0	131,593
IT_Admin	2.95843333	1.45281667	7,841,042	17,176	9,057
sadm	0.55898333	0.04601667	2,197,860	9,760	2,363
Untracked	0	0.02701667	0	0	299

Most resource-consuming resource keys (backend)

Resource	Uses	Server time (m)	Query time (m)	JVM time	Average server time (ms)	Average Query Time
ThreadPoolStats	340	359.22823333	0.02028333	1,690,000,000	359.22823333	0.02028333
root.command.medicaid.uracalc.nightly	1	224.11715	184.87426667	1,081,970,000,000	224.11715	184.87426667
root.command.intg.flow	1,216	28.13873333	5.35758333	250,700,000,000	28.13873333	5.35758333
root.command.audit	28,738	25.63705	5.09745	266,640,000,000	25.63705	5.09745
TimerSync	337	22.1422	0.49581667	35,730,000,000	22.1422	0.49581667
TimerWait	1,227	4.72693333	0.03088333	4,640,000,000	4.72693333	0.03088333

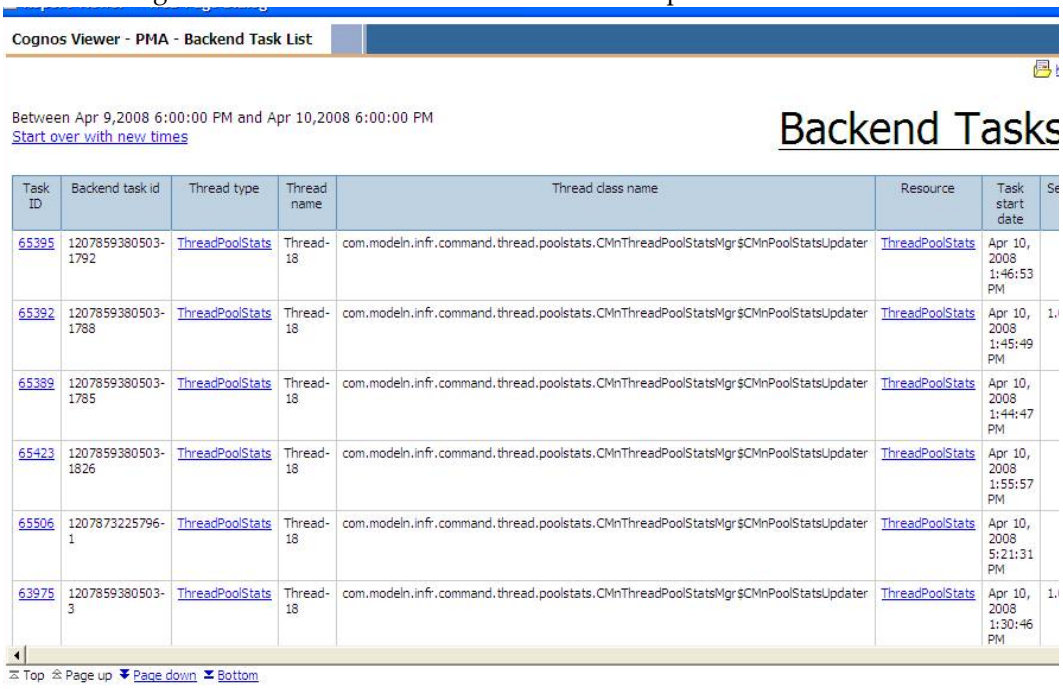
Most resource-consuming resource keys (frontend)

Resource	Uses	Query time (h)	Response bytes
root.app.root.main.bodyComp.browserControls.home.submitAction	1	0.00000806	168,198
root.app.root.main.bodyComp.browserControls.refresh.submitAction	3	0.00006278	932,882
root.app.root.main.bodyComp.documentFrame.conf.searcher.viewContainer.configurationSetResults.BODY-0-0.submitAction	1	0.00000667	200,070
root.app.root.main.bodyComp.documentFrame.contracts.offer.computePricingAlertsButton.submitAction	1	0.00010111	361,598
root.app.root.main.bodyComp.documentFrame.contracts.searcher.viewContainer.contract.BODY-2-0.submitAction	1	0.00008528	392,970
root.app.root.main.bodyComp.documentFrame.customers.searcher.viewContainer.ent-mco-account.BODY-2-0.submitAction	1	0.00007583	271,040

Slowest queries

Top Page up Page down Bottom

The following is a screenshot of the Backend Threads Report:



Cognos Viewer - PMA - Backend Task List

Between Apr 9,2008 6:00:00 PM and Apr 10,2008 6:00:00 PM

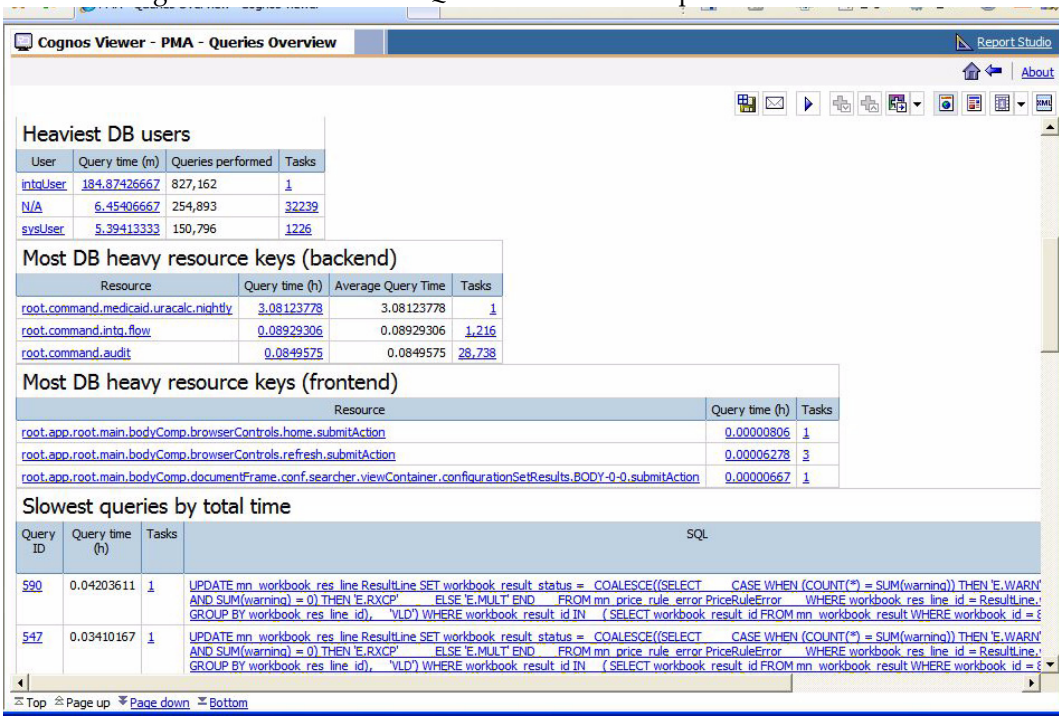
Start over with new times

Backend Tasks

Task ID	Backend task id	Thread type	Thread name	Thread class name	Resource	Task start date
65395	1207859380503-1792	ThreadPoolStats	Thread-18	com.modeln.infr.command.thread.poolstats.CMnThreadPoolStatsMgr$CMnPoolStatsUpdater	ThreadPoolStats	Apr 10, 2008 1:46:53 PM
65392	1207859380503-1788	ThreadPoolStats	Thread-18	com.modeln.infr.command.thread.poolstats.CMnThreadPoolStatsMgr$CMnPoolStatsUpdater	ThreadPoolStats	Apr 10, 2008 1:45:49 PM
65389	1207859380503-1785	ThreadPoolStats	Thread-18	com.modeln.infr.command.thread.poolstats.CMnThreadPoolStatsMgr$CMnPoolStatsUpdater	ThreadPoolStats	Apr 10, 2008 1:44:47 PM
65423	1207859380503-1826	ThreadPoolStats	Thread-18	com.modeln.infr.command.thread.poolstats.CMnThreadPoolStatsMgr$CMnPoolStatsUpdater	ThreadPoolStats	Apr 10, 2008 1:55:57 PM
65506	1207873225796-1	ThreadPoolStats	Thread-18	com.modeln.infr.command.thread.poolstats.CMnThreadPoolStatsMgr$CMnPoolStatsUpdater	ThreadPoolStats	Apr 10, 2008 5:21:31 PM
63975	1207859380503-3	ThreadPoolStats	Thread-18	com.modeln.infr.command.thread.poolstats.CMnThreadPoolStatsMgr$CMnPoolStatsUpdater	ThreadPoolStats	Apr 10, 2008 1:30:46 PM

Top Page up Page down Bottom

The following is a screenshot of the Queries Overview Report:



Deploying PMA Reports

To deploy PMA reports:

1. Copy the latest version of `PlatformModelsAndReports.zip` to the deployment folder of the Cognos installation
2. In the Cognos Administration panel, go to **Launch > Cognos Administration**.
3. Click the Configuration tab at the top.
4. Click the content administration tab on the left.
5. Click New Import.
6. Import the reports in the PlatformModelsAndReports.zip file.
7. Click Home at the top of the page.
8. Click the PMA folder, select all objects, and click Cut.
9. Navigate to the `Standar` folder, and click Paste.

The reports are now accessible from the Administration page and from inside the application.

Properties

Following are properties let you to monitor the Model N application:

Table 4-15: Model N Monitoring Properties

Property	Description	Default Values
com.modeln.track.beans	List of all monitoring beans to be initialized in CMnEnv. The name that is given here is used to register this bean with the JMX MbeanServer.	System User_Sessions User_And_Server_Threads
com.modeln.track.saver.sql.cutoff	This is the first few characters of the SQL to be compared against the clob in the database. **Note:** For performance reasons, do not modify it to be more than 1000.	1000
com.modeln.track.enable	Determines whether tracking is enabled.	true

Application Switches

Following are properties let you to monitor the Model N application:

Table 4-16: Model N Monitoring Application Switches

Property	Description	Default Values
com.modeln.PMA Switches.query.threads.storeGreaterThanRunCount	Store only those queries for server threads that run more than X number of times.	15
com.modeln.PMA Switches.query.threads.storeGreaterThanRunTime	Store only those queries for server threads that run for more than X number of milliseconds.	180000
com.modeln.PMA Switches.query.requests.storeGreaterThanRunCount	Store only those queries for user requests that run more than X number of times.	15

Table 4-16: Model N Monitoring Application Switches

Property	Description	Default Values
`com.modeln.PMA Switches.query. requests.store GreaterThanRunTime`	Store only those queries for user requests that run for more than X number of milliseconds.	15000
`com.modeln.PMA Switches.saver. waitTimeSeconds` **Note:** Contact Model N before modifying this property.	This is the time the saver thread sleeps. This thread is used for flushing the in-memory tracker beans to the database.	300
com.modeln.PMA Switches.cpu.store GreaterThanRunTime **Note:** Contact Model N before modifying this property.	Saves tasks that have a total server time greater than X number of milliseconds.	15000
com.modeln.PMA Switches.task.query. storeGreaterThan TotalTime **Note:** Contact Model N before modifying this property.	Saves tasks that spent longer than a cutoff value of total running queries.	15000

Configurations

The following configurations are used by the PurgeTrackers timer to delete old records from the tracker tables. As of the 5.3 release, you can locate these configurations (previously properties) using the user interface's Configuration Console:

Table 4-17: PurgeTrackers Timer Configurations

Configuration	Description
`com.modeln.HistoricalQueryUsage.ora.query.DELETE_OLD_QUERYUSAGES`	This configuration is located in the **Configuration Console > a configuration > Business Objects > HistoricalQueryUsage > DELETE_OLD_QUERYUSAGES**. This query deletes old query usages and is used internally by the Model N PurgeTrackers timer. The criterion to delete should match the criterion in `DELETE_OLD_TASKS`. For example, if you delete tasks older than seven days, you should also delete query usages older than seven days. Otherwise you may run into foreign key constraint problems while deleting old tasks. The default value is: `DELETE FROM [DF] WHERE task_id IN \(SELECT task_id FROM mn_hist_task \WHERE date_created +7 < [?:1])`
`com.modeln.HistoricalTask.ora.query.DELETE_OLD_TASKS`	This configuration is located in the **Configuration Console > a configuration > Business Objects > HistoricalTask > DELETE_OLD_TASKS**. This query deletes old user and server thread information and is used internally by the Model N PurgeTrackers timer. The criterion to delete should match the criterion in `DELETE_OLD_QUERYUSAGES`. For example, if you delete tasks older than seven days, you should also delete query usages older than seven days. Otherwise you may run into foreign key constraint problems while deleting old tasks. The default value is: `DELETE FROM mn_hist_task \WHERE date_created +7 < [?:1`

Table 4-17: PurgeTrackers Timer Configurations (Continued)

Configuration	Description
`com.modeln.HistoricalSession.ora.query.DELETE_OLD_SESSIONS`	This configuration is located in the **Configuration Console > a configuration > Business Objects > HistoricalSession > DELETE_OLD_SESSIONS**. This query deletes old user and server thread information and is used internally by the Model N PurgeTrackers timer. The default value is: `DELETE FROM mn_hist_session \WHERE date_created +60 < [?:1]`
com.modeln.HistoricalHealth.ora.query.DELETE_OLD_HEALTH	This configuration is located in the **Configuration Console > a configuration > Business Objects > HistoricalHealth > DELETE_OLD_HEALTH**. This query purges old system health data. The default value is: `DELETE FROM [DF] WHERE date_created +60 < [?:1]` You can change the 60 to any number of days.

The following properties already existed and are listed here because they relate to monitoring the Model N application:

Table 4-18: Model N Monitoring Properties

Property	Description
`com.modeln.jmxEnabled`	When set to `true`, this property enables the JMX server within the Model N application.
`com.modeln.jmxHttpDisabled`	When set to `false`, this property enables the integrated HTTP adapter on the JMX server. The tracker beans can then be viewed via the following URL: `http://hostname:portNumber` Default value: false
`com.modeln.jmxHttpPort`	The port for the JMX HTTP adapter. The tracker beans can then be viewed via the following URL: `http://hostname:portNumber` Default Value: 10100

Table 4-18: Model N Monitoring Properties

Property	Description
com.modeln.jmxRmiDisabled	When set to false, enables the RMI connector on the JMX server. JMX consoles can then connect via the url: `service:jmx:rmi:///jndi/rmi://hostname:portNumber/jmxrmi` to look at the beans exposed via JMX. Default Value: true
com.modeln.jmxRmiPort	Used to specify the port number for the RMI connector. Default Value: 10099

Using JMX to Look at Real Time In-Memory Beans

JMX technology provides a tiered architecture where managed resources and management applications can be integrated in the plug-and-play approach as shown in Figure 1. A given resource is instrumented by one or more Java objects known as Managed Beans (or MBeans), which are registered in a core managed object server known as the MBean server. This server acts as a management agent and can run on most Java-enabled devices.

The following tiers, or levels, exist in this architecture:

- Instrumentation
- Agent
- Manager

Instrumentation Level

This tier contains MBeans and their manageable resources. In the Model N application, all the following trackers are used to monitor the Model N application and are exposed as MBeans:

- `User_Sessions`

  Bean used to track current user sessions.

- `User_And_Server_Threads`

  Bean used to track the last few user requests and back-end threads.

Agent Level

This tier contains the JMX agents used to expose the MBeans. The Model N application exposes the MBeans via the integrated HTTP adapter.

Manager Level

This tier contains the components that enable management applications to communicate with JMX agents. You can use the integrated HTTP adapter as the console to look at the JMX beans exposed in the Model N application.

Using JManage

JManage is a web-based JMX console. Version 1.0 runs with JDK 1.4 and version 2.0 runs with JDK 1.5.

To run JManage:

1. Download JManage from www.jmanage.org.
2. Start JManage by following the instructions from the website.
3. Connect to JManage (the default URL is http://hostname:9090).

   Add a JSR 160 application to be managed by specifying the URL in the following format:

   service:jmx:rmi:///jndi/rmi://hostname:portNumber/jmxrmi
4. Go into the application and locate managed objects.

   Now you can view values for the various MBeans as shown here.

JManage also has a command line interface that you can use to script out getting values from the application. It still uses the JManage server as the proxy for communicating with the JMX-enabled applications. For more information on the command line interface, go to http://jmanage.org/wiki/index.php/Command_Line.

Following is an example of the command line (after you change the directory to `JMANAGE_HOME/bin`):

```
./jmanage.sh -username admin -password 123456 print 50App/
com.modeln:name=User_Sessions/Sessions
```

This command prints out the sessions in the following tabular format.

Figure 4-17: Sample JVM Sessions

```
SessionId     UserId     StartDate                       LastRequestDate                 ClientIpAddr
---------     ------     ---------                       ---------------                 ------------
221           sadm       Thu Aug 02 12:22:15 PDT 2007    Thu Aug 02 12:27:17 PDT 2007    10.16.6.103
```

Using The Integrated HTTP Adapter to Monitor Model N

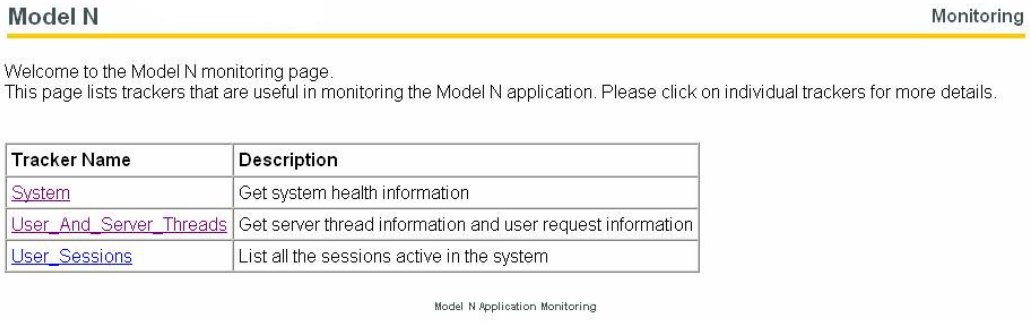
The Model N application is bundled with an HTTP Adapter for JMX which you can use to see real-time status of tracker beans in the system, to see all the active user sessions in the system, and to see information on the last few user requests and user threads that the Model N application serviced.

The following properties control the HTTP Adapter:

Table 4-19: HTTP Adapter Properties

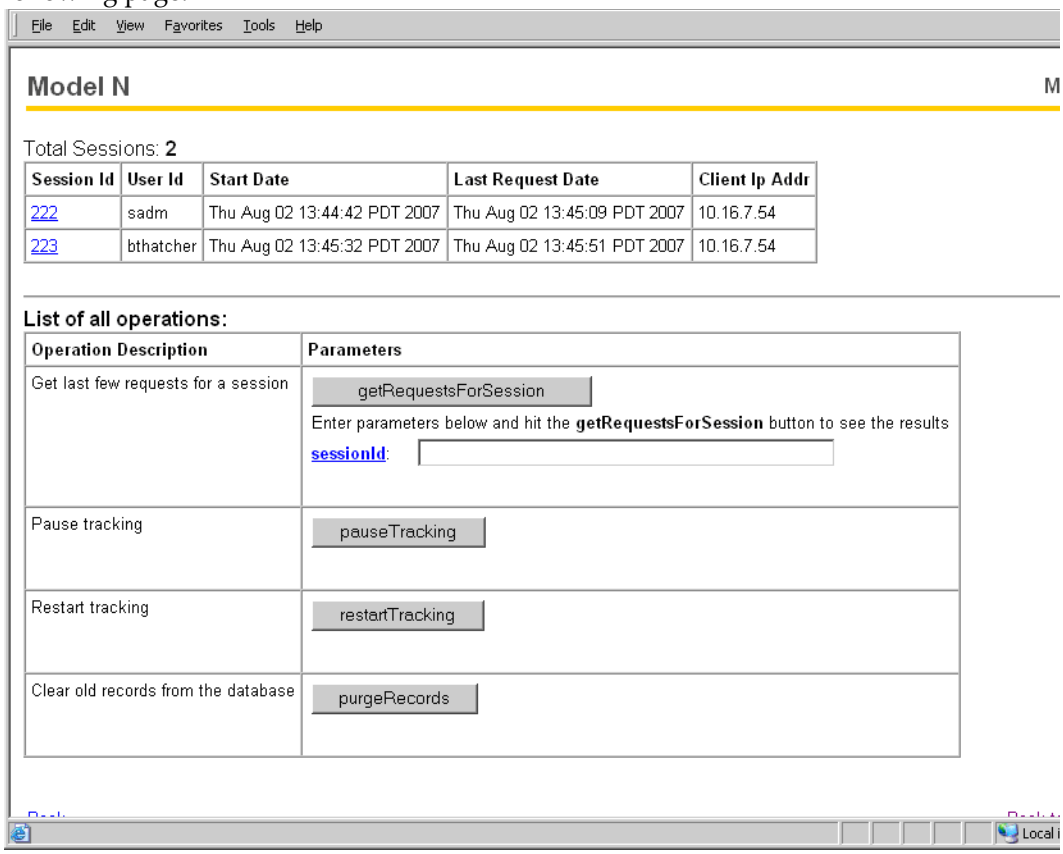
Property	Default Value
`com.modeln.jmxEnabled`	true
`com.modeln.jmxHttpDisabled`	false
`com.modeln.jmxHttpPort`	10100

The HTTP Adapter is enabled by default in the product and is available at port 10100 on the application server. To get to the HTTP Adapter home page, go to `http://host:10100`. The home page appears as follows:



User Sessions Tracker Bean

If you click the `User_Sessions` link in the HTTP Adapter home page, you get to the following page:



This page lists all of the active sessions in the system. To see the last few requests for a session, you can click the session ID link. This page also lists all the operations that can be performed on the `User_Sessions` tracker bean. The operations and their usages are as follows:

Table 4-20: User_Sessions Tracker Bean

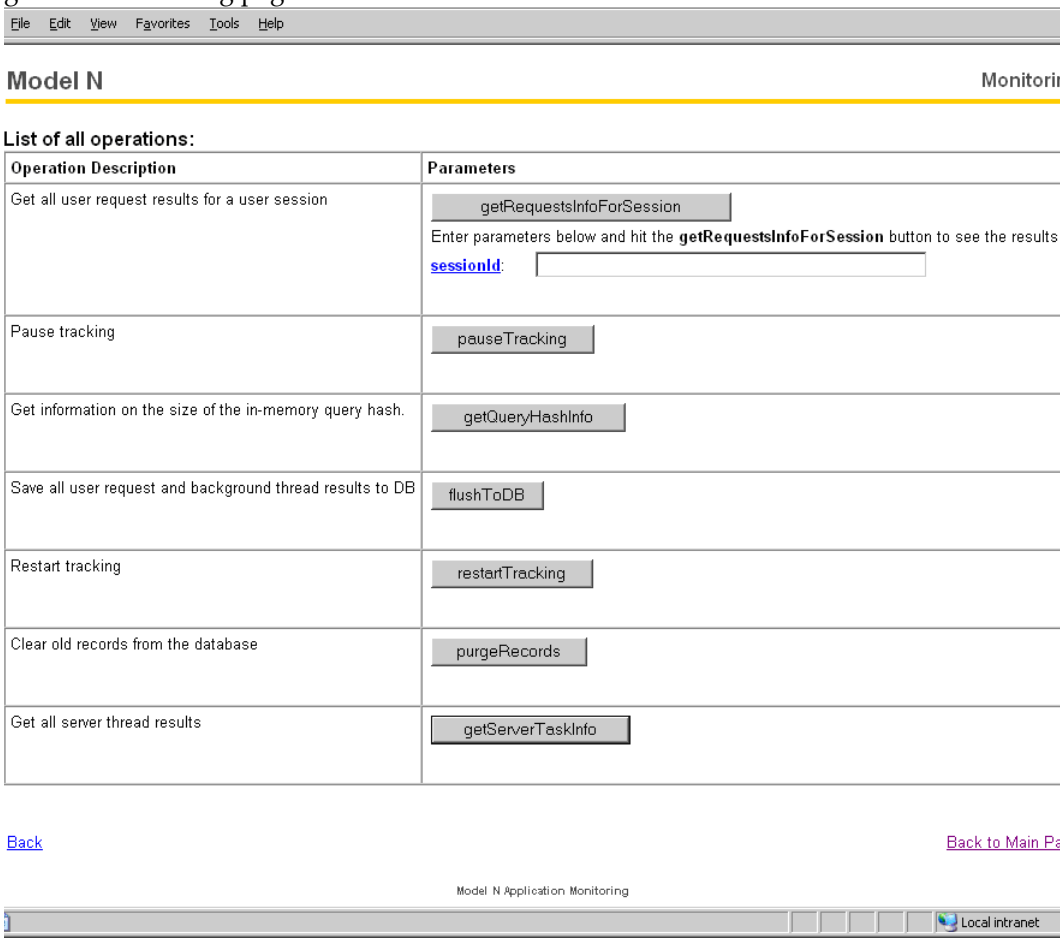
Operation	Usage
`purgeRecords`	This operation removes old records from the database. The default behavior is to delete all sessions older than 60 days from the database.
`pauseTracking`	This operation is used to pause tracking. This should be rarely used. The main use of this operation is to stop tracking in a running system.
`restartTracking`	This operation is used to restart tracking.

Table 4-20: User_Sessions Tracker Bean

Operation	Usage
`getRequestsForSession`	This operation is used to get the last few requests for a session. The same can also be accomplished by clicking on the session id link for individual sessions.

User and Server Threads Tracker Bean

If you click the `User_And_Server_Threads` link on the HTTP Adapter home page, you get to the following page:

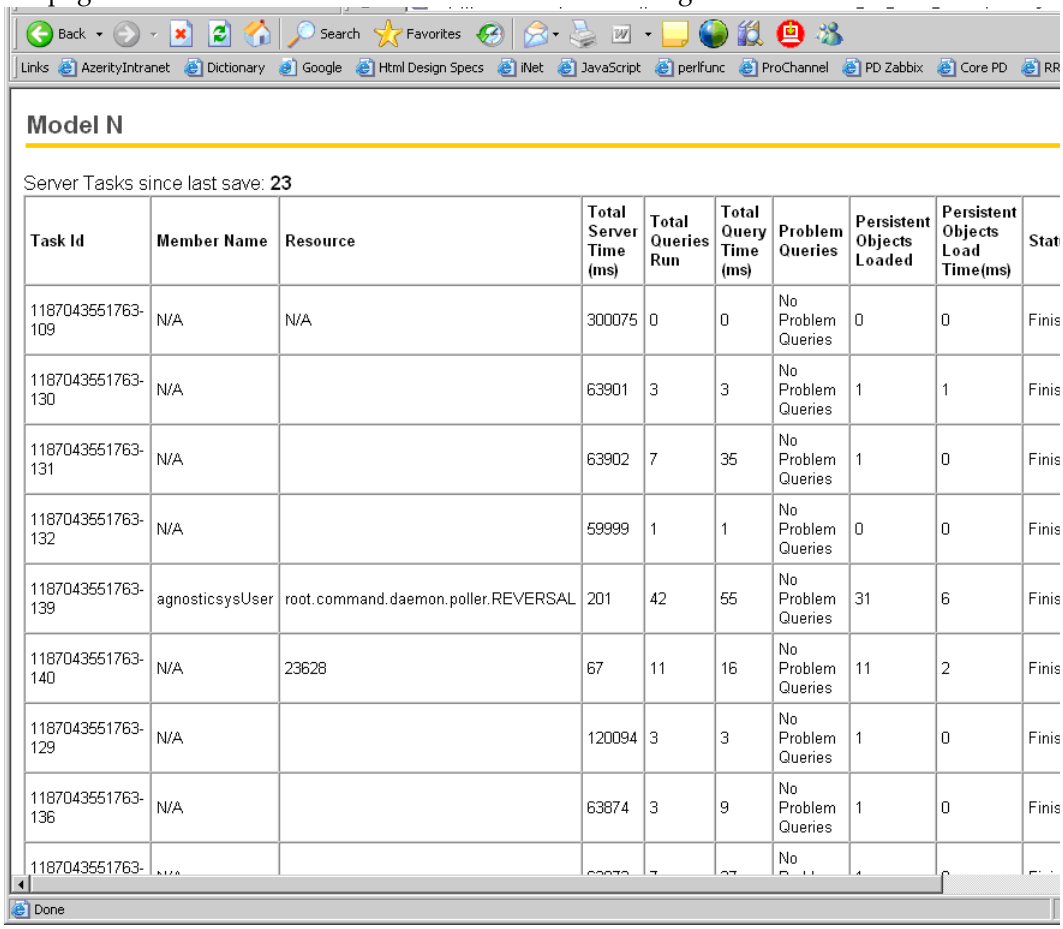


This page lists all the operations that can be performed on the `User_And_Server_Threads` tracker bean. The operations and their usages are as follows:

Table 4-21: User_And_Server_Threads Tracker Bean

Operation	Usage
`getRequestsForSession`	This operation is used to get the last few requests for a session. The same can also be accomplished by clicking on the session id link for individual sessions.
`pauseTracking`	This operation is used to pause tracking. This should be rarely used. The main use of this operation is to stop tracking in a running system.
`getQueryHashInfo`	This operation is meant as a debugging tool to see how big the in-memory query hash has grown.
`flushToDB`	This operation is used to send all in-memory tracker data back to the database. The in-memory data is saved back to the database at five minute intervals in the server but if there is a need to flush the in-memory back to the database immediately, you should use this operation.
`restartTracking`	This operation is used to restart tracking.
`purgeRecords`	This operation removes old records from the database. The default behavior is to delete all user and server threads and queries associated to them that are older than seven days.
`getServerTaskInfo`	This operation retrieves the last few finished server tasks.

The following page is used to get the last few requests for a session and looks similar to the page used to retrieve the last few finished and running server tasks:



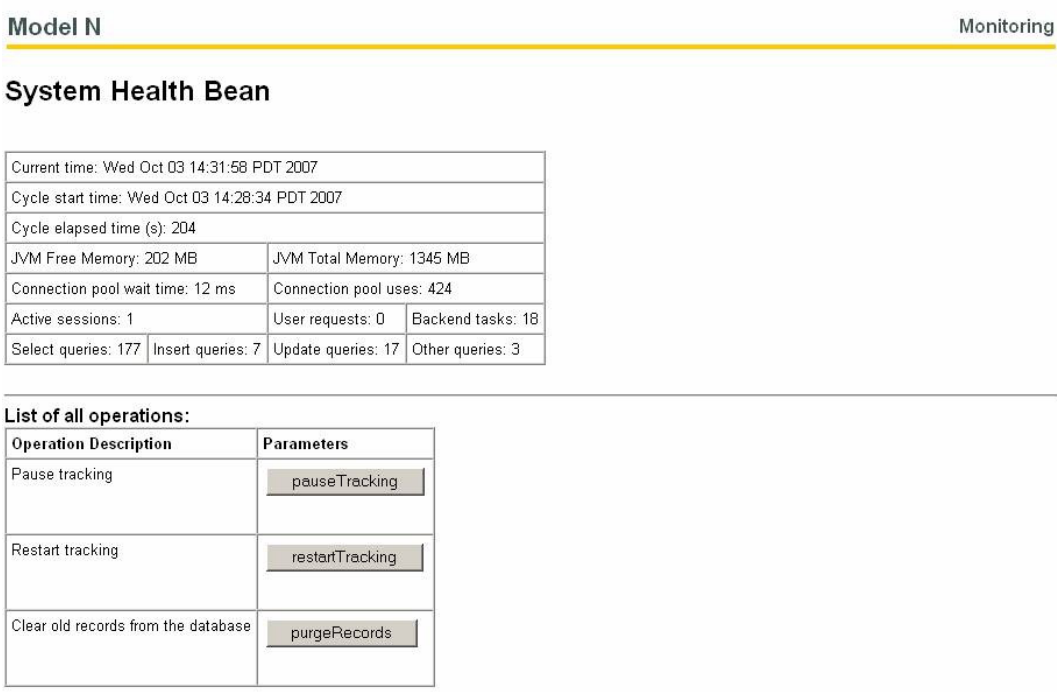
Model N

Server Tasks since last save: **23**

Task Id	Member Name	Resource	Total Server Time (ms)	Total Queries Run	Total Query Time (ms)	Problem Queries	Persistent Objects Loaded	Persistent Objects Load Time(ms)	Stat
1187043551763-109	N/A	N/A	300075	0	0	No Problem Queries	0	0	Finis
1187043551763-130	N/A		63901	3	3	No Problem Queries	1	1	Finis
1187043551763-131	N/A		63902	7	35	No Problem Queries	1	0	Finis
1187043551763-132	N/A		59999	1	1	No Problem Queries	0	0	Finis
1187043551763-139	agnosticsysUser	root.command.daemon.poller.REVERSAL	201	42	55	No Problem Queries	31	6	Finis
1187043551763-140	N/A	23628	67	11	16	No Problem Queries	11	2	Finis
1187043551763-129	N/A		120094	3	3	No Problem Queries	1	0	Finis
1187043551763-136	N/A		63874	3	9	No Problem Queries	1	0	Finis
1187043551763-						No			

System Health Bean

If you click the `System` link on the HTTP Adapter home page, you get to the following page:



This page explains the detailed information that appears on the System Health Bean page.

Table 4-22: System Health Bean

Field	Description
Current time	The current time.
Cycle start time	The time when the counters were last reset to zero.
Cycle elapsed time(s)	The time since the counters were last reset to zero.
JVM Free Memory	The free memory for the node whose JMX page is being viewed.
JVM Total Memory	The total memory for the node whose JMX page is being viewed.
Connection pool wait time	The cumulative wait time of the shared connection pool across all nodes since the counters were last reset to zero.

Table 4-22: System Health Bean

Field	Description
Connect pool uses	The cumulative uses of the shared connection pool across all nodes since the counters were last reset to zero.
Active sessions	The number of sessions currently active on this node.
User requests	The number of front-end requests handled by this node since the counters were reset to zero.
Backend tasks	The number of back-end tasks handled by this node since the counters were reset to zero.
Select queries	The number of select queries run by this node since the counters were last reset to zero.
Insert queries	The number of insert queries run by this node since the counters were last reset to zero.
Update queries	The number of update queries run by this node since the counters were last reset to zero.
Other queries	The number of other queries run by this node since the counters were last reset to zero.
`pauseTracking`	This operation is used to pause tracking. This should be rarely used. The main use of this operation is to stop tracking in a running system.
`restartTracking`	This operation is used to restart tracking.
`purgeRecords`	This operation removes old records from the database. The default behavior is to delete all user and server threads and queries associated to them that are older than seven days.

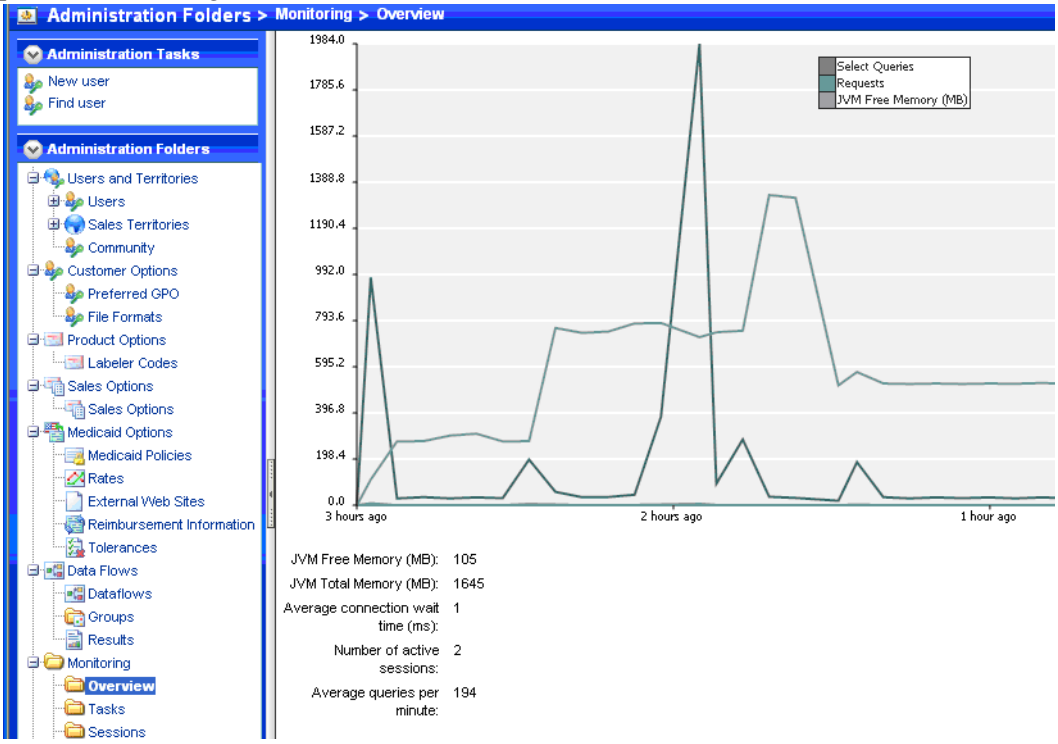
Monitoring Pages

As of the 5.3 release, the IT Admin role can see the following user interface pages that contain information similar to what is in the HTTP adapter:

- Overview Page
- Tasks Page
- Sessions Page
- Queries Page

Overview Page

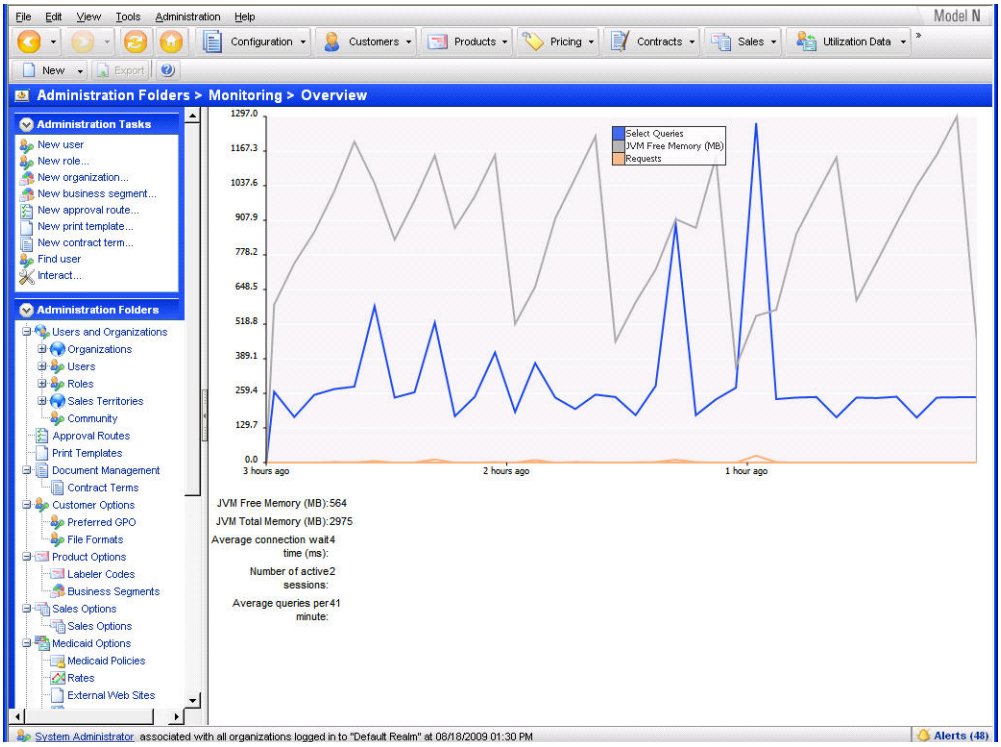
This page displays a graph that provides a view of the last three hours of system performance and gives the values for some current metrics.



4.13 Overview Page

Navigation Path: **Administration > Monitoring > Overview**

Figure 4-18: Overview Page



The Overview page displays a measurement of memory usage by the Model N application.

Table 4-23: Overview Page

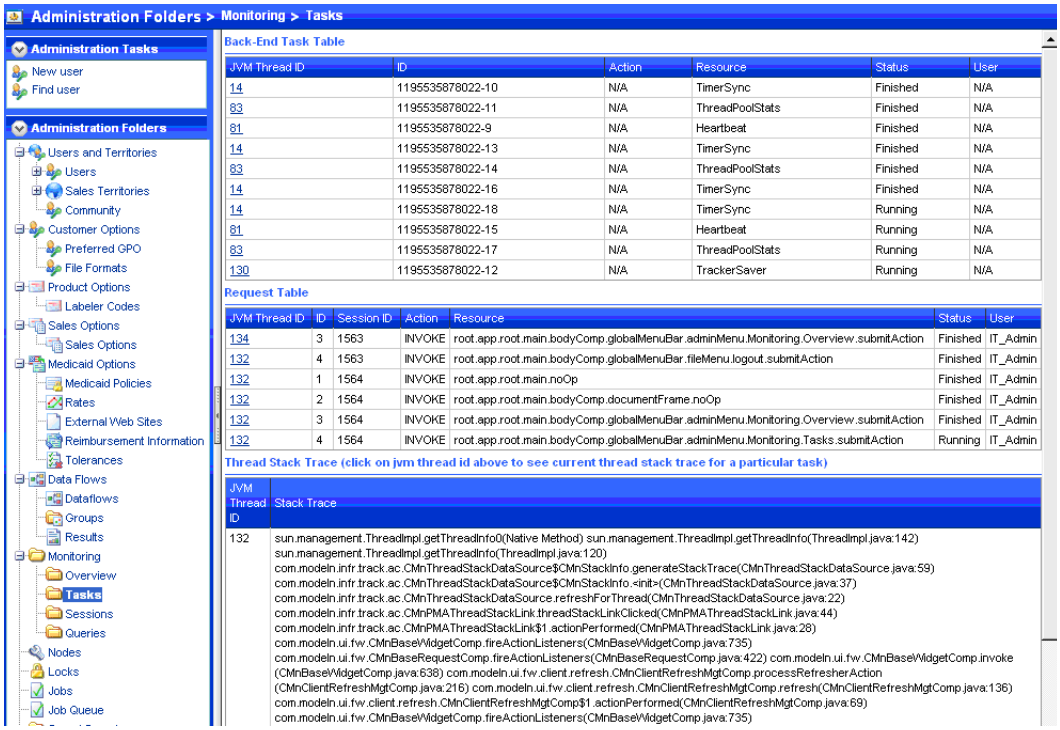
Name	Type	Description
[graph]	Graph	The measurement over time of: • Select queries • JVM free memory • Requests
Details		
JVM Free Memory	String	The amount of JVM free memory in MB.
JVM Total Memory	String	The amount of JVM total memory in MB.
Average connection wait time	String	The average connection wait time in milliseconds.

Table 4-23: Overview Page (Continued)

Name	Type	Description
Number of active sessions	String	The number of active sessions.
Average queries per minute	String	The average number of queries per minute.

Tasks Page

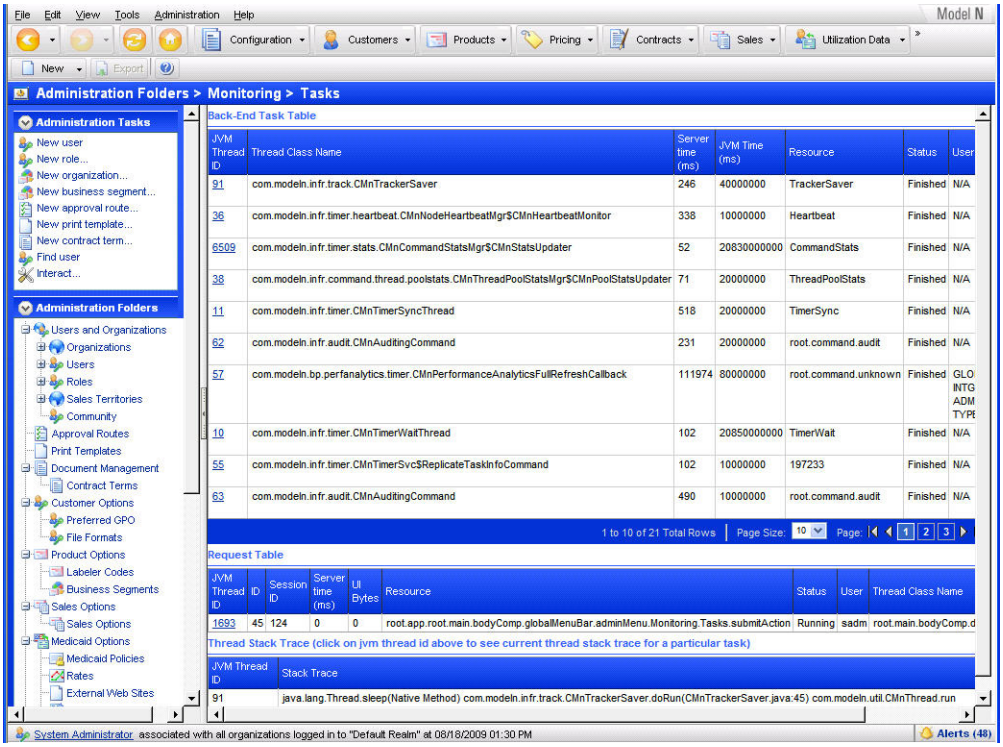
The Tasks page shows a list of all backend tasks performed and user requests sent since the last save. You can see the stack trace for a particular task by clicking the task ID.



4.14 Tasks Page

Navigation Path: **Administration > Monitoring > Tasks**

Figure 4-19: Tasks Page



The Tasks page tracks both backend threads, such as commands and timers, and user request threads.

Table 4-24: Tasks Page

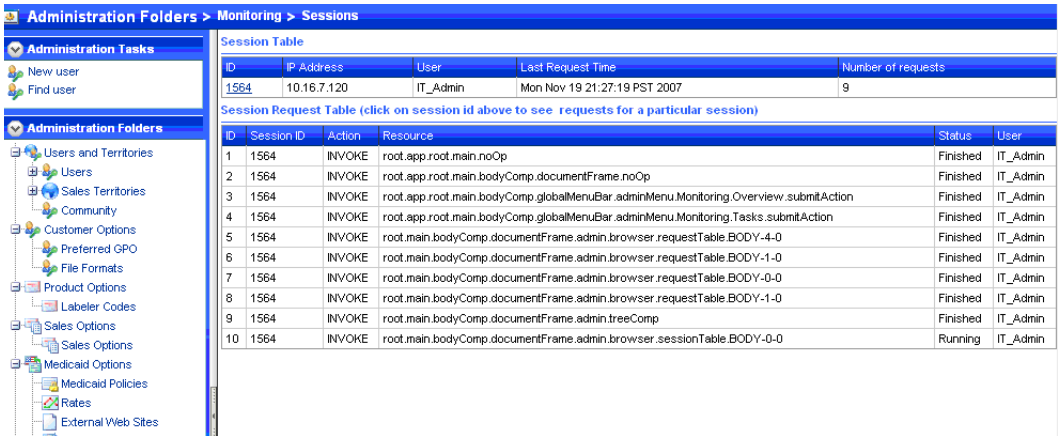
Name	Type	Description
Back-End Task Table		
JVM Thread ID	Link	Thread ID of the JVM.
Thread Class Name	String	Value depends on the task. • For commands, the exact command class name is shown. • For timers, the timer task's callback class name is shown.
Server time (ms)	String	Time it took for the server to complete the request.

Table 4-24: Tasks Page (Continued)

Name	Type	Description
JVM Time (ms)	String	Time the thread took as reported by the JVM.
Resource	String	Resource for the action.
Status	String	Status of the task.
User	String	User that launched the request.
ID	String	Back-end thread ID.
Request Table		
JVM Thread ID	Link	Thread ID of the JVM.
ID	String	Request ID.
Session ID	String	Session ID.
Server time (ms)	String	Time it took for the server to complete the request
UI Bytes	String	Size of the OutPutStream from servlet reponse object.
Resource	String	Resource for the action.
Status	String	Status of the request
User	String	User that launched this request.
Thread Class Name	String	Class name of the thread.
Thread Stack Trace		
JVM Thread ID	String	Thread ID of the JVM.
Stack Trace	String	Stack trace of the thread.

Sessions Page

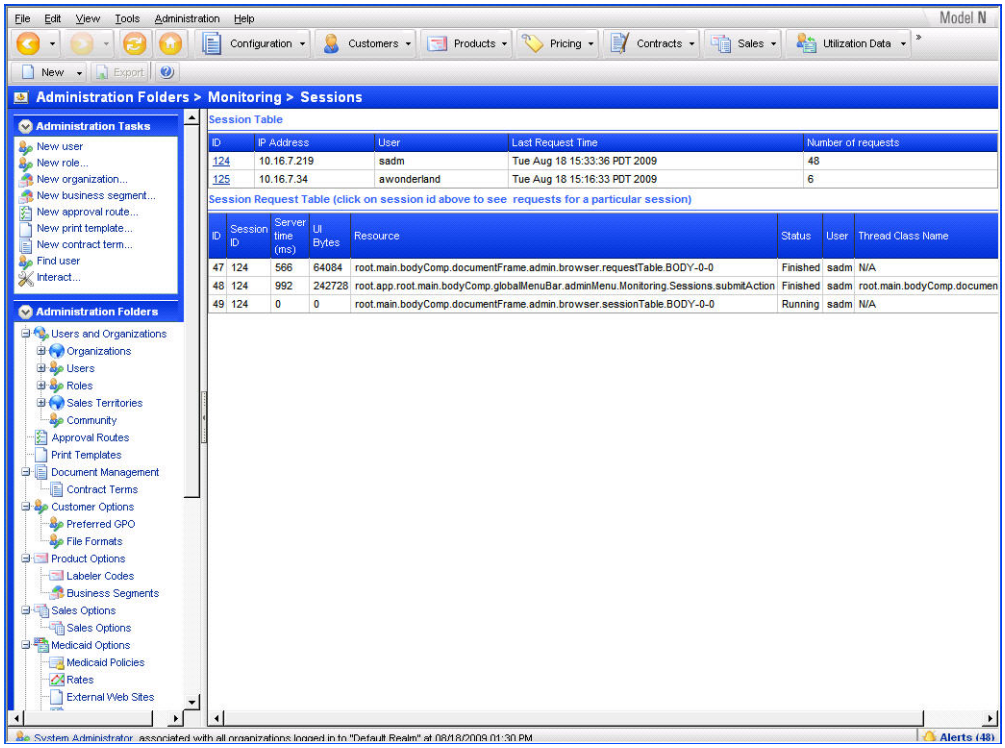
The Sessions page displays a list of all currently active sessions. You can see a list of the sessions requests by clicking a session.



4.15 Sessions Page

Navigation Path: **Administration > Monitoring > Sessions**

Figure 4-20: Sessions Page



The Sessions page is used to track user sessions in the application.

Table 4-25: Sessions Page

Name	Type	Description
Session Table		
ID	Link	Provides details about this session in the Session Request Table section.
IP Address	String	The IP address for the user's client machine.
User	String	The member name for the user session.
Last Request Time	String	The time when the last request was received on this session.
Number of requests	String	The number of requests made in this session.
Session Request Table		

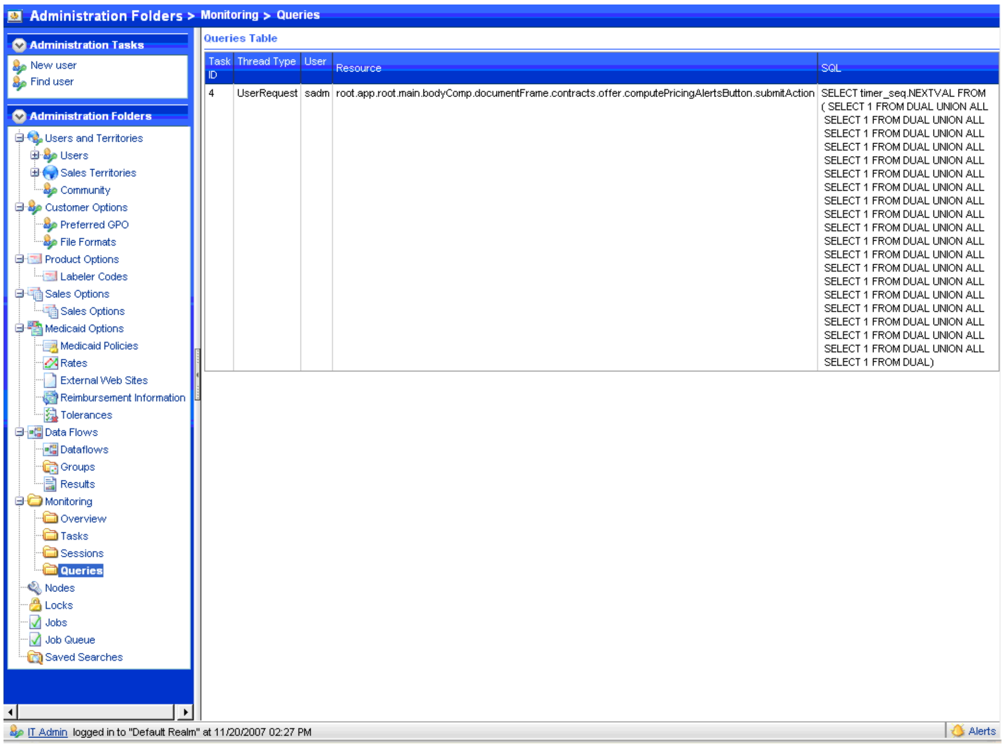
Table 4-25: Sessions Page (Continued)

Name	Type	Description
ID	String	ID of the session request.
Session ID	String	Session ID.
Server time (ms)	String	Time it took for the server to complete the request.
UI Bytes	String	Size of the OutPutStream from servlet reponse object.
Resource	String	Resource for the action.
Status	String	Status of the request
User	String	User that launched this request.
Thread Class Name	String	Class name of the thread.

Queries Page

The Queries page displays a list of all currently running queries.

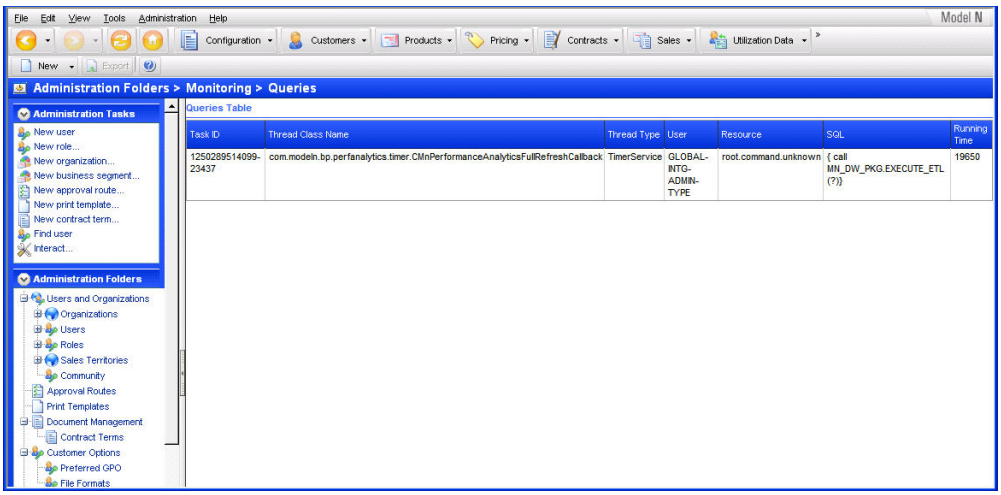
Figure 4-21: Queries Page



4.16 Queries Page

Navigation Path: **Administration > Monitoring > Queries**

Figure 4-22: Queries Page



The Queries page

Table 4-26: Queries Page

Name	Type	Description
Queries Table		
Task ID	String	
Thread Class Name	String	
Thread Type	String	
User	String	
Resource	String	
SQL	String	
Running Time	String	

JSON Servlet

In addition to the `CMnApplicationServlet`, as of the 5.3 release, the `CMnJsonServlet` has been added.

Code 4-16: CMnJsonServlet

```
public class CMnJsonServlet extends HttpServlet {
    protected void doGet(HttpServletRequest req, HttpServletResponse
resp) throws java.io.IOException{
        PrintWriter writer = resp.getWriter();
        String[] chunks = req.getPathInfo().split("/");
        String controller = chunks[1];
        String action = chunks[2];
        try {
            Class c = Class.forName(controller);
            Method m = c.getMethod(action);
            ((JSONObject)m.invoke((Object)null,
(Object[])null)).write(writer);
        } **Catch clauses **
    }
}
```

4.16.1 Oracle Database Monitoring

This section provides information on monitoring the Oracle database.

Port Monitoring

Port	Protocol	Purpose

1521	TCP/IP
	The entry in the `/etc/services` file should look as follows:
	`listener 1521/tcp`
	`# Oracle Listener`
		TNS Listener Port

Process Monitoring

Name	Purpose
`ora_pmon_<SID>`	Process Monitor Process - PMON
`ora_dbw0_<SID>`	Database Writer Process - DBWn (one or more)
`ora_lgwr_<SID>`	Log Writer Process - LGWR
`ora_ckpt_<SID>`	Checkpoint Process - CKPT
`ora_smon_<SID>`	System Monitor Process - SMON

Name	Purpose
ora_reco_<SID>	Recover Process - RECO
ora_cjq0_<SID>*	Job queue monitoring process - CJQ0
ora_p00n_<SID>*	Job queue processes (one or more) dynamically spawned by CJQ0 if required.
tnslsnr	TNS Listener

Note: Processes in the preceding table that are marked with an asterisk (*) do not need to be monitored because they are only temporary, but be aware that Oracle may create such processes as needed.

Log Monitoring

Type	Location
TNS	$ORACLE_HOME/network/log/listener.log
Instance Log	$ORACLE_BASE/admin/$ORACLE_SID/bdump/ alert_<SID>.log
Trace Files	$ORACLE_BASE/admin/$ORACLE_SID/bdump/*.trc $ORACLE_BASE/admin/$ORACLE_SID/udump/*.trc

Note: The TNS server log is not automatically rotated. Operations may need to create a process for truncating this log on a regular basis. The trace files are generated by various Oracle processes and may contain status information or error messages if Oracle encountered an error condition. None of the logs generated by Oracle are deleted automatically, so the number of trace files and the size of the instance log will increase over time.

4.17 Cognos ReportNet

This section explains how to monitor the Cognos ReportNet server and includes information on memory consumption, CPU utilization, log messages, and dispatchers and services.

For information on monitoring Cognos ReportNet see Server Administration, in the *Cognos Administration and Security Guide*.

4.17.1 Memory Consumption and CPU Utilization

Use tools provided by your operating system to monitor Cognos' memory consumption and CPU utilization. There is no separate Cognos tool to monitor memory consumption or CPU utilization. Cognos processes consist of `BIBusTKServerManagers` and Java processes.

4.17.2 Log Messages

You can configure Cognos logs to be written to the logs directory (`c8_location/logs`) or to the database. You can use Cognos Connection to set different log levels for each Cognos service.

For information on Cognos logging, refer to the Log Messages section of the *Cognos Administration and Security Guide*. For detailed information on how to use the logging console application, see the *Cognos Logging Console and Monitoring* document.

Important: High logging levels can impact performance and therefore setting every service to the highest logging level is not recommended.

4.17.3 Dispatchers and Services

Cognos consists of a set of services and dispatchers that dispatch requests to those services. Cognos Connection provides a user interface to view the status of the services and dispatchers including:

- the current and maximum number of processes
- the average latency
- seconds per request
- requests per minute
- the update interval for the preceding information

For more information on Cognos dispatcher and service monitoring, refer to the Dispatchers and Services section in Part 2, Server Administration, of the *Cognos Administration and Security Guide*.

4.17.4 Session Management

Cognos maintains a session for each logged in user. By default, users who are idle for more than thirty minutes are logged out. The timeout property, `SessionInactivityTimeout`, is in the Cognos configuration screen, but the Model N override is in the `rsvpproperties.xml` file in the Cognos configuration directory.

The default Model N configuration uses a shared Cognos connection for all Model N users. To keep this connection alive, Model N periodically pings the shared connection. To set the frequency with which Model N pings the shared connection, set the Model N property, `com.modeln.ReportingSvc.ConnectionTestPeriod`.

Note: The value of `com.modeln.ReportingSvc.ConnectionTestPeriod`, which is in minutes, must be less than the value of `SessionInactivityTimeout`, which is in seconds.

Note that the Model N property is in minutes and the Cognos property is in seconds.

4.17.5 Database Connections

Cognos uses two sets of database connections. It connects to the Cognos Content Store where it stores its reporting content and also connects to the Model N application as a reporting data source.

You must configure the number of connections used by Cognos so that the total number of connections used by all applications (Model N, Cognos, any other) does not exceed the number of connections allowed by your Oracle database configuration.

If you use the Model N database as your content store, even as a different user or schema, the Cognos Content Store connections are also included in the total number of connections.

4.17.5.1 Configuring Cognos Data Source Connection Pools

Cognos maintains a pool of connections to a data source.

By default, Cognos ships with sample versions of `CQEConfig.xml` and `rsvpproperties.xml` files in the Cognos configuration directory and Model N provides default `CQEConfig.xml` and `rsvpproperties.xml` files to configure and limit Cognos connections to the Model N database (data source).

To configure the number of database connections in the pool, set the value of the `PoolSize` parameter in the `CQEConfig.xml` file.

To configure the timeout for when unused connections are placed back in the pool by setting the value of the Timeout parameter found in the `CQEConfig.xml` file.

After setting the values for the properties in the `CQEConfig.xml` and `rsvpproperties.xml` files, restart Cognos.

For more instructions on how to configure Cognos connections to the data sources, see the Manage Data Source Connections section of the *Cognos Administration and Security Guide*.

4.17.5.2 Configuring Cognos Content Store Connections

Cognos creates connections to the Cognos Content Store. These are controlled through a different set of Cognos properties.

You must configure these connections because, by default, the connections are unlimited.

For more information on how to configure Cognos connections to the Cognos Content Store, see the Managing Database Connection Pool Settings for Content Manager section in the *Cognos Administration and Security Guide*.

4.17.6 Monitoring the Reporting Server

4.17.6.1 CPU Utilization Trend

The CPU utilization for the reporting servers should be measured over time. The utilization is the load on the CPU. The common measurements are either the CPU load or the % utilization.

Running reports within Cognos seem to be CPU intensive on the reporting server as well as having an impact on the database through the underlying queries. By reviewing the reporting server CPU load, we can see if the reports need to be tuned or the reporting server configuration may need to be modified (such as the maximum number of concurrently running reports).

Tools

Most monitoring tools provide some CPU monitoring capability. At Model N, we use an open source tool called internally to monitor our servers. For more information, see http://www.zabbix.com.

4.17.6.2 Top Execution Times by Report

Cognos reports may be intensive for both the Cognos Report Server as well as the database. We have seen the Cognos Report Server take significant CPU cycles to build a report. We have also seen some report queries be complex and database intensive, potentially adding to the contention in the database. The report execution times should provide an indicator of potential problems from reports.

5 Monitoring Pages

The following UI elements apply to the various pages available from the **Administration > Monitoring** menu.

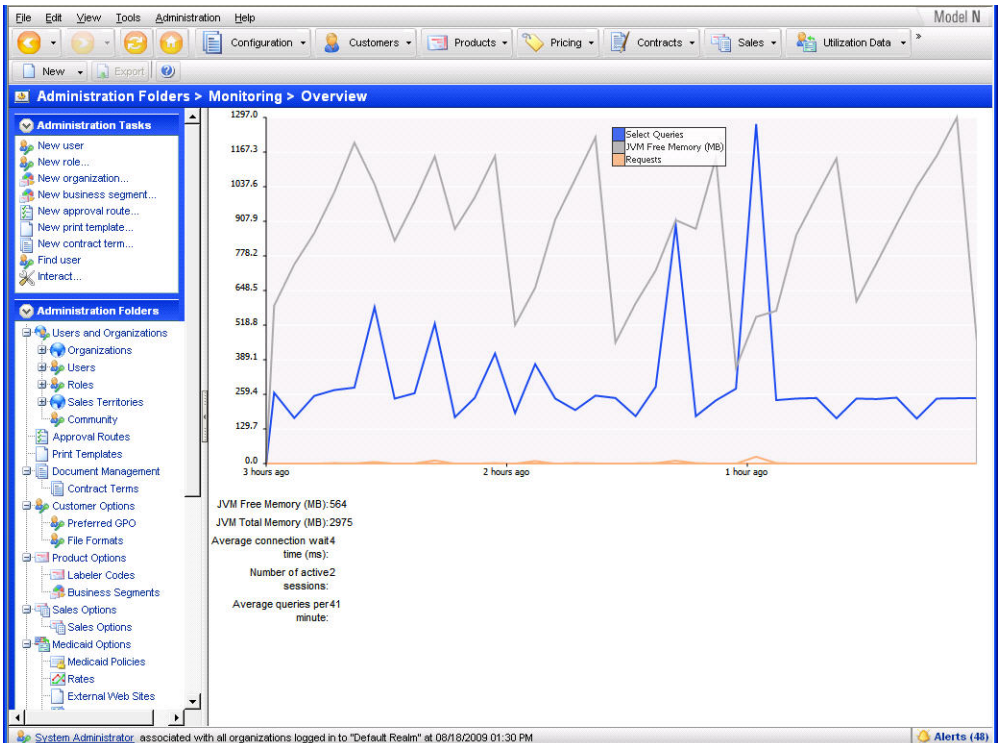
For information on the toolbar and navigation folders on these pages, see "Administration Toolbar and Navigation Panel" of the *Application Administration Guide* for more information.

Note: The application pages are documented as displayed in edit mode in the Pharmaceutical version of the Model N system. The field types and availability may differ depending on Medical Device installations, access settings, customizations, and whether you're in Read-Only or Edit mode or creating a new object.

5.1 Overview Page

Navigation Path: **Administration > Monitoring > Overview**

Figure 5-1: Overview Page



The Overview page displays a measurement of memory usage by the Model N application.

Table 5-1: Overview Page

Name	Type	Description
[graph]	Graph	The measurement over time of: • Select queries • JVM free memory • Requests
Details		
JVM Free Memory	String	The amount of JVM free memory in MB.
JVM Total Memory	String	The amount of JVM total memory in MB.
Average connection wait time	String	The average connection wait time in milliseconds.

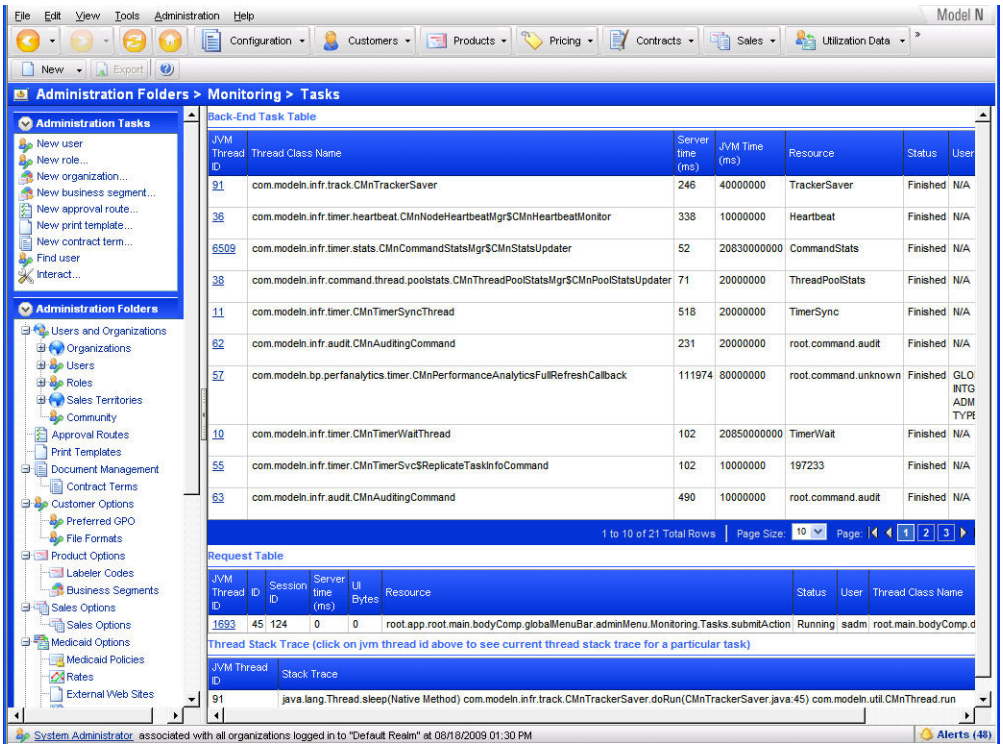
Table 5-1: Overview Page (Continued)

Name	Type	Description
Number of active sessions	String	The number of active sessions.
Average queries per minute	String	The average number of queries per minute.

5.2 Tasks Page

Navigation Path: **Administration > Monitoring > Tasks**

Figure 5-2: Tasks Page



The Tasks page tracks both backend threads, such as commands and timers, and user request threads.

Table 5-2: Tasks Page

Name	Type	Description
Back-End Task Table		

Table 5-2: Tasks Page (Continued)

Name	Type	Description
JVM Thread ID	Link	Thread ID of the JVM.
Thread Class Name	String	Value depends on the task. • For commands, the exact command class name is shown. • For timers, the timer task's callback class name is shown.
Server time (ms)	String	Time it took for the server to complete the request.
JVM Time (ms)	String	Time the thread took as reported by the JVM.
Resource	String	Resource for the action.
Status	String	Status of the task.
User	String	User that launched the request.
ID	String	Back-end thread ID.
Request Table		
JVM Thread ID	Link	Thread ID of the JVM.
ID	String	Request ID.
Session ID	String	Session ID.
Server time (ms)	String	Time it took for the server to complete the request
UI Bytes	String	Size of the OutPutStream from servlet reponse object.
Resource	String	Resource for the action.
Status	String	Status of the request
User	String	User that launched this request.
Thread Class Name	String	Class name of the thread.
Thread Stack Trace		
JVM Thread ID	String	Thread ID of the JVM.

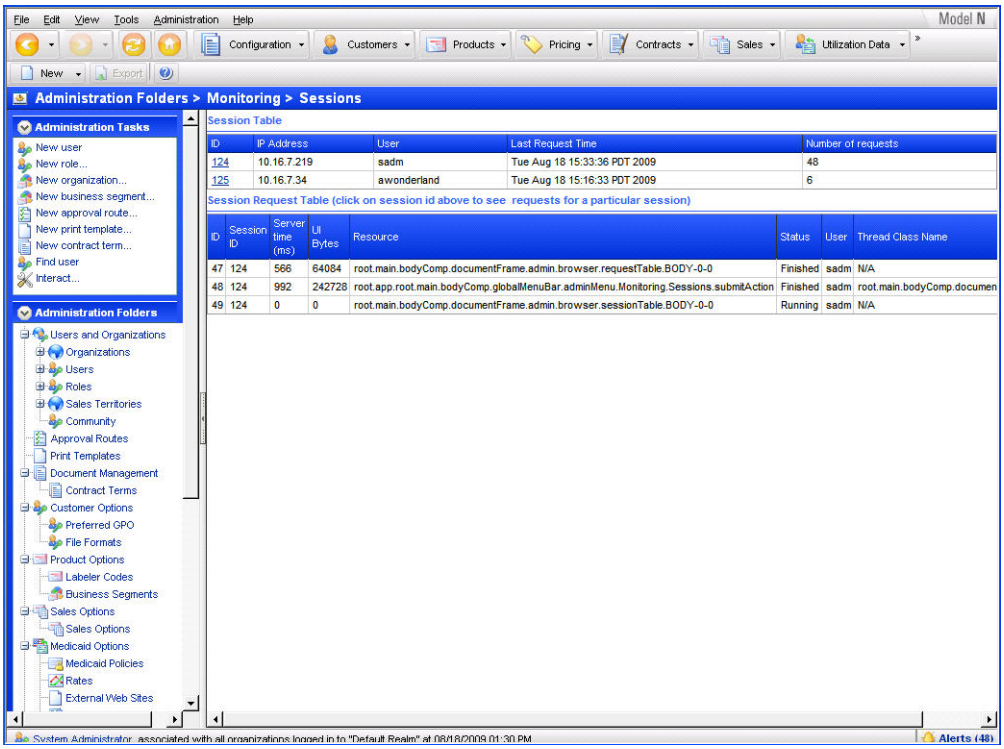
Table 5-2: Tasks Page (Continued)

Name	Type	Description
Stack Trace	String	Stack trace of the thread.

5.3 Sessions Page

Navigation Path: **Administration > Monitoring > Sessions**

Figure 5-3: Sessions Page



The Sessions page is used to track user sessions in the application.

Table 5-3: Sessions Page

Name	Type	Description
Session Table		
ID	Link	Provides details about this session in the Session Request Table section.
IP Address	String	The IP address for the user's client machine.

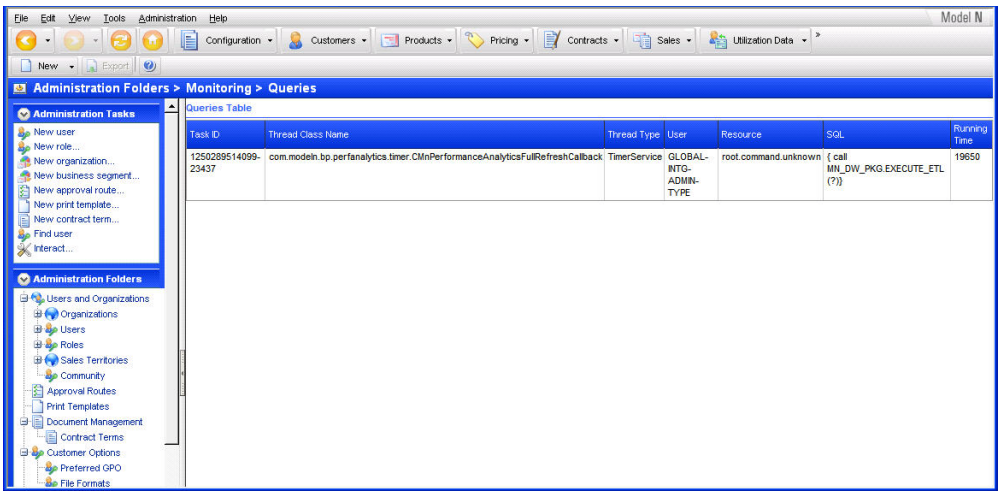
Table 5-3: Sessions Page (Continued)

Name	Type	Description
User	String	The member name for the user session.
Last Request Time	String	The time when the last request was received on this session.
Number of requests	String	The number of requests made in this session.
Session Request Table		
ID	String	ID of the session request.
Session ID	String	Session ID.
Server time (ms)	String	Time it took for the server to complete the request.
UI Bytes	String	Size of the OutPutStream from servlet reponse object.
Resource	String	Resource for the action.
Status	String	Status of the request
User	String	User that launched this request.
Thread Class Name	String	Class name of the thread.

5.4 Queries Page

Navigation Path: **Administration > Monitoring > Queries**

Figure 5-4: Queries Page



The Queries page

Table 5-4: Queries Page

Name	Type	Description
Queries Table		
Task ID	String	
Thread Class Name	String	
Thread Type	String	
User	String	
Resource	String	
SQL	String	
Running Time	String	

6 Management Console Pages

The following UI elements apply to the various pages available from the **Administration > Management** menu.

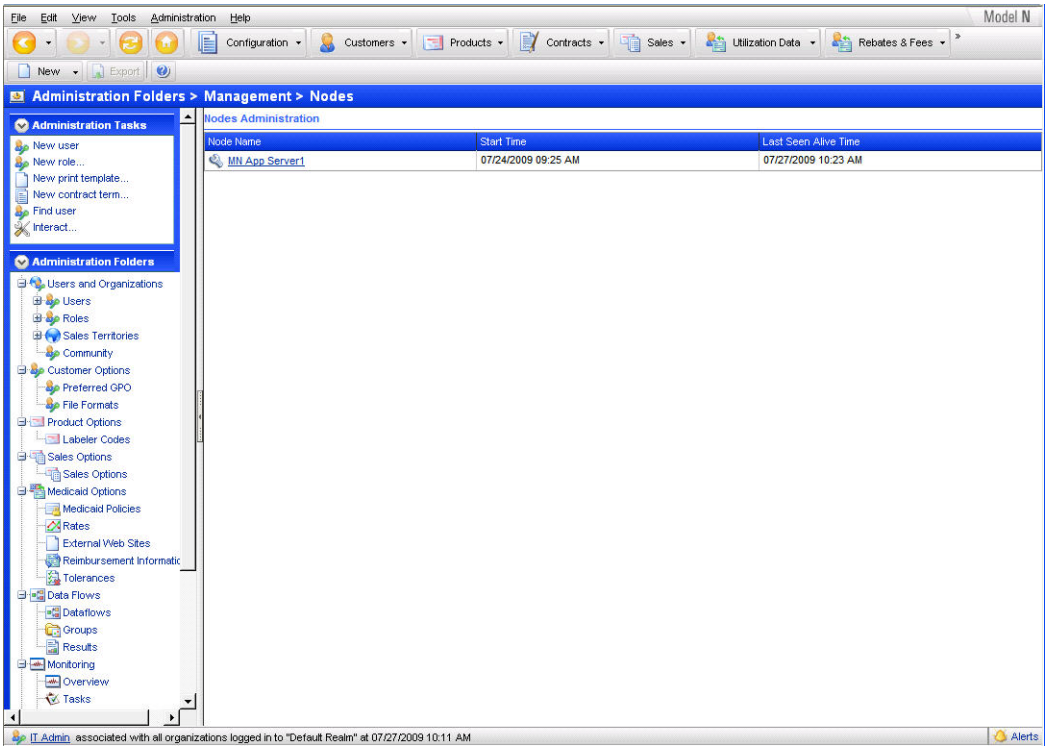
For information on the toolbar and navigation folders on these pages, see "Administration Toolbar and Navigation Panel" of the *Application Administration Guide* for more information.

Note: The application pages are documented as displayed in edit mode in the Pharmaceutical version of the Model N system. The field types and availability may differ depending on Medical Device installations, access settings, customizations, and whether you're in Read-Only or Edit mode or creating a new object.

6.1 Nodes Page

Navigation Path: **Administration > Management > Nodes**

Figure 6-1: Nodes Page



The Nodes page shows all nodes that are either part of the cluster at that instant or have been in the past. To manage an individual node, click the name of the node in the list.

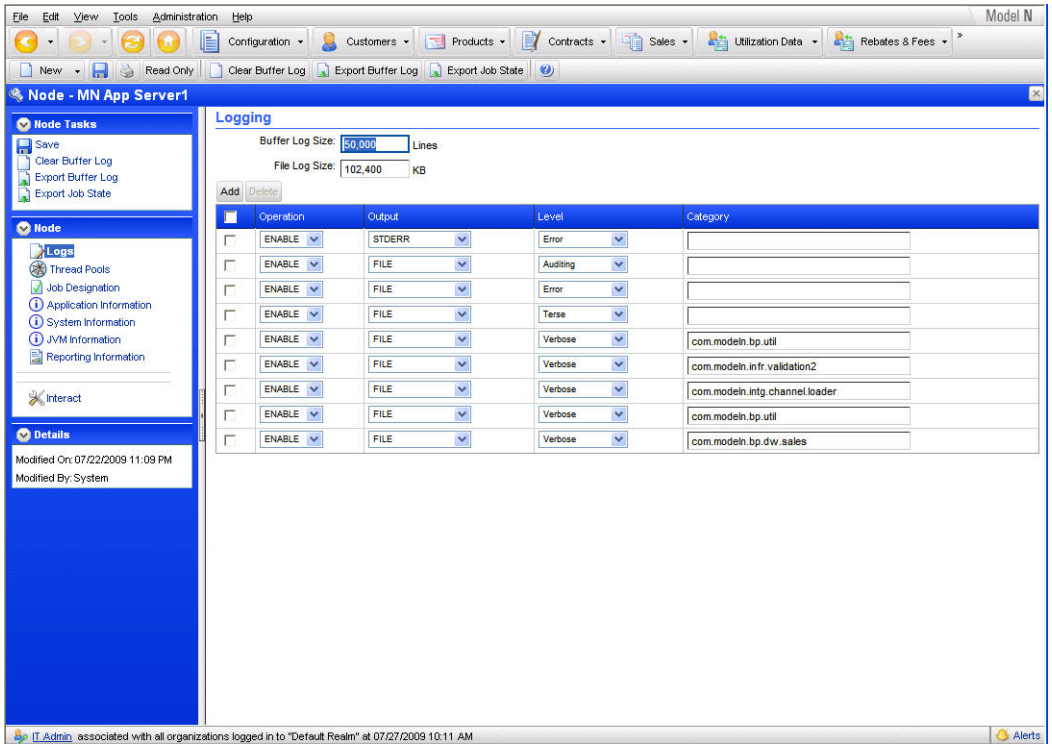
Table 6-1: Nodes Page

Name	Type	Description
Nodes Administration		
Node Name	Link	Name of an individual node in a Model N clustered environment.
Start Time	Date	The time when the node was started.
Last Seen Alive Time	Date	The last time when the node was known to be functional within the cluster.

6.2 Logging Page

Navigation Path: **Administration > Management > Nodes > Node Name** link

Figure 6-2: Logging Page



The Logging page lets you view and make changes to the log settings on a selected node.

Table 6-2: Logging Page

Name	Type	Description
Buffer Log Size	Text box	Lets you specify the number of lines present in the in-memory buffer in which the log is captured.
File Log Size	Text box	Used to identify the maximum size of the log file if any output setting is selected to be of type *File*. The logging sub-system creates new files to capture log output once the maximum file size had been reached.
Add	Button	Lets you add a log setting.
Delete	Button	Lets you delete a selected log setting.
{check box}	Check box	Enable the Delete button to delete the selected item.

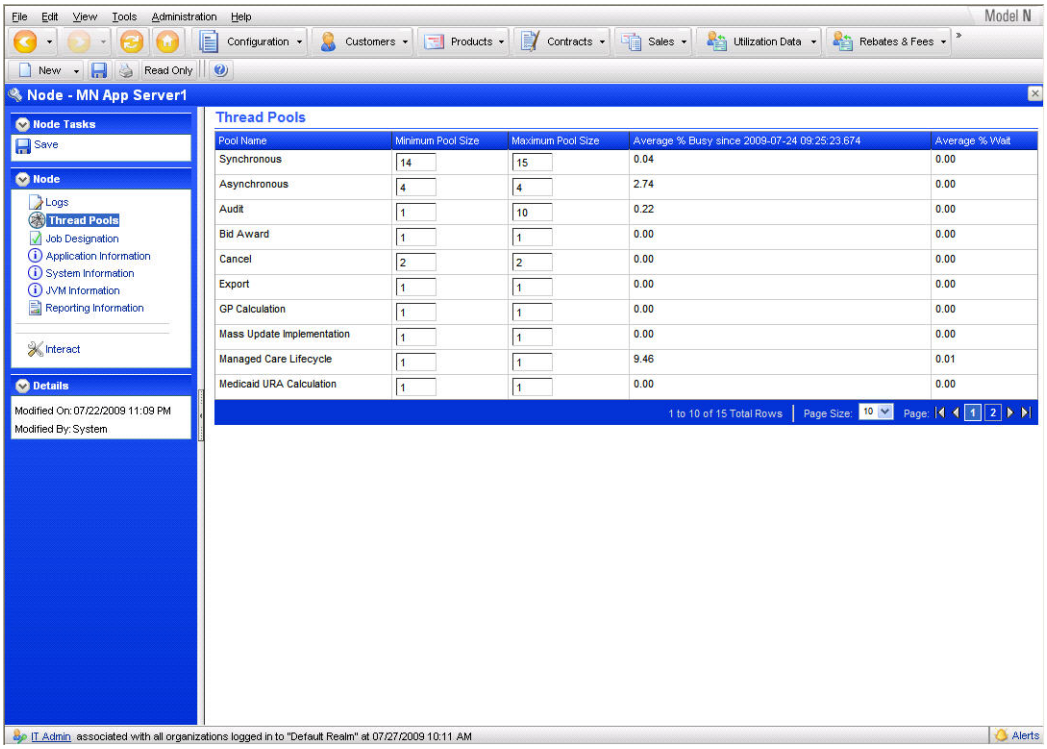
Table 6-2: Logging Page (Continued)

Name	Type	Description
Operation	Drop-down list	Sets whether to enable or disable the log setting.
Output	Drop-down list	Sets the output type for the log.
Level	Drop-down list	Sets the logging level.
Category	Text box	The class name you want to log.

6.3 Thread Pools Page

Navigation Path: **Administration > Management > Nodes > Node Name** link **> Thread Pools**

Figure 6-3: Thread Pools Page



The Thread Pools page lets you define the minimum and maximum pool size.

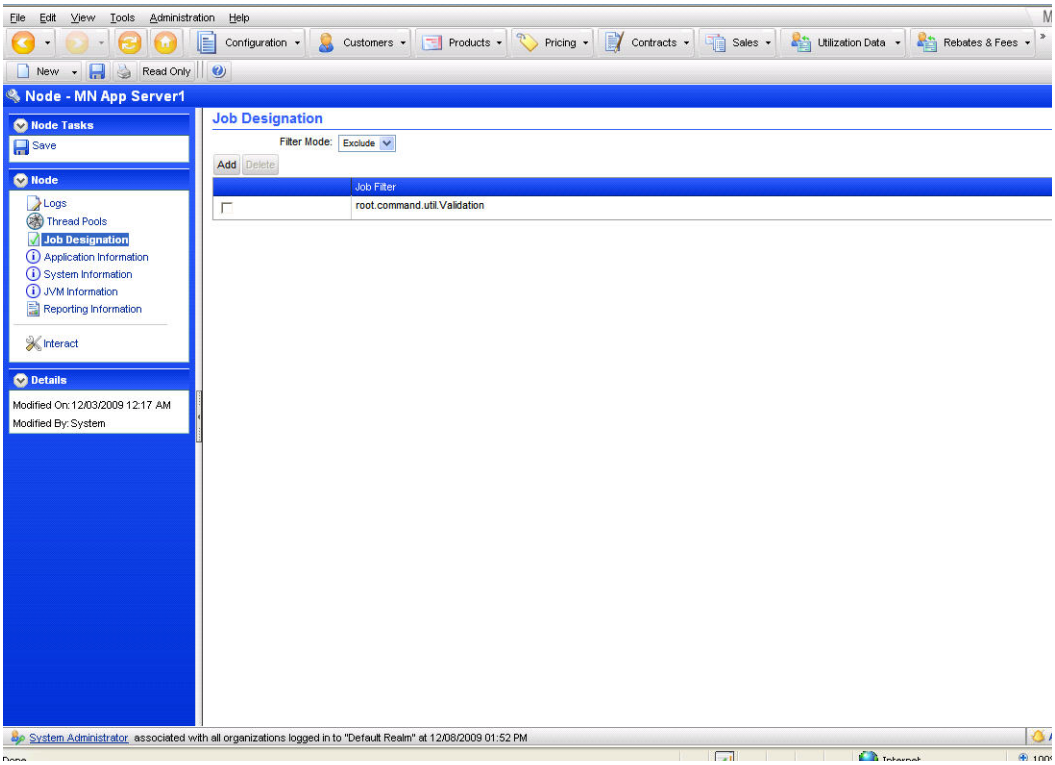
Table 6-3: Thread Pools Page

Name	Type	Description
Pool Name	String	Name of the thread pool.
Minimum Pool Size	Text box	Minimum thread pool size.
Maximum Pool Size	Text box	Maximum thread pool size.
Average % Busy Since	String	Average percent use of the thread pool.
Average % Wait	String	Average percent wait of the thread pool.

6.4 Job Designation Page

Navigation Path: **Administration > Management > Nodes > Node Name** link **> Job Designation**

Figure 6-4: Job Designation Page



The Job Designation page lets you direct background jobs to specific nodes.

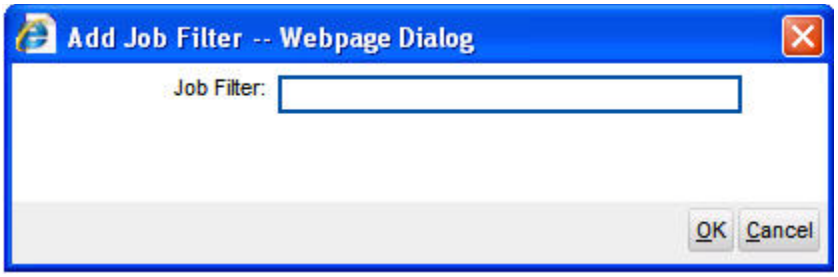
Table 6-4: Job Designation Page

Name	Type	Description
Filter Mode	Drop-down list	Lets you filter the jobs displayed by whether they are excluded from execution or not.
Add	Button	Lets you add job filters.
Delete	Button	Lets you delete job filters.
{check box}	Check box	Select the check box to enable the Delete button.
Job Filter	String	Lists the job filters found.

6.4.1 Add Job Filter Dialog Box

Navigation Path: **Administration > Management > Nodes > Node Name** link **> Job Designation > Add**

Figure 6-5: Add Job Filter Dialog Box



The Add Job Filter dialog box lets you specify resources to use as job filters.

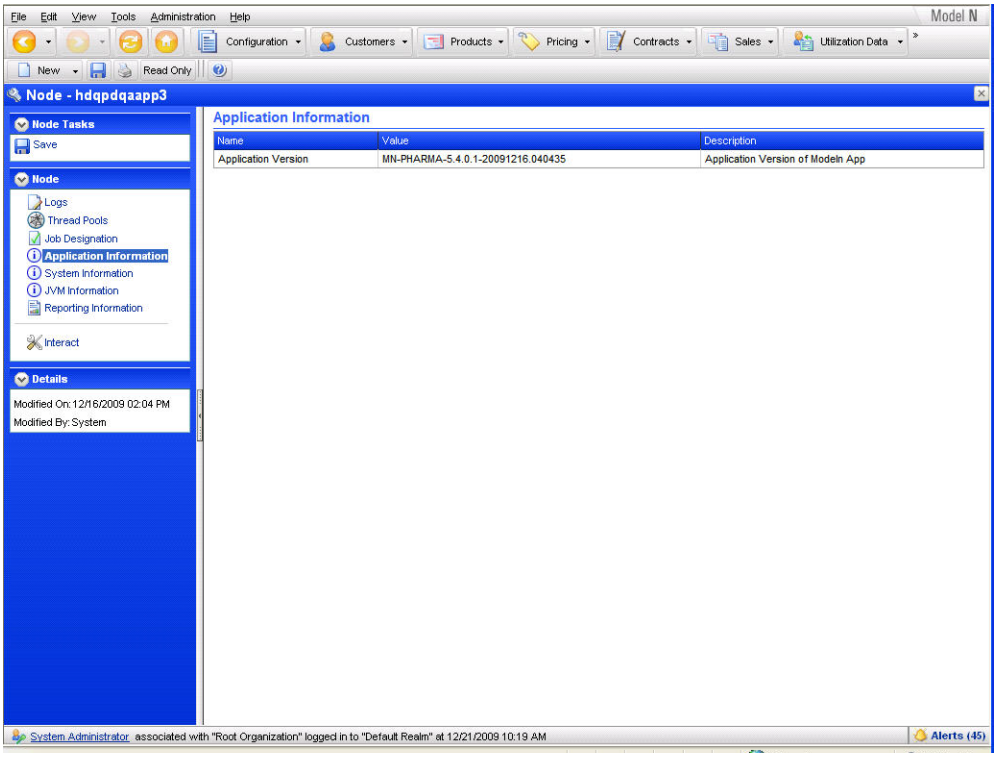
Table 6-5: Add Job Filter Dialog Box

Name	Type	Description
Job Filter	Text box	The command type to add to as a job filter. A valid entry would be resource names starting with `root.command`, such as: `root.command.util.Validation`.

6.5 Application Information Page

Navigation Path: **Administration > Management > Nodes > Node Name** link **> Application Information**

Figure 6-6: Application Information Page

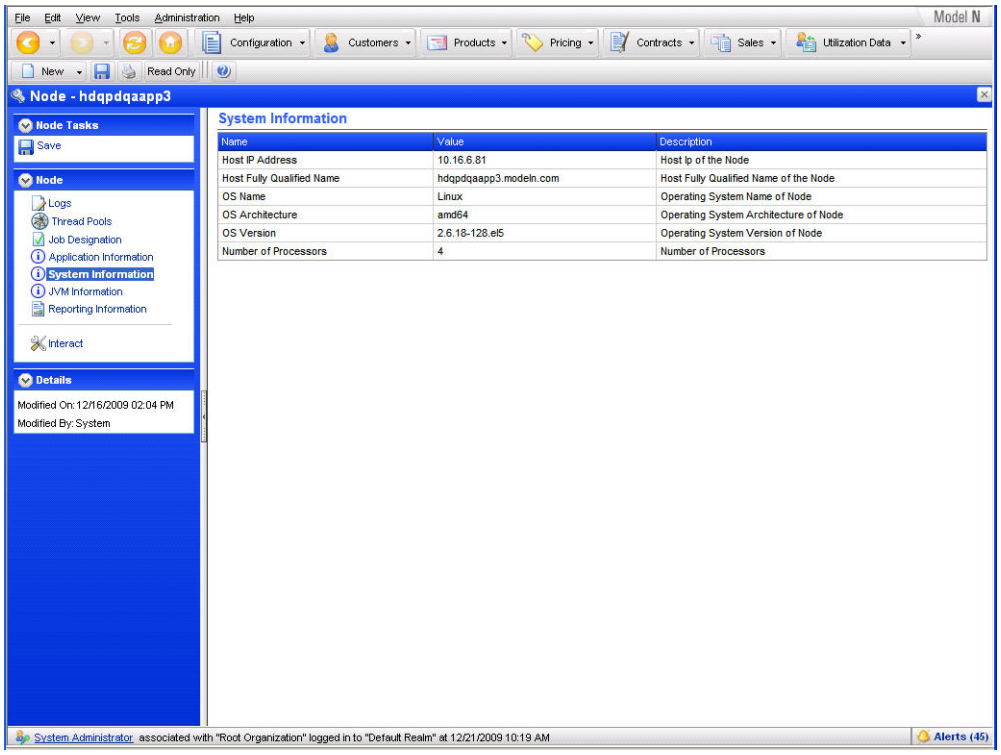


The Application Information Page provides information about Model N application build version.

6.6 System Information Page

Navigation Path: **Administration > Management > Nodes > Node Name** link **> System Information**

Figure 6-7: System Information Page



The System Information page provides information about the system hosting your Model N application.

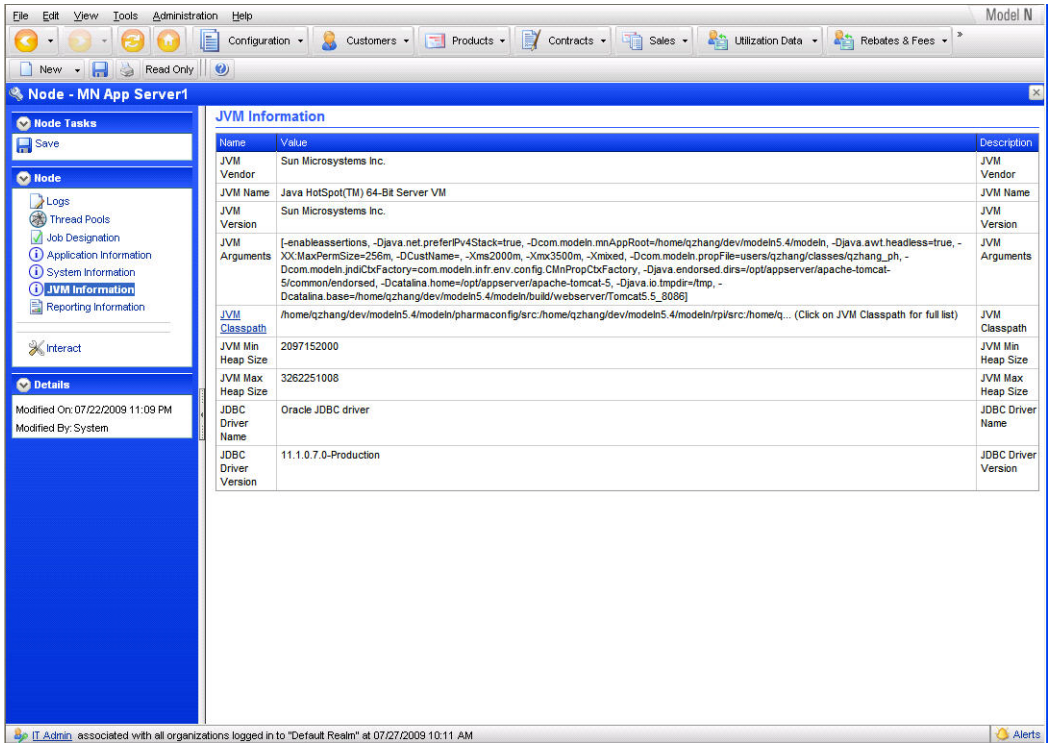
Table 6-6: System Information Page

Name	Type	Description
Name	String	The types of information presented.
Value	String	The values for the specified information types.
Description	String	A description of the information presented.

6.7 JVM Information Page

Navigation Path: **Administration > Management > Nodes > Node Name** link **> JVM Information**

Figure 6-8: JVM Information Page

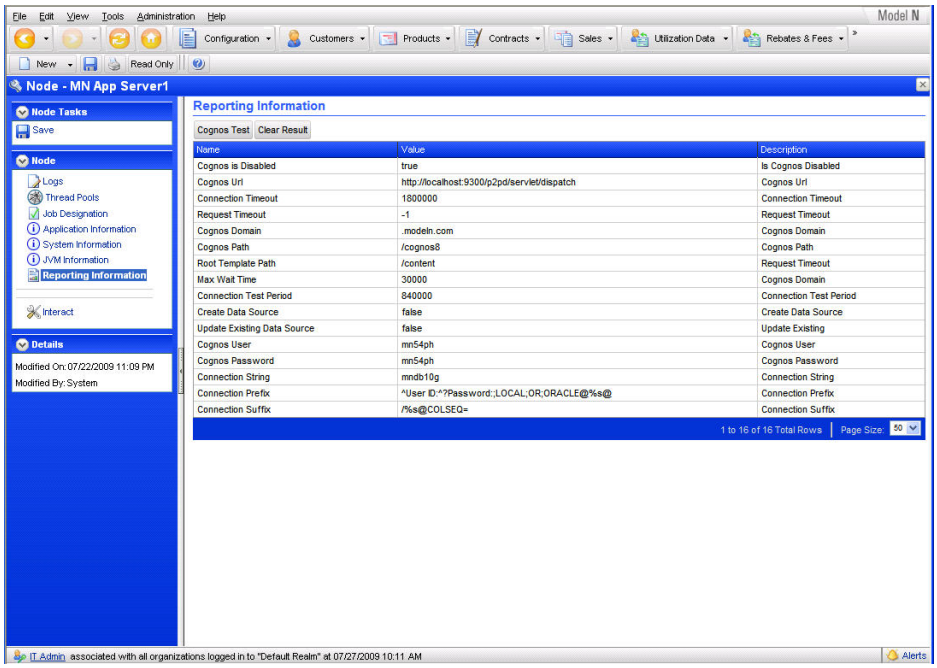


This page contains information about the Java Virtual Machine (JVM) on which this node is running. The JVM Classpath link opens the JVM Class Path dialog box, which contains the full list of JVM classpaths.

6.8 Reporting Information Page

Navigation Path: **Administration > Management > Nodes > Node Name** link **> Reporting Information**

Figure 6-9: Reporting Information Page



This page contains information about the Cognos report integration and its current status.

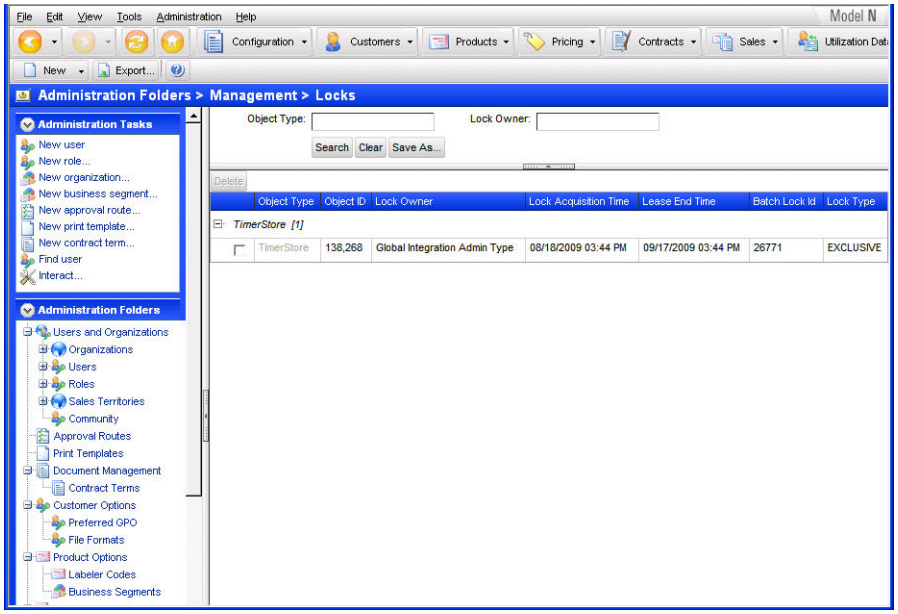
Table 6-7: Reporting Information Page

Name	Type	Description
Cognos Test	Button	Tests the connection to the Cognos reporting server.
Clear Result	Button	Clears the result of the Cognos Test.

6.9 Locks Page

Navigation Path: **Administration > Management > Locks**

Figure 6-10: Locks Page



Lock Management provides visibility into persistent locks being held by the application.

Table 6-8: Locks Page

Name	Type	Description
Search		
Object Type	Text box	Identity name of the locked object.
Lock Owner	Text box	Owner of the lock.
Search	Button	Searches for locks with the specified criteria.
Clear	Button	Clears your search criteria, letting you perform another search.
Save As	Button	Saves your search criteria so you many repeat the search.
Results		
[check box]	Check box	Lets you select a persistent lock and enables the Delete button.

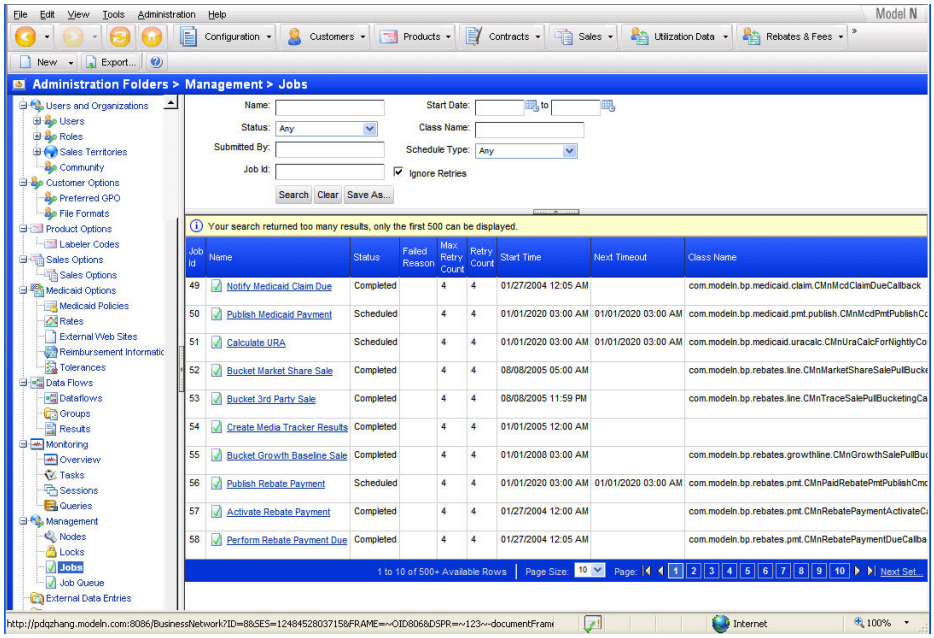
Table 6-8: Locks Page (Continued)

Name	Type	Description
Object Type	String	Identity name of the locked object.
Object ID	String	Identifies the identifying information of the locked object.
Lock Owner	String	Owner of the lock.
Lock Acquisition Time	String	Time when the lock was acquired on the object.
Lease End Time	String	Time when the object is set to lose the lease on the lock. Locks are valid for pre-allocated time intervals known as lease time.
Batch Lock ID	String	Batch Lock ID associated with the lock. In some cases the application might obtain locks for a collection of objects as a batch operation. The operation to obtain a batch lock on a collection of objects associates a Batch Lock ID with all of the locks involved.
Lock Type	String	Identifies the lock as either a shared or an exclusive lock. Locks assigned to objects can either be shared or exclusive. Objects in the Model N application can belong to locking hierarchies. Obtaining a lock on an object results in an exclusive lock being granted to the object being locked and a shared lock being granted with the locking operation to all of the parent objects in the hierarchy of the object being locked.
Delete	Button	Lets you delete the selected locks.

6.10 Jobs Page

Navigation Path: **Administration > Management > Jobs**

Figure 6-11: Job Page



The Jobs page lets you view jobs in the Model N framework.

Table 6-9: Job Page

Name	Type	Description
Search		
Name	String	Name of the job being executed.

Table 6-9: Job Page (Continued)

Name	Type	Description
Status	Drop-down list	Helps to identify the status of the job. Possible values are: • **Cancelled:** jobs that had begun execution but were cancelled by the user. • **Completed:** jobs that have completed execution. • **Failed:** jobs that have failed execution upon termination of the job. • **On Hold:** jobs that are placed on hold using the Job Queue section of the Management Console. • **Pending Cancel:** jobs that are in the process of being cancelled. • **Recovered and Rescheduled:** jobs that have been automatically recovered and rescheduled for execution. • **Recovered and Terminated:** jobs that have automatically recovered and terminated. Some jobs are automatically terminated because the job is structured to be that way. • **Running:** jobs that are currently executing. • **Scheduled:** jobs that are scheduled to run in the future. • **Waiting:** jobs that should be executing as per their scheduling time, but are waiting because of lack of availability of threads in the cluster.
Submitted By	String	User who submitted the job.
Job Id	String	The ID number of the job.
Start Date	Date selector	Date range over which the job was or is scheduled to execute.
Class Name	String	Name of the class implementing the job.
Schedule Type	Drop-down list	Search by the various schedule types allowed in the application. Possible vales are: • **Single:** scheduled for a single run. • **Periodic:** scheduled to run after a periodic interval, specified in milliseconds. • **Daily:** scheduled to be run on a daily basis. • **Monthly:** scheduled to be run once a month.

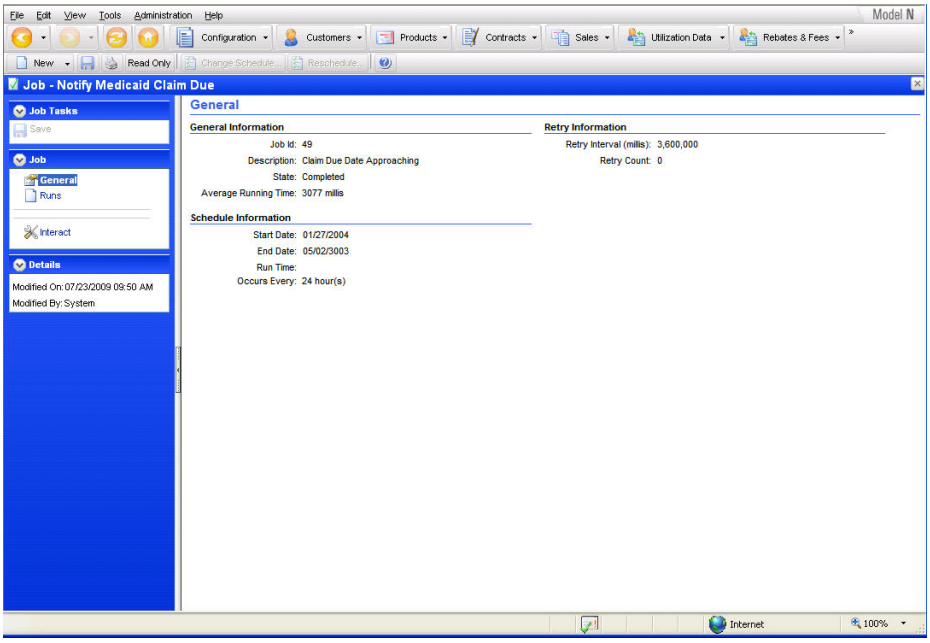
Table 6-9: Job Page (Continued)

Name	Type	Description
Ignore Retries	Check box	Select this option to ignore jobs that are retries of other jobs.
Search	Button	Searches for jobs with the specified criteria.
Clear	Button	Clears your search criteria, letting you perform another search.
Save As	Button	Saves your search criteria so you many repeat the search.
Results		
Job Id	String	The ID number of the job.
Name	Link	Name of the job being executed.
Status	String	Status of the job.
Failed Reason	String	Reason the job failed.
Max Retry Count	String	Maximum retries the job can attempt.
Retry Count	String	Current retry count for the job.
Start Time	String	When the job was last started.
Next Timeout	String	When the next timeout is set to occur.
Class Name	String	Class name for the job.
Schedule Type	String	The schedule type for the job.
Schedule Interval	Integer	The interval of when the job can be scheduled.
Submitted By	String	User that submitted the job
Node Name	String	Name of the server node from which the job applies.

6.11 Job General Page

Navigation Path: **Administration > Management > Jobs > Name** link

Figure 6-12: Job General Page

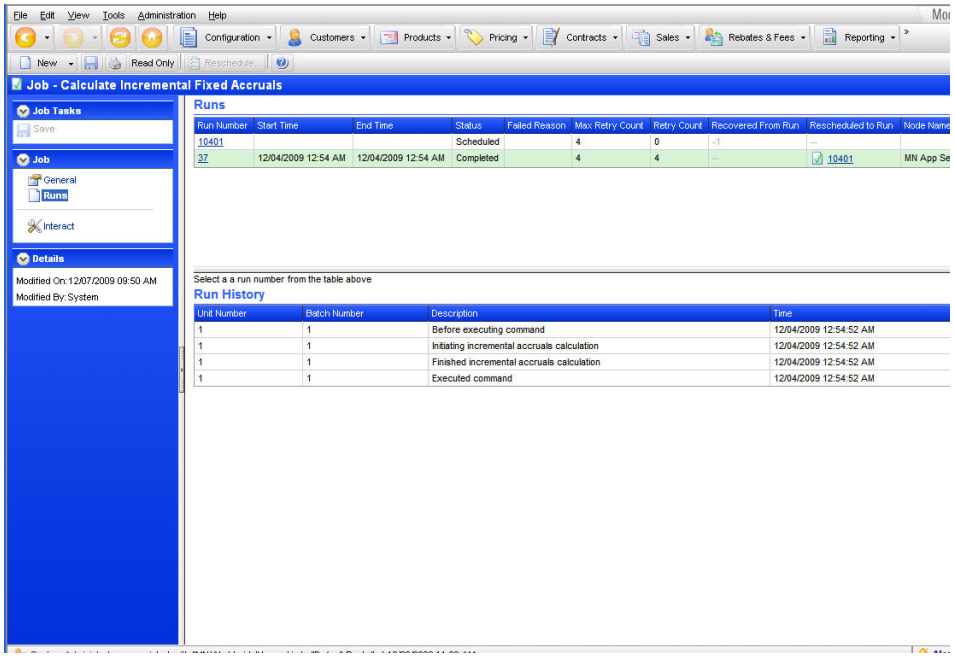


The Job General page provides general details on the job such as a short description of the job, its current state, and its average running time as observed by the application. It also provides details on the schedule setup for the job which includes details about the next start time of the job, the time when the schedule ends, the time when the job is scheduled to start, and its recurrence frequency.

6.12 Job Runs Page

Navigation Path: **Administration > Management > Jobs > Name** link **> Runs**

Figure 6-13: Job Runs Page



The Runs page provides details of historical runs associated with the job.

Table 6-10: Job Runs Page

Name	Type	Description
Runs		
Run Number	Link	Unique system identifier for the run.
Start Time	String	When the job was started.
End Time	String	When the job completed.
Status	String	Status of the job run.
Failed Reason	String	Reason the job failed, if applicable.
Max Retry Count	String	Maximum retries the job can attempt.
Retry Count	String	Current retry count for the job.

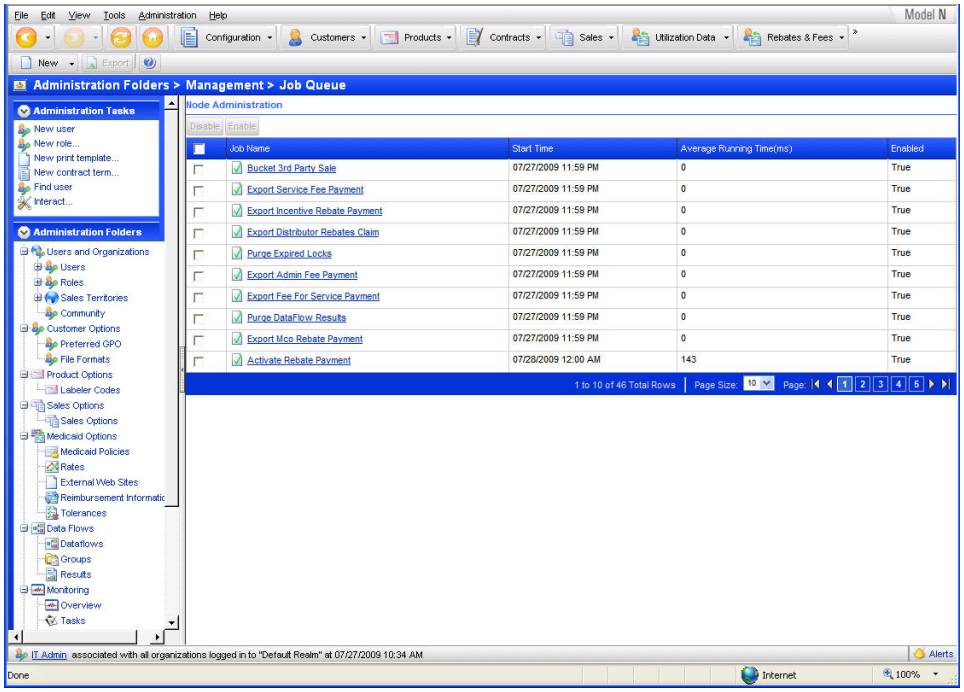
Table 6-10: Job Runs Page (Continued)

Name	Type	Description
Recovered From Run	String	Identifies the job from which this job might have recovered. If the job did not recover from any prior run, the field is left blank.
Rescheduled To Run	Link	Run number for a job rescheduled to run.
Node Name	String	Name of the node on which the job is to run.
Run History		
Unit Number	String	
Batch Number	String	
Description	String	
Time	String	

6.13 Jobs Queue Page

Navigation Path: **Administration > Management > Jobs Queue**

Figure 6-14: Jobs Queue Page



The Jobs Queue page shows the active job queue present in the cluster.

Table 6-11: Jobs Queue Page

Name	Type	Description
[check box]	Check box	Lets you select one or more jobs , then enables the **Enable** and **Disable** buttons.
Job Name	Link	Name of the scheduled job.
Start Time	String	Time when the job is scheduled to run next.
Average Running Time (ms)	String	Represents the average time taken by this job in the past runs.
Enabled	String	Indicates whether a particular job is currently enabled or not.
Disable	Button	Lets you disable the selected job.
Enable	Button	Lets you enable the selected job.

7 Configuration Console

This chapter provides detailed information on the Configuration Console. The Configuration Console is a user interface with which you can manage all of the configurations in the Model N application.

The major parts that comprise the Configuration Console are described in the following sections:

- Configuration Console
- Model N Application Bootstrap Process
- Configuration Set Importer and Exporter

7.1 Configuration Console

This chapter describes the Configuration Console and how to use it, including maintaining revisions of configurations, changing and activating configurations, and reverting to a previously-existing configuration. This chapter also provides information on the tools you can use to migrate your configurations from one release to another.

The following sections provide information about the Configuration Console:

- Managing Configurations
- Configurations History
- Configurations and Configuration Values

7.1.1 Managing Configurations

A *configuration set,* or simply a *configuration,* is a logical grouping entity for grouping configuration values. One or more configuration sets can exist in the system, but only one configuration set can be active at a given time.

Configurations can be managed through the Configuration Console user interface by IT Admin users.

You cannot edit configurations in an active console. To make any changes to an active console, you can edit any a configuration set provided the it is unlocked. The moment you modify the values, the status of the configuration set flips to *Currently Active - Needs Restart* which indicates to the user that application server needs to be restarted for changes to take effect. The system supports reverting to a previously-active configuration console.

Out-of-the-box, the product ships with a product configuration set called the Initial Configuration, which is read-only through the Configuration Console user interface. You can only edit it through back-end tasks such as bootstrap, importers, or data flows while loading configurations from the product's application modules.

You can manage the configuration set as follows:

- Through the Configuration Console, which is the user interface IT Admin users use to log into the system and amend the set.
- When configurations are loaded into the system using back-end tasks such as bootstrap, importers, or data flows, the system manages the sets according to well-defined rules.

IT Admin users log into the Configuration Console, search for sets and activate one of them.

7.1.2 Configurations History

The system maintains a history of when a console was created, who created it, when it was activated, and when it was deactivated. You can view the history through the Configuration Console (**Configuration > Configurations > Configurations History**).

7.1.3 Configurations and Configuration Values

A configuration within Model N defines:

- the property or the configuration name.
- the data type of its value.
- when it should take effect upon a change in value and its scope.

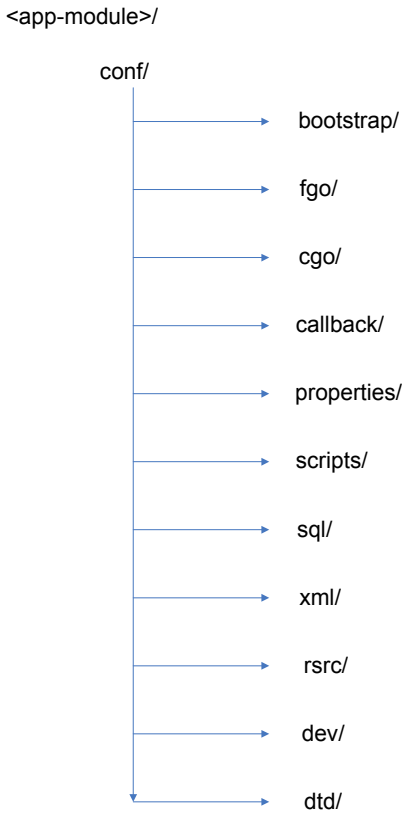
A configuration can have one or more values in the system.

7.2 Directory Structure for the Configurations

Every module or feature in the Model N application, has a `/conf` directory. All properties and configurations are defined in this directory and adhere to the following directory structure.

Distinctions exist between the properties defined in property files and application switches in the current content files.

To accommodate these differences, the following directory structure contains the configurations that replace the properties that existed prior to the 5.3 release. The following structure uses `sales/` as an example of the `<app-module>`.



The following table describes the folders in the directory.

Table 7-1: Configurations Directory Structure

Directory	Description
Bootstrap Directory	Contains the configurations for properties, FGO, and CGO definitions that are essential for bringing up the application server. These configurations are not loaded into the database.
FGO, CGO, and Callback Directories	These directories contain the XML specifications for FGO, CGO, and callbacks, respectively. These configurations are loaded into the database.

Table 7-1: Configurations Directory Structure (Continued)

Directory	Description
Properties Directory	Contains the properties that could not be classified as particular configurations.
Scripts Directory	Contains property files that define external queries that are not associated with specific FGO/CGOs.
SQL Directory	Contains stored procedure definitions and other SQL files.
XML Directory	Contains XML files used in the application.
Rsrc Directory	Contains resource definitions such as those used in display strings.
Dev Directory	Contains properties that are defined for use in a development environment.
Dtd Directory	Contains `.dtd` files.

7.3 Application Feature

An *application feature* is a feature as specified in packaging. You can override a configuration defined in the application feature using a configuration in the pharma module. In this case, configurations from both the application and pharma modules are loaded into the console in their respective modules. When the server starts up and configurations are loaded, the loading engine along with packaging can use the pharma configuration over the application configuration.

Each configurable value belongs to only one feature. The custom feature is a special feature, so that any changes you make to a configuration through the Configuration Console user interface gets automatically put into the custom features and not into one of the existing features.

7.4 Configuration Categories

Category indicates a classification of properties/configurations that exists in the system.

You can view and edit the following types of configurations using the Configuration Console.

- Business Objects (FGOs)
- Services
- Application Switches
- Local Services

Also available within the Configuration Console are:

- Configurations History
- Enumerations
- Global Application Switches

7.4.1 Business Objects (FGOs)

The Business Objects or FGOs are configured in the Model N system using FGO configuration .xml files defined under `<app-module>/conf/fgo/`.

These configuration files are loaded into the configuration repository during bootstrap and are available for you to edit through the Configuration Console user interface.

7.4.1.1 Queries

Using the Configuration Console, you can edit the queries associated with FGOs. The following query attributes are editable:

- The SQL Query associated with the Query
- The Query Arguments
- Result Limit

7.4.1.2 Validations

You can enable or disable validation plug-ins for an FGO using the Configuration Console. If an object is organization aware, the validations associated with that object may be applied to specific organizations.

How to Include or Exclude an Organization in a Validation

Note: Validations can only be modified in configurations that are not active.

1. Go to the Business Objects page: **Configuration > Configurations > Configuration Name** link **> Business Objects**
2. Select the object to modify.
3. Select the Validations tab.
4. In the validation for which you want to specify an organization, click the Chooser button in the **Include In** or **Exclude From** column. The Select an Organization dialog box opens. If you include an organization, then all child (or descendant) organizations are also included. The same is also true for exclude. This allows the ability to easily manage validations across organizations. If you explicitly include and exclude the same organization, the include takes precedence.

   Note: **Include In** and **Exclude From** columns with a value of `NA` indicate the object is not organization-aware, so the validations cannot be designated to specific organizations and apply globablly.

5. Select the organization and click **Add**.

6. Select **Enabled** and click **Save**.
7. Activate the configuration and restart the server for the changes to take effect.

7.4.1.3 Importers and Exporters

You can use the Configuration Console to view the Importer and Exporter information associated with an FGO.

7.4.1.4 FGO Extensions

You can use the Configuration Console to view the extensions registered for an FGO.

7.4.2 Services

The following types of services are available in configurations:

- Callbacks
- Notifications
- Visibility

7.4.2.1 Callbacks

You can use the Configuration Console to enable and disable the registered callbacks in the system. The Configuration Console importer registers callbacks in the system.

7.4.2.2 Notifications

You can use the Configuration Console user interface to view the notifications registered in the Model N application by each module. You cannot edit notifications from the user interface.

The existing `TaskRouterImporter` is used to import notifications into the system.

7.4.2.3 Visibility

You can use the Configuration Console to view visibility plug-ins registered in the Model N application by each module. You cannot edit visibility plug-ins through the user interface.

The `VisibilityEntry` importer is used to import visibility plug-ins into the system.

7.4.3 Application Switches

Application switches map directly to `Configuration` and `ConfigurationValue` FGOs. Therefore, the existing classes are supported for application switches and the existing configuration XML format is used to load application switches.

7.4.3.1 Organization Aware Application Switches

Certain application switches can be set so that different organizations have different values. These switches can be found by going to the Application Switches page (**Configuration > Configurations >** Configuration Name link **> Application Switches**) and searching for switches with a `Yes` value in the Org Aware column.

7.4.4 Local Services

Local services are functional-based services. Unlike business objects, which store data, local services let you view certain tasks that the application performs such as running timers and callback events.

7.4.5 Configurations History

The Configurations History page provides insight into when configurations have been activated and deactivated.

7.4.6 Enumerations

You can use the Configuration Console to look up enumerations in the system and create runtime enumerations.

You can also edit enumerations in the system using the Configuration Console by:

- Adding new enumeration values.
- Deleting enumeration values.
- Editing the display value and display order of the existing enumerations.
- Showing static enums and bit flags. These entries get created during bootstrap (provided the enum class was listed in `com.modeln.preloadClasses property`.)
- Setting organization-specific values for organization-aware enumerations.
- Specifying locales for enumeration values.

You can load enumerations by populating `Enumerations.EnumerationReader`. By default, you must re-start the server for the enumerations to take effect. Place the enumeration files in the contents folder, according to the specific feature.

You can also use the *DynamicEnumNamespaceToXml* data flow to export out all enums in the system.

Note: The Configuration Set export does not export enumeration values.

7.4.6.1 How to Configure Organization-Aware Enumerations

1. Go to the Enumerations page: **Configuration > Enumerations**. See Enumerations Page on page 186 for more information on this page.
2. Click on the **Object Name** link of an org aware object (indicated by a *Yes* value in the **Org Aware** column) that you want to modify.
3. Either click on the enumeration **Name** link or click **Add** to create a new enumeration value. The Add dialog box opens.
4. Click the **Included In** chooser button to open the Select an Organization dialog box and add the organizations that this enumeration value applies to.
5. Click the **Excluded From** chooser button to open the Select an Organization dialog box and specify the organizations that are not associated with this value.

6. Click **Save**.

Once set up, the values for a given organization are determined by taking the union of all enumeration values associated with the given organization or any ancestor organization minus any enumeration values that are marked as not applicable to your organization or ancestor organization.

7.4.6.2 How to Create a Dynamic Enumeration at Runtime

There may be times an IT Developer will want to create and configure a new dynamic enumeration at runtime, perhaps to create a new custom attribute based on that enumeration.

1. Go to the Enumerations page: **Configuration > Enumerations**. See Enumerations Page on page 186 for more information on this page.
2. On the Configuration Tasks panel, click **Create Enumeration**. The Create Enumeration dialog box opens.
3. Enter the name for the enumeration and select Organization Aware if applicable, then click **OK**. Organization-aware enumerationss are those which allow different sets of values to be set in different organizations.
4. Supply enumeration values by clicking **Add**.

7.4.7 Global Application Switches

While the configuration set application switches require the application server to be restarted, most global application switches do not. This information is provided in the right-most 'Needs Restart' column.

7.5 Model N Application Bootstrap Process

The bootstrap process is required to create the application schema and loading configurations into the system. The bootstrap process is for a new customer installation or deployment, and not for a customer migration or upgrade from an earlier release to a later release.

The following scripts initiate a bootstrap process that reads bootstrap configurations, connects to the database, creates or updates the schema, and loads all the configurations that are persisted in the system.

- `ant -f ops.xml BootstrapPharma`
- `ant -f ops.xml BootstrapMeddev`

To run the bootstrap script, you first need to configure your environment to run ANT utilities. To see the steps on how to do this, go to the Configuring the Environment section of the Deployment chapter of the *Installation & Migration Guide*. To populate content into the database, follow the steps in the Running the Tools section of the Deployment chapter of the *Installation & Migration Guide*.

7.6 Configuration Set Importer and Exporter

In the 5.3 release, a tool was provided to upload or import configurations. This tool uses a naming scheme to assign a new name to the configuration console that it might create. One of the tool's arguments indicates whether the console being created must be activated upon completion of the import.

This tool is only available on the command line.

The ant utility lets you import custom configuration sets from a file and export them to a file. You can export any custom configuration set regardless of its current status. You cannot be export the initial configuration set.

Use the following ant targets to import and export configuration sets. The ant targets are available in the operations tool, `ops.xml`.

7.6.1 ExportConfigurationSet

This ant target exports an existing custom configuration set to a file which is later used or imported into another database.

The following two parameters are required for exporting a configuration set:

- the file name to export the configuration set

  Enter the file name, including any path, absolute or relative, of the export file to be created.

- the name of the configuration set to export

  Enter the name of the configuration set to export.

7.6.2 ImportConfigurationSet

This ant target imports a configuration set export file to a different database. The name of the configuration set must not already exist in the new database.

The following parameter is required for importing a configuration set:

- the configuration set export filename

  Enter the filename including any path, absolute or relative, of the export file.

7.7 Configuration Console Activate/Deactivate Mechanism

The 5.3 release provides a tool to activate and deactivate a configuration set as well as a tool to make a configuration set uneditable. This section describes each tool. For the steps on how to run these tools, see the Running the Tools section in the Deployment chapter of the *Installation & Migration Guide*.

7.7.1 ActivateConfigurationSet

This ant task can be used to activate a configuration set.

7.7.2 DeactivateConfigurationSet

This ant task can be used to deactivate a configuration set.

7.7.3 StopEditingCustomConfigurationSet

This ant task can be used to make a configuration set uneditable.

8 Configuration Console Pages

The following UI elements apply to the various pages available from the **Configuration** menu button.

Note: The application pages are documented as displayed in edit mode in the Pharmaceutical version of the Model N system. The field types and availability may differ depending on Medical Device installations, access settings, customizations, and whether you're in Read-Only or Edit mode or creating a new object.

8.1 Configuration Navigation Panel

The navigation panel contains links to basic tasks and documents. The links are dynamic and depend on the area in which you are working and the status of the document you are working on. Refer to the following table for a complete list of links available.

Table 8-1: Configuration Navigation Panel

Navigation Panel	Link	Description
Configuration Tasks		
	Lock/Unlock	Locks/unlocks the configuration for editing.
	Save As	Opens the Save As dialog box to let you save the selected configuration as a new configuration.
	Revert to Initial Configuration Set	Lets you set the initial configuration as the active configuration.
	Make Active	Makes the selected configuration active.
Configuration Console		
	Configurations	Opens the Configuration Page.
	Configurations History	Opens the Configurations History Page.
	Enumerations	Opens the Enumerations Page.
	Global Application Switches	Opens the Global Application Switches Page.
	Saved Searches	Opens the Saved Searches Page where you can access to the Configuration Console searches that were previously saved.
Configuration		
	General	Opens the Configuration Page general information about the selected configuration.
	Business Objects	Opens the Business Objects Page for the selected configuration.
	Services	Opens the Services Page for the selected configuration.

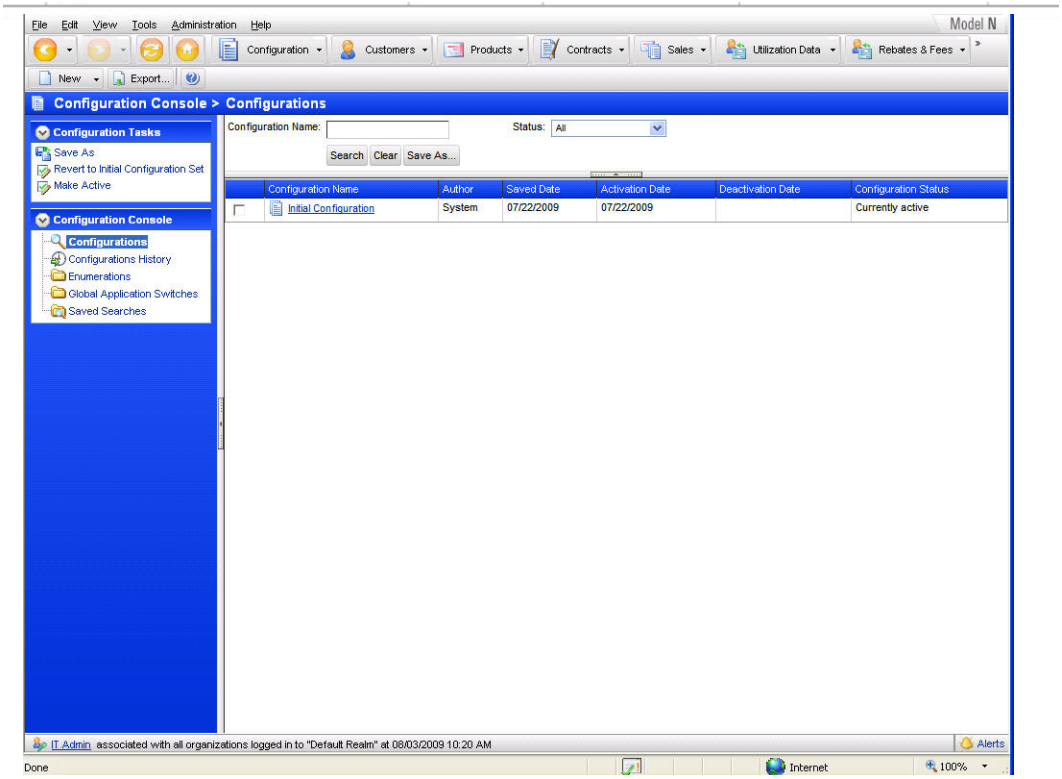
Table 8-1: Configuration Navigation Panel (Continued)

Navigation Panel	Link	Description
	Application Switches	Opens the Application Switches Page for the selected configuration.
	Local Services	Opens the Local Services Page for the selected configuration.
	Interact	Opens the Interact Page where you can access the Model N expression language.

8.2 Configurations Page

Navigation Path: **Configuration**

Figure 8-1: Configurations Page



The Configurations page provides access to the saved configuration sets for the Model N application.

Table 8-2: Configurations Page

Name	Type	Description
Search		
Configuration Name	Text box	Name of the configuration to search for.
Status	Drop-down list	Configuration status to search for. Possible options are: • All • Active on next start • Currently active • Inactive • Never activated

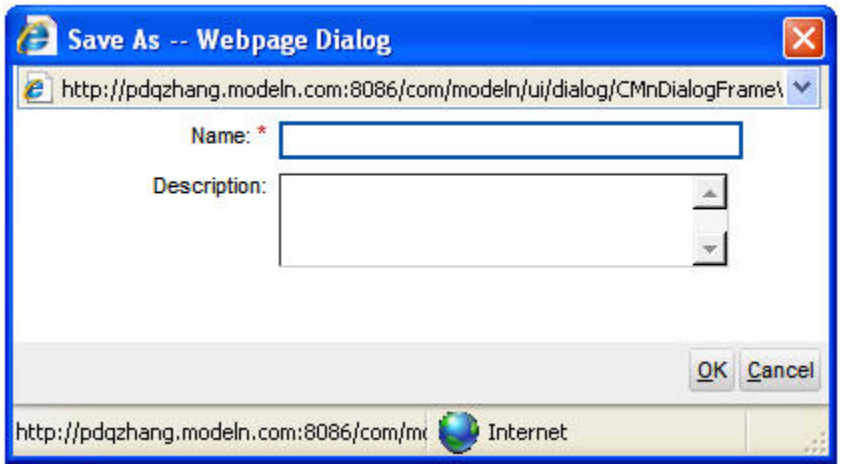
Table 8-2: Configurations Page (Continued)

Name	Type	Description
Search	Button	Searches for configurations with the specified criteria.
Clear	Button	Clears your search criteria, letting you perform another search.
Save As	Button	Saves your search criteria so you many repeat the search.
Results		
[check box]	Check box	Lets you select the configuration to perform a task.
Configuration Name	Link	Name of the saved configurations.
Author	String	User that created the configuration.
Saved Date	Date	Date the configuration was saved.
Activation Date	Date	Date the configuration was last activated.
Deactivation Date	Date	Date the configuration was deactivated.
Configuration Status	String	Current status of the configuration.

8.2.1 Save As Dialog Box

Navigation Path: **Configurations > Save As**

Figure 8-2: Save As Dialog Box



The Save As dialog box lets you save the selected configuration as a new configuration.

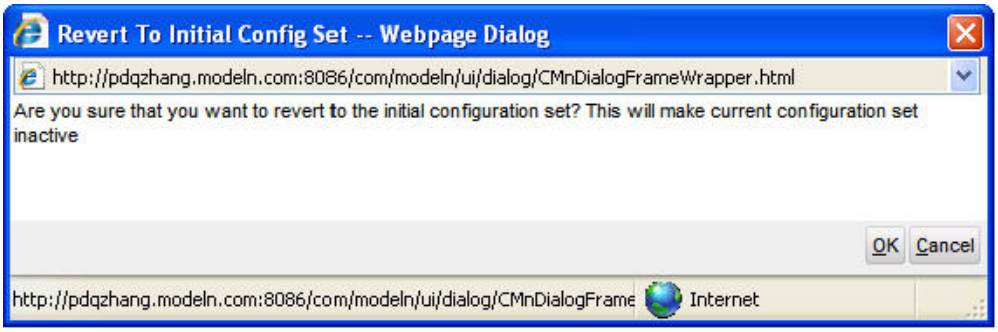
Table 8-3: Save As Dialog Box

Name	Type	Description
Name*	Text box	Enter a name for the saved configuration
Description	Text box	Enter a description for the configuration.
OK	Button	Saves the configuration and closes the dialog box.
Cancel	Button	Closes the dialog box without saving the configuration.

8.2.2 Revert to Initial Configuration Set Dialog Box

Navigation Path: **Configuration > Revert to Initial Configuration Set**

Figure 8-3: Revert to Initial Configuration Set Dialog Box



The Revert to Initial Configuration Set dialog box lets you reapply the initial configuration set to your system.

Table 8-4: Revert to Initial Configuration Set Dialog Box

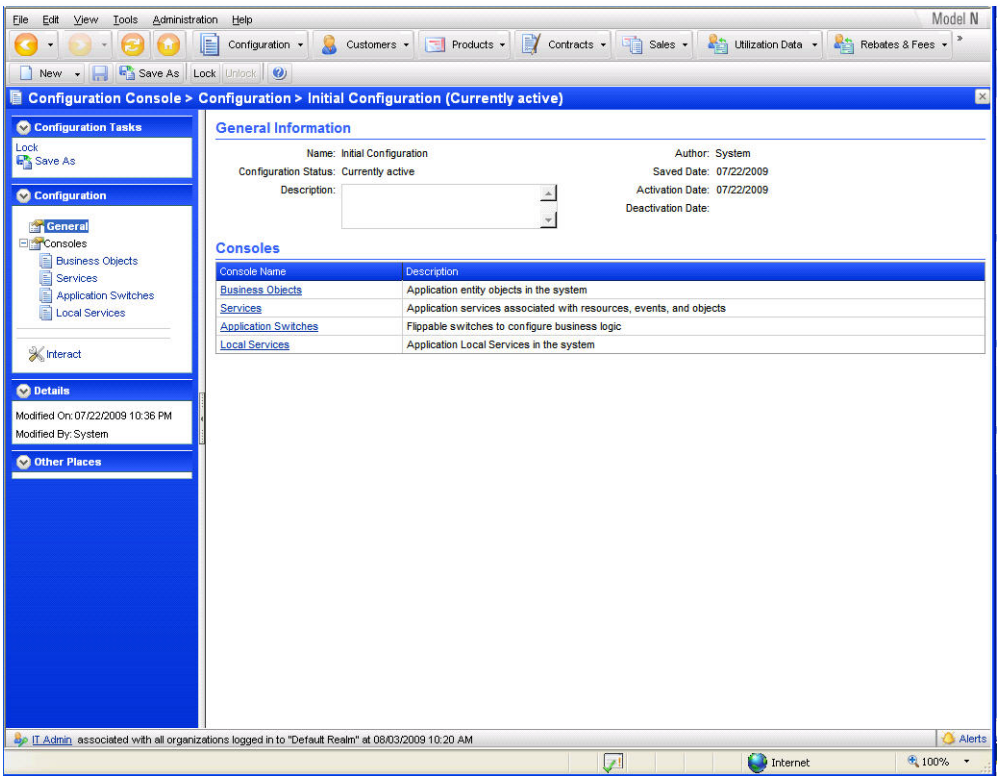
Name	Type	Description
OK	Button	Reverts the system to the initial configuration set and closes the dialog box.
Cancel	Button	Closes the dialog box without reverting to the initial configuration set.

Note: There is also a Restore to Initial Config option on the Queries and Properties tabs of **Configuration > Configuration Name** link **> Business Objects > Object Name** link where you can revert specific configurations.

8.3 Configuration Page

Navigation Path: **Configuration > Configurations >** *Configuration Name* link

Figure 8-4: Configuration Page



The Configuration page provides access to the details of the selected configuration set.

Table 8-5: Configuration Page

Name	Type	Description
General Information		
Name	String	Name of the selected configuration.
Configuration Status	String	Current status of the selected configuration.
Description	Text box	Description of the configuration.
Author	String	User that created this configuration set.
Saved Date	Date	Date this configuration set was saved
Activation Date	Date	Date this configuration was last activated.
Deactivation Date	Date	Date this configuration was deactivated.

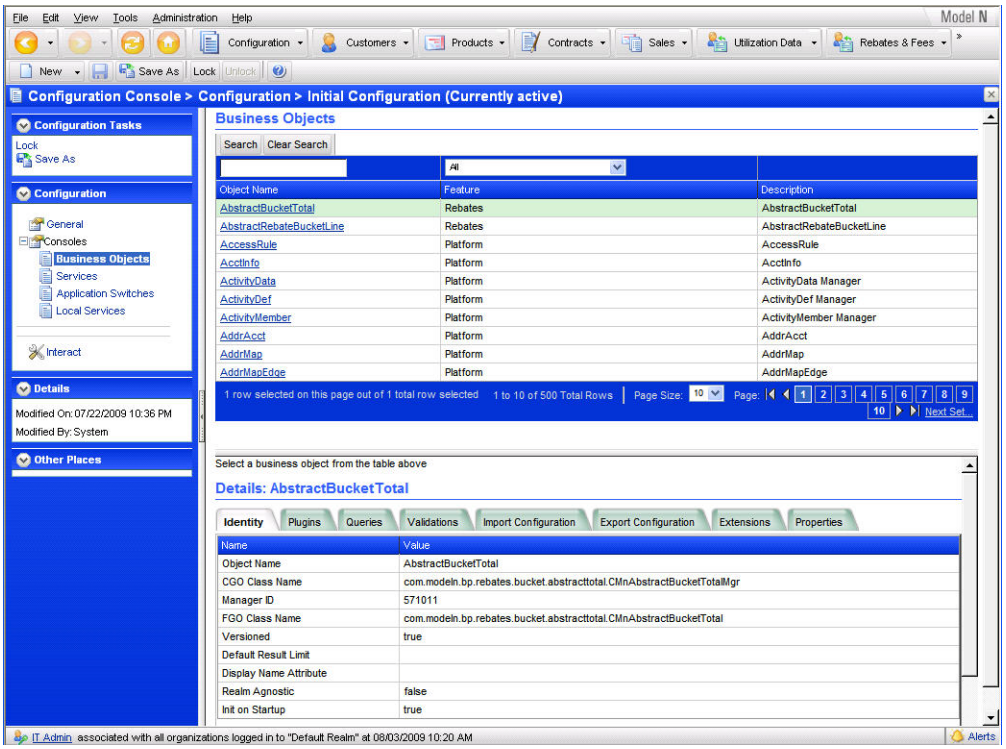
Table 8-5: Configuration Page (Continued)

Name	Type	Description
Consoles		
Console name	Link	Categories of configurations that make up the configuration set. The consoles available are: • Business Objects: takes you to the Business Objects Page • Services: takes you to the Services Page • Application Switches: takes you to the Application Switches Page • Local Services: takes you to the Local Services Page
Description	String	Description of the console.

8.4 Business Objects Page

Navigation Path: **Configuration > Configurations >** *Configuration Name* link **> Business Objects**

Figure 8-5: Business Objects Page



The Business Objects page provides access to the business objects, or FGOs, of the selected configuration set.

Table 8-6: Business Objects Page

Name	Type	Description
Search		
Search	Button	Searches for business objects with the specified criteria.
Clear Search	Button	Clears your search criteria, letting you perform another search.
Object Name	Text box	Name of a business object to search for.

Table 8-6: Business Objects Page (Continued)

Name	Type	Description
Feature	Drop-down list	Associated feature to search for. Possible options are: • All • Base Application • Base Life Science Application • Bid Awards • Compliance • Custom Features • Distributor Rebates • FSS Compliance • Government Pricing • Government Pricing - FSS Compliance • Government Pricing - Managed Care • Managed Care • Media Tracking • Medicaid • Order Quantity Discount • Pharma • Pharma Distributor Rebates • Pharma Rebates • Pharma Config • Platform • Price Master • Rebates • Revenue Planning and Intelligence • Sales • US Regulatory Pharma
Results		
Object Name	Link	Name of the business object.
Feature	String	Feature to which the business object is associated.
Description	String	Description of the business object.
Details		

Table 8-6: Business Objects Page (Continued)

Name	Type	Description
Identity	Tab	Name-Value pairs of information pertaining to the identity of the object. • **Object Name**: The unique name for the object. • **CGO Class Name:** The Java class name for the object's manager. • **Manager ID:** The unique ID for the object. • **FGO Class Name:** The Java class name for the object. • **Versioned:** Flag indicating if this object is versioned (only applicable for root objects). • **Default Result Limit:** Default row limit for queries of this object. • **Display Name Attribute:** Attribute on the object that is used in the UI. • **Realm Agnostic:** [Deprecated] Indicates if the object is realm-aware. • **Init on Startup:** Flag indicating if the manager is initialized on startup or on first access. • **Is Org Aware:** Flag indicating if the object has an organization attribute indicating what organization the object belongs to.

Table 8-6: Business Objects Page (Continued)

Name	Type	Description

Plugins	Tab	Provides the list of plug-ins that are currently registered. Object plug-ins are used to inject behavior during specific actions, such as save or remove.
		• **Plugin Name:** The unique name for the plug-in.
		• **Type:** The type of plug-in. The plug-ins are associated with specific events (such as create or save). Plug-ins may be registered to run prior to the event (as in `preSave`) or after the event (as in `postSave`). The possible types are:
		  • **postCreate:** The create event is when the object is newly created. No `preCreate` is possible, since the object has not yet been created.
		  • **preSave/postSave:** The save event is when the object is being persisted to the database.
		  • **preRemove/postRemove:** The remove event is when the object is being removed from the database.
		  • **postLoad:** The load event is when the object is being loaded from the database (such as in a `SELECT` query). No `preLoad` is possible, since the object has not yet been loaded.
		  • **preSetObj/postSetObj:** The set object event is when any value is being set on an object.
		  • **preRemoveObj/postRemoveObj:** The remove object event is when the value is being removed from an object.
		  • **preSetRelObj/postSetRelObj:** The set relationship object is when an object is being set on a 1-1 relationship.
		  • **preAddRelObj/postAddRelObj:** The add relationship object is when an object is being added to a 1-N relationship.
		  • **preRemoveRelObj/ postRemoveRelObj:** The remove relationship object is when an object is being removed from a 1-1 or 1-N relationship.
		  • **preRemoveAllRelObj/ postRemoveAllRelObj:** The remove all relationship objects is when all objects in a 1-N relationship are being removed.

Table 8-6: Business Objects Page (Continued)

Name	Type	Description
		• **Plugin Class:** The Java class name for the plug-in. This class must implement the `com.modeln.infr.pof.obj.IMnFIneGrainObjPlugin.` • **Order:** A floating point number that indicates the order of plugin execution. Plug-ins are executed in ascending order.
Queries	Tab	Provides the list of external queries that are associated with the object. • **Name:** The unique name for the query. • **Type:** Indicates what database this query is supported. Possible values are: all, ora, and db2. Model N currently only support Oracle. • **Description:** A description for the query. • **Result Limit:** A row limit for this specific query.
Validations	Tab	Provides the list of business object validations associated with the object. The object validations are executed either explicitly in the user interface by way of the **Validate** button or implicitly during lifecycle events, such as submission for approval. • **Enabled:** Indicates if the validation is enabled in the system. If disabled, this validation will never run. • **Mode:** Indicates what validation set this validation should be associated with. The sets are specific to each business object. • **Plugin Class:** The Java class name for the validation. The class must implement the `com.modeln.infr.pof.obj.IMnValidator` interface. • **Included In/Excluded From:** If the object is an organiztion-aware object (see the **Identity** tab), then FGO validations may be targeted to specific organizations. If **Included In** and **Excluded From** are not specified, then the validation is considered global and run in all organizations. Otherwise, what is specified in these columns are used to determine which organizations should execute this validator.
Import Configuration	Tab	Name-Value pairs of information pertaining to imported configurations.

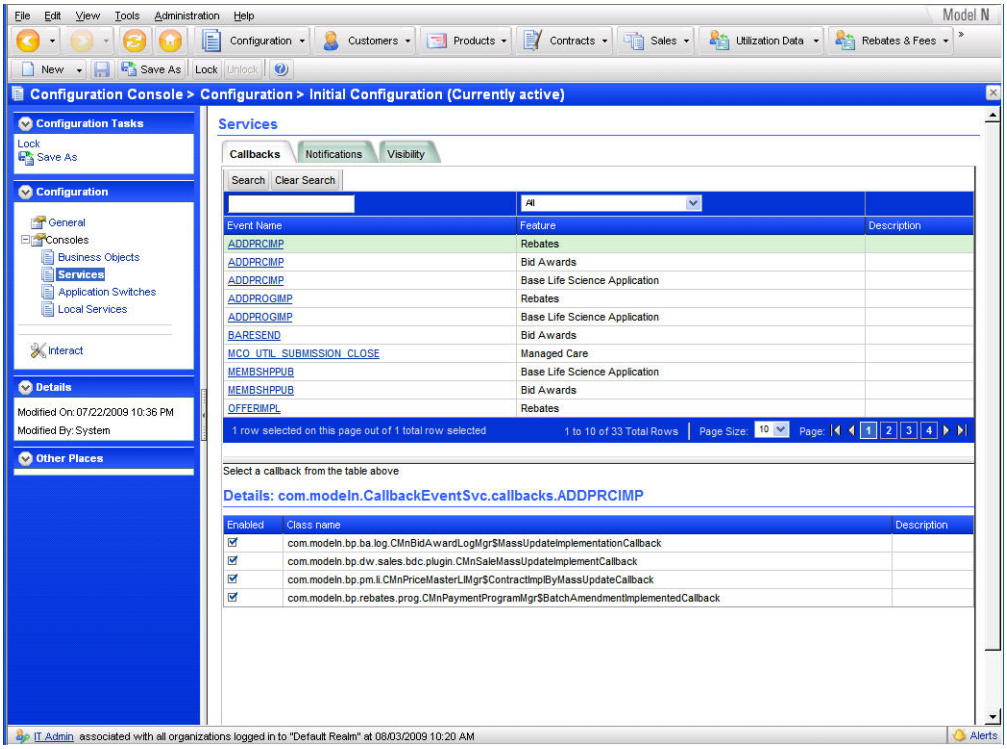
Table 8-6: Business Objects Page (Continued)

Name	Type	Description
Export Configuration	Tab	Name-Value pairs of information pertaining to exported configurations.
Extensions	Tab	Provides the list of extensions to the object. • **Name:** The unique name for the extension. • **Class Name:** The Java class name for the object extension. This class must derive from `com.modeln.infr.pof.obj.CMnFineGrainObjExtension`. • **Importer Class Name:** The Java class name for the extension to the object importer: This class must derive from `com.modeln.tools.publish.publisher.CMnBaseEntryImporterExtn`.
Properties	Tab	Provides the list of associated properties used to configure the object. • **Name:** The unique name for the property. • **Value:** The value of the property.

8.5 Services Page

Navigation Path: **Configuration > Configurations >** *Configuration Name* link **> Services**

Figure 8-6: Services Page



The Services page provides access to the callbacks, notifications, and visibility plug-ins of the selected configuration set.

Table 8-7: Services Page, Callbacks Tab

Name	Type	Description
Search		
Search	Button	Searches for callback services with the specified criteria.
Clear Search	Button	Clears your search criteria, letting you perform another search.
Event Name	Text box	The event service name to search for.

Table 8-7: Services Page, Callbacks Tab (Continued)

Name	Type	Description
Feature	Drop-down list	The associated feature to search for. Possible options are: • All • Base Application • Base Life Science Application • Bid Awards • Compliance • Custom Features • Distributor Rebates • FSS Compliance • Government Pricing • Government Pricing - FSS Compliance • Government Pricing - Managed Care • Managed Care • Media Tracking • Medicaid • Order Quantity Discount • Pharma • Pharma Distributor Rebates • Pharma Rebates • Pharma Config • Platform • Price Master • Rebates • Revenue Planning and Intelligence • Sales • US Regulatory Pharma
Results		
Event Name	Link	Callback service name. Click the link to display the Details information.
Feature	String	Feature the callback service is associated with.
Description	String	Description of the callback service.
Details		

Table 8-7: Services Page, Callbacks Tab (Continued)

Name	Type	Description
Enabled	Check box	Indicates whether or not the class name listed is enabled.
Class Name	String	Class name associated with the callback service.
Description	String	Description of the class.

Table 8-8: Services Page, Notifications Tab

Name	Type	Description
Search		
Resource Name	Text box	Notification service name to search for.
Router Class Name	Text box	Router class name for a notification service.
Search	Button	Searches for notifications with the specified criteria.
Clear Search	Button	Clears your search criteria, letting you perform another search.
Results		
Resource Name	Link	Name of a notification service
Router Class Name	String	Router class name for a notification resource.
Details		
Router Class Name	String	Router class name for the notification service.
Associated Member	String	Name of the member associated with the router class name.

Table 8-9: Services Page, Visibility Tab

Name	Type	Description
Search		
Object Name	Text box	Name of the visibility service.

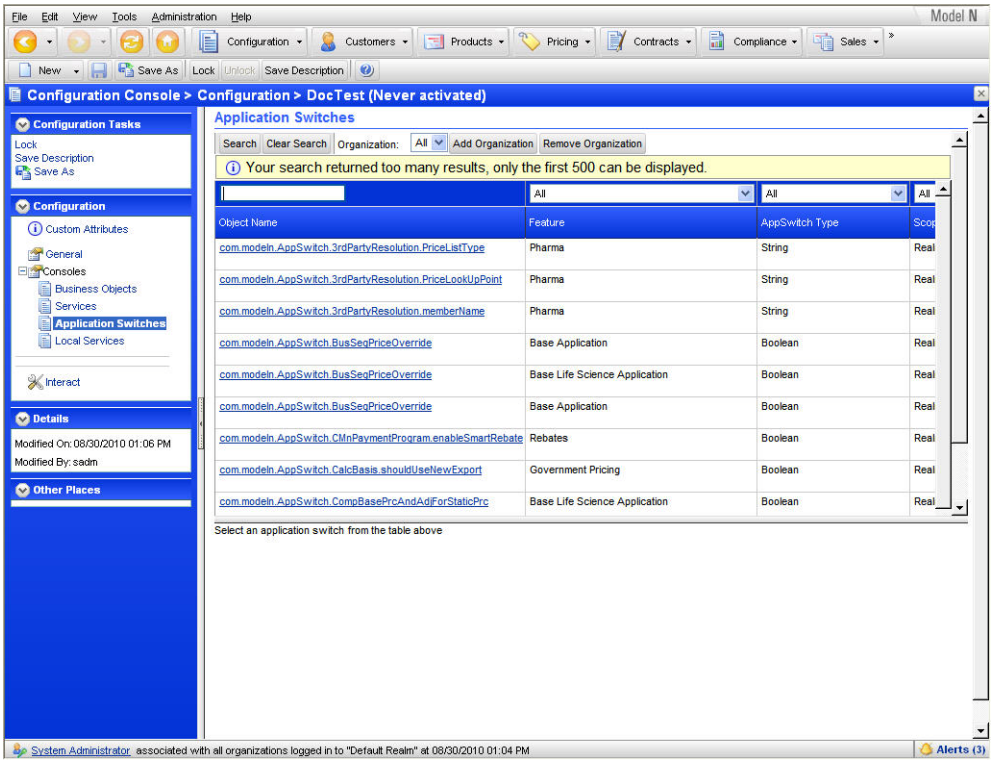
Table 8-9: Services Page, Visibility Tab (Continued)

Name	Type	Description
Plugin Class Name	Text box	Plug-in class name for the visibility service.
Search	Button	Searches for visibility services with the specified criteria.
Clear Search	Button	Clears your search criteria, letting you perform another search.
Results		
Object Name	Link	Name of the visibility service.
Plugin Class Name	String	Plug-in class name for the visibility service.
Details		
Plugin Class Name	String	Plug-in class name for the visibility service.
Feature	String	Feature with which the plug-in is associated.
Associated Member	String	Member associated with this plug-in.

8.6 Application Switches Page

Navigation Path: **Configuration > Configurations >** *Configuration Name* link **> Application Switches**

Figure 8-7: Application Switches Page



The Application Switches page provides access to the non-global application switches for the selected configuration set

Table 8-10: Application Switches Page

Name	Type	Description
Search		
Search	Button	Searches for application switches with the specified criteria.
Clear Search	Button	Clears your search criteria, letting you perform another search.
Organization	Drop-down list	Searches for application switches that are organization aware and displayes their values for the selected organization. The default value, `All`, displays all application switches regardless of organization association.

Table 8-10: Application Switches Page (Continued)

Name	Type	Description
Add Organization	Button	Opens the Select Organization dialog box to let you add an organization to the Organization drop-down list. This button is not availabe in locked configurations or the initial configuration.
Remove Organization	Button	Removes the selected organization from the Organization drop-down list. This button is not availabe in locked configurations or the initial configuration.
Object Name	Text box	Application switch name to search for.
Feature	Drop-down list	The associated feature to search for. Possible options are: • All • Base Application • Base Life Science Application • Bid Awards • Compliance • Custom Features • Distributor Rebates • FSS Compliance • Government Pricing • Government Pricing - FSS Compliance • Government Pricing - Managed Care • Managed Care • Media Tracking • Medicaid • Order Quantity Discount • Pharma • Pharma Distributor Rebates • Pharma Rebates • Pharma Config • Platform • Price Master • Rebates • Revenue Planning and Intelligence • Sales • US Regulatory Pharma

Table 8-10: Application Switches Page (Continued)

Name	Type	Description
AppSwitch Type	Drop-down list	Programmatic type of switch to search for. Option available are: • All • Short • Integer • Long • Float • Double • Boolean • String • Enumeration • Map • List • Set • Comma Separated Values • XML
Org Aware	Drop-down list	Search by whether this application switch value is specific to this organization or of it is being inherited from an ancestor organization. Options available are: • **Any** • **Yes:** inherited • **No:**specific to this organization
Organization	Chooser button	Opens the Select Organization dialog box to let you search for switches associated with a specific organization.
Results		
Object Name	Line	Application switch name. Click on the link to display the Details section.
Feature	String	Application feature to which the switch is associated.
AppSwitch Type	String	The programmatic type of switch.
Details		

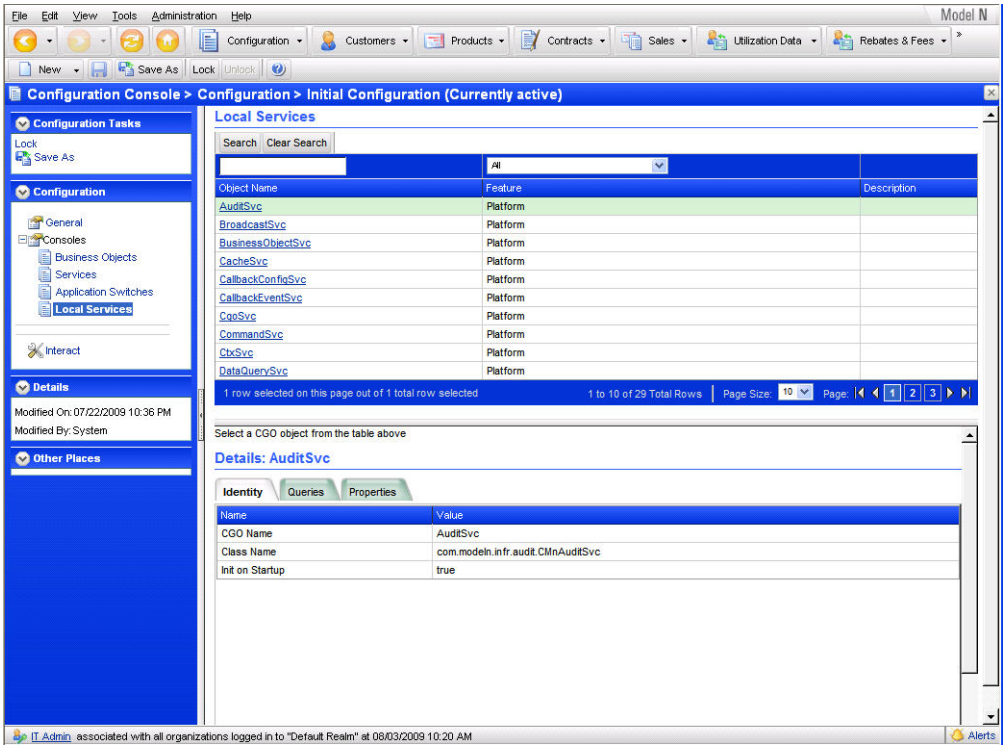
Table 8-10: Application Switches Page (Continued)

Name	Type	Description
Value	Link	Current value for the selected switch. Click on the link to change the value. **Note:** Inherited list values are not editable.
Description	String	Description of the value.
Inherited	String	Lists whether this application switch value is specific to this organization or of it is being inherited from an ancestor organization. The possible values are: • **Yes:** inherited • **No:** not inherited
Add Value	Button	Opens the Edit dialog box to add a new value to list-type application switch.
Remove	Button	Lets you remove a value from a list type applicaiton switch.
Restore to Initial Config	Button	Restores the values to their initial version. If the switch is organization aware, it will restore the switch to the inherited value.

8.7 Local Services Page

Navigation Path: **Configuration > Configurations >** *Configuration Name* link **> Local Services**

Figure 8-8: Local Services Page



The Local Services page provides access to the functional-based services of the selected configuration set.

Table 8-11: Local Services Page

Name	Type	Description
Search		
Search	Button	Searches for local services with the specified criteria.
Clear Search	Button	Clears your search criteria, letting you perform another search.
Object Name	Text box	Service name to search for.

Table 8-11: Local Services Page (Continued)

Name	Type	Description
Feature	Drop-down box	Application feature to which the service is associated. Possible values are: • All • Base Application • Base Life Science Application • Bid Awards • Compliance • Custom Features • Distributor Rebates • FSS Compliance • Government Pricing • Government Pricing - FSS Compliance • Government Pricing - Managed Care • Managed Care • Media Tracking • Medicaid • Order Quantity Discount • Pharma • Pharma Distributor Rebates • Pharma Rebates • Pharma Config • Platform • Price Master • Rebates • Revenue Planning and Intelligence • Sales • US Regulatory Pharma
Results		
Object Name	Link	Name of a local service.
Feature	String	The application feature to which the service is associated.
Description	String	Description of the local service.
Details		

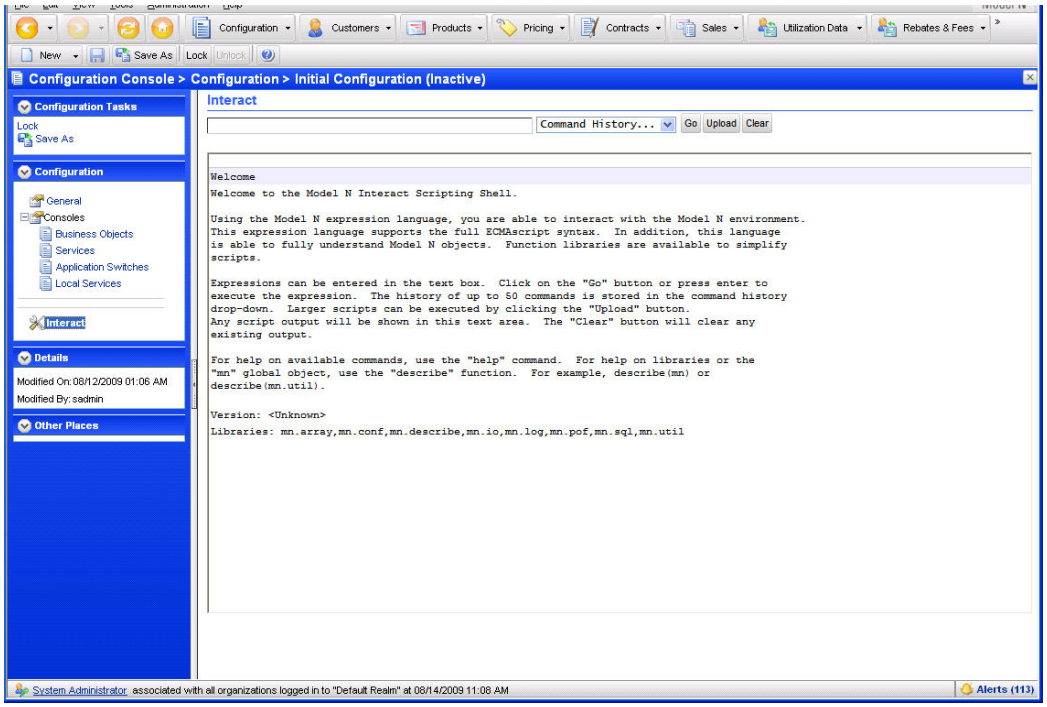
Table 8-11: Local Services Page (Continued)

Name	Type	Description
Identity	Tab	Name-value pairs providing the following information: • CGO Name • Class Name • Init of Startup
Queries	Tab	Name, type, description, and resultlimit of associated queries.
Properties	Tab	Name-value pairs of property information for the service.

8.8 Interact Page

Navigation Path: **Configuration > Configurations >** *Configuration Name* link **> Interact**

Figure 8-9: Interact Page



The Interact page provides access to the Model N interactive scripting shell where you are able to use the Model N expression language.

Table 8-12: Interact Page

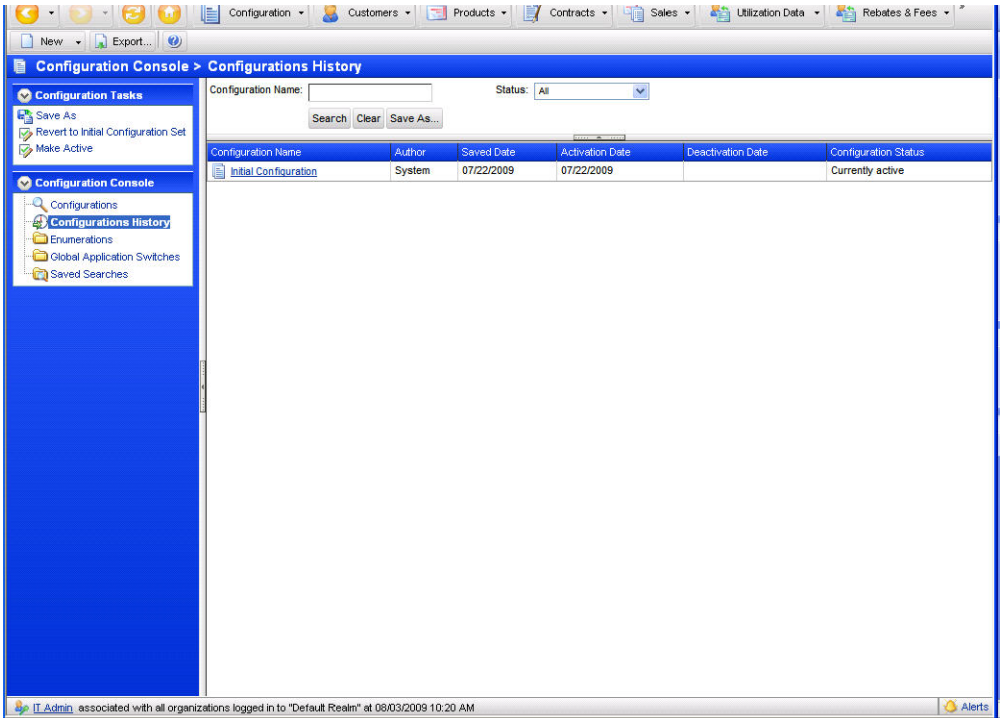
Name	Type	Description
[text box]	Text box	The expression you want to execute.
Command History	Drop-down	Lets you re-use expressions previously executed.
Go	Button	Executes the expression.
Upload	Button	Opens the Upload Script dialog box to let you execute larger scripts.
Clear	Button	Clears any existing output.

For more information on the expression languages, see Expression Language on page 205 for more information.

8.9 Configurations History Page

Navigation Path: **Configuration > Configurations History**

Figure 8-10: Configurations History Page



The Configurations History page provides information about the configuration sets that have been used in your instance of the Model N application.

Table 8-13: Configurations History Page

Name	Type	Description
Search		
Configuration Name	Text box	Name of the saved configuration to search for.
Status	Drop-down list	The current status of the configuration to search for. Possible values are: • All • Active on next start • Currently active • Inactive • Never activated

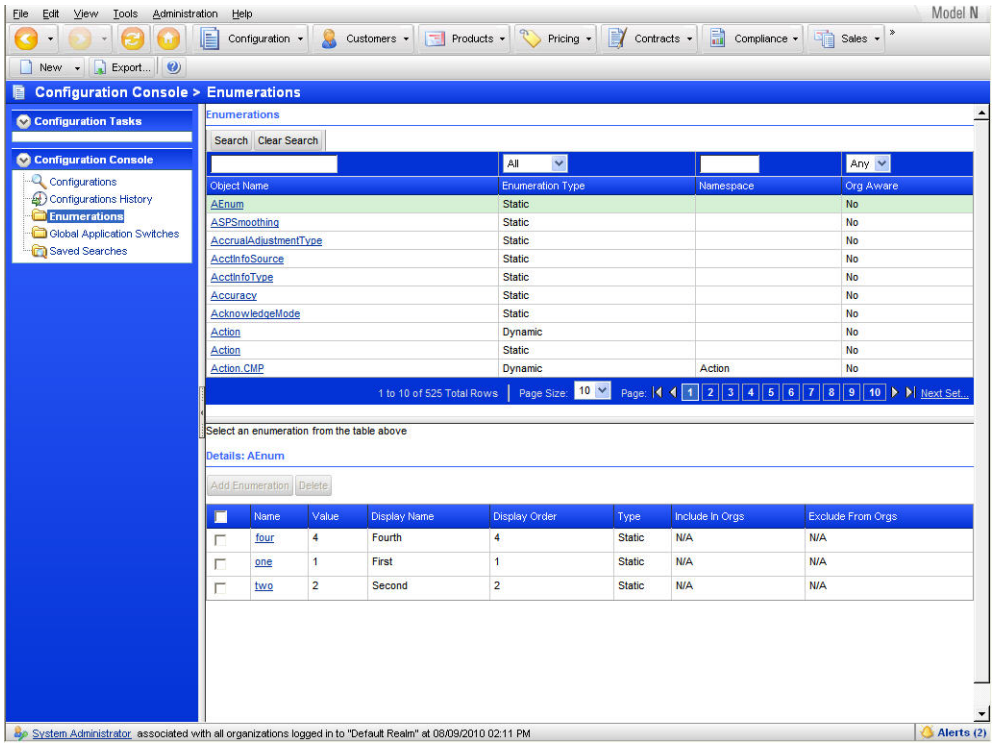
Table 8-13: Configurations History Page (Continued)

Name	Type	Description
Search	Button	Searches for configurations with the specified criteria.
Clear	Button	Clears your search criteria, letting you perform another search.
Save As	Button	Lets you save your search criteria to perform the search again at a later date.
Results		
Configuration Name	Link	Name of the saved configuration.
Author	String	Creator of the saved configuration.
Saved Date	Date	Date the configuration was saved.
Activation Date	Date	Date the configuration was last activated.
Deactivation Date	Date	Date the configuration was deactivated.
Configuration Status	String	The current status of the configuration. Possible values are: • All • Active on next start • Currently active • Inactive • Never activated

8.10 Enumerations Page

Navigation Path: **Configuration > Enumerations**

Figure 8-11: Enumerations Page



The Enumerations page provides access to enumerations in the Model N application.

Table 8-14: Enumerations Page

Name	Type	Description
Configuration Tasks		
Create Enumeration	Link	Opens the Create Enumertion dialog box where you can create a new enumeration and specify if it is organization aware.
Search		
Search	Button	Searches for enumerations with the specified name, type, and namespace. If the name or namespace fields are left blank.
Clear Search	Button	Clears your search criteria, letting you perform another search.
Object Name	Text box	Enumeration name to search for.

Table 8-14: Enumerations Page (Continued)

Name	Type	Description
Enumeration Type	Drop-down list	Type of enumeration to search for. Options available are: • All • Bit Flags • Dynamic • Static • Runtime
Namespace	Text box	Enumeration namespace to search for.
Org Aware	Drop-down list	Lets you search by whether or not an enumeration is org aware. Options available are: • Any • Yes • No
Results		
Object Name	Link	Name of the enumeration.
Enumeration Type	String	The type of enumeration. Possilbe values are: • Bit Flags • Dynamic • Static • Runtime
Namespace	String	The namespace of the enumeration.
Org Aware	String	Whether or not an enumeration is organization aware, meaning that specific values can be targeted to specific organizations.
Details		
Add	Button	Opens the Add Dialog Box where you can add enumeration values to the selected enumeration.
Delete	Button	Lets you delete enumeration values. The button is only active if the check box is selected.

Table 8-14: Enumerations Page (Continued)

Name	Type	Description
Edit Enumeration	Button	Opens the Edit Enumeration dialog box where custom attribute values can be added.
[check box]	Check box	Selected enumerations can be deleted.
Name	Link	Name of the enumeration. Clicking on this link opens the Add Dialog Box where you can modify the display name and display order.
Value	String	Value of the enumeration.
Display Name	String	User-friendly name for that enumeration as it will display in the user interface.
Display Order	String	User-friendly version for the enumeration values as it will display in the user interface.
Type	String	The type of enumeration.
Included In	String	The organizations that this enumeration value applies to.
Excluded From	String	The organizations that are not associated with this enumeration value.

8.10.1 Create Enumeration Dialog Box

Navigation Path: **Configuration > Enumerations > Create Enumeration**

The Create Enumeration dialog box lets you create a dynamic runtime enumeration..

Table 8-15: Create Enumeration Dialog Box

Name	Type	Description
Name *	Text box	Name of the enumeration.
Organization Aware	Check box	Select this option to make this enumeration organization aware.
Add	Button	Lets you add an attribute.
Delete	Button	Lets you remove an attribute.
[check box]	Check box	Used with the Delete button to indicate which attribute to remove.
Attribute Name*	Text box	Lets you specify a name for the attribute.

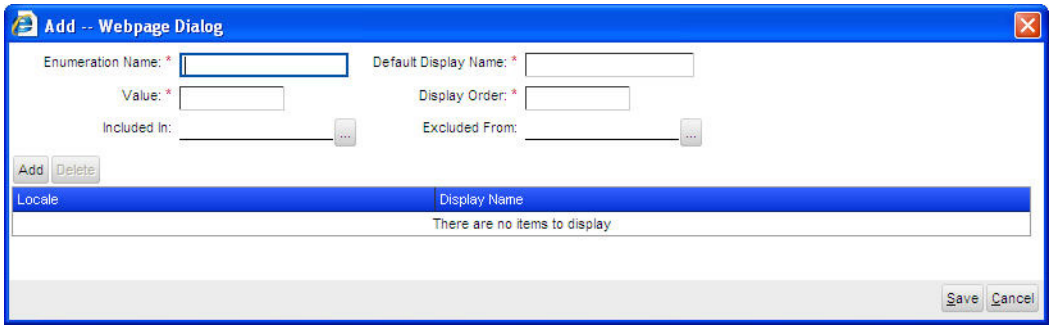
Table 8-15: Create Enumeration Dialog Box (Continued)

Name	Type	Description
Attribute Type *	Drop-down list	Lets you select the type of the attribute.
OK	Button	On close, the newly created enumeration displays in the table so you can immediately add values.
Cancel	Button	Closes the dialog box without saving changes.

8.10.2 Add Dialog Box

Navigation Path: **Configuration > Enumerations >** *Object Name* link **> Add**

Figure 8-12: Add Dialog Box



The Add dialog box provides access to the display information for enumerations. Any additional attributes defined for this enumeration will also be displayed with an input box of the attribute type.

Table 8-16: Add Dialog Box

Name	Type	Description
Name *	Text box	Name of the enumeration.
Display Name *	Text box	User-friendly name for that enumeration as it will display in the user interface.
Value *	Text box	Value of the enumeration.
Display Order *	Text box	User-friendly version for the enumeration values as it will display in the user interface.

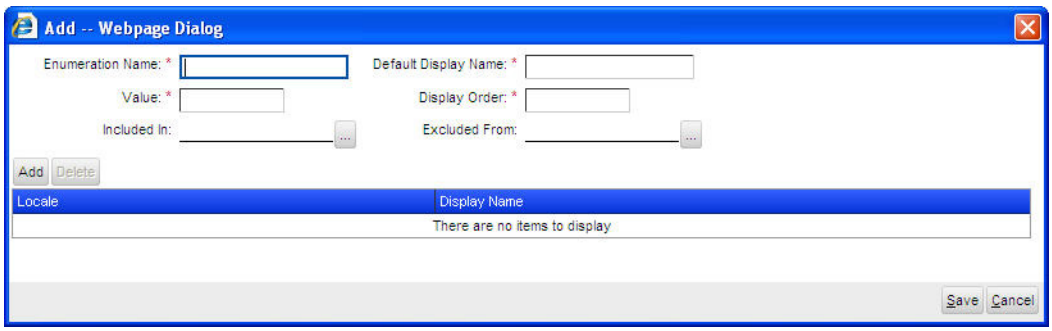
Table 8-16: Add Dialog Box (Continued)

Name	Type	Description
Included In	Text box and Chooser button	Opens the Select Organization dialog box to select the organizations to which this enumeration applies. This field is only visible for organization-aware enumerations.
Excluded From	Text box and Chooser button	Opens the Select Organization dialog box to select the organizations explicitly not associated with this enumeration value. This field is only visible for organization-aware enumerations.
Add	Button	Lets you add a locale to the enumeration value.
Delete	Button	Lets you remove a locale from an enumeration value.
[check box]	Check box	Used with the Delete button to indicate which locale to remove
Locale *	Drop-down list	Lets you select the locale to apply to the enumeration value.
Display Name *	String	Lets you specify a display name for the locale.
Save	Button	Saves any changes made to the enumeration.
Cancel	Button	Closes the dialog box without saving changes.

8.10.3 Edit Dialog Box

Navigation Path: **Configuration > Enumerations >** *Object Name* link **> Edit Enumeration**

Figure 8-13: Edit Dialog Box



The Edit Enumeration dialog box provides the ability to the define custom attributes for enumerations. Any additional attributes defined for this enumeration will also be displayed with an input box of the attribute type.

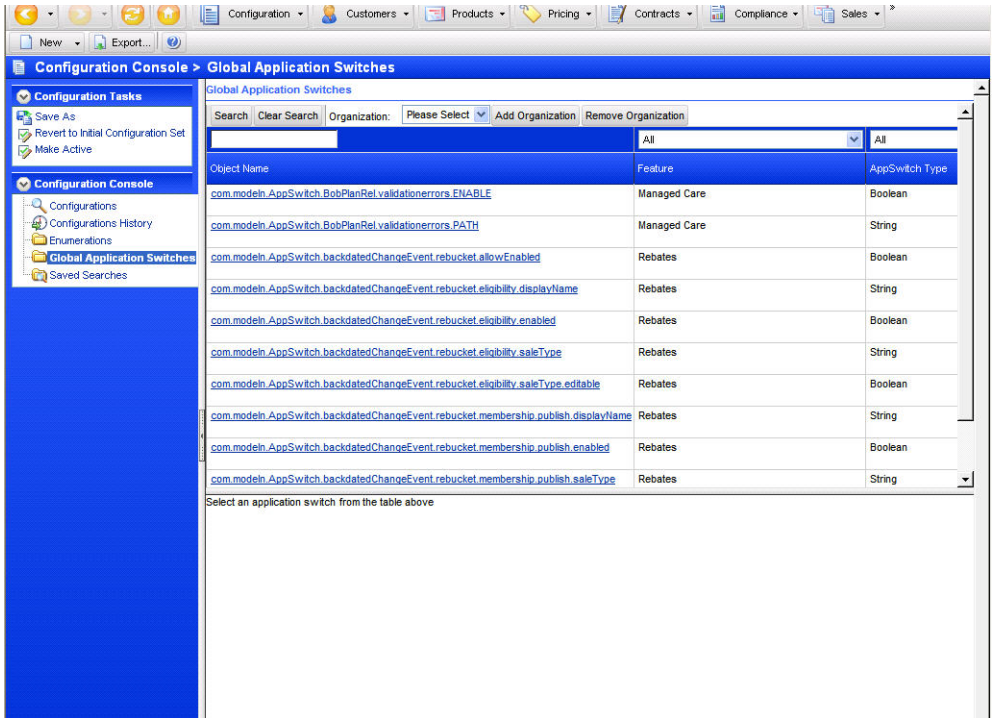
Table 8-17: Edit Dialog Box

Name	Type	Description
Name *	Text box	Name of the enumeration.
Organization Aware	Check box	Indicates if the enumeration is organization aware.
Add	Button	Adds a row to the table where you can specify an attribute.
Delete	Button	Lets you delete selectedattributes from the enumeration.
[check box]	Check box	Lets you select attributes for deletion.
Attribute Name *	Text box	Enter the name for the attribute.
Attribute Type *	Drop-down list	Lets you select the type of attribute to add.
OK	Button	Saves any changes made to the enumeration.
Cancel	Button	Closes the dialog box without saving changes.

8.11 Global Application Switches Page

Navigation Path: **Configuration > Global Application Switches**

Figure 8-14: Global Application Switches Page



The Global Application Switches page provides access to the global application switches for the Model N application.

Table 8-18: Global Application Switches Page

Name	Type	Description
Search		
Search	Button	Searches for application switches with the specified criteria.
Clear Search	Button	Clears your search criteria, letting you perform another search.
Organization	Drop-down list	Searches for application switches that are organization aware and displays their values for the selected organization. The default value, `Please Select`, displays all application switches regardless of organization association.

Table 8-18: Global Application Switches Page (Continued)

Name	Type	Description
Add Organization	Button	Opens the Select Organization dialog box to let you add an organization to the Organization drop-down list.
Remove Organization	Button	Removes the selected organization from the Organization drop-down list.
Object Name	Text box	Application switch name to search for.
Feature	Drop-down list	Search criteria for the feature to which the switch is associated. Options available are: • All • Base Application • Base Life Science Application • Bid Awards • Compliance • Custom Features • Distributor Rebates • FSS Compliance • Government Pricing • Government Pricing - FSS Compliance • Government Pricing - Managed Care • Managed Care • Media Tracking • Medicaid • Order Quantity Discount • Pharma • Pharma Distributor Rebates • Pharma Rebates • Pharma Config • Platform • Price Master • Rebates • Revenue Planning and Intelligence • Sales • US Regulatory Pharma

Table 8-18: Global Application Switches Page (Continued)

Name	Type	Description
AppSwitch Type	Drop-down list	Programmatic type of switch to search for. Option available are: • All • Short • Integer • Long • Float • Double • Boolean • String • Enumeration • Map • List • Set • Comma Separated Values • XML
Realm	Text box	Search criteria to limit results to those with the specified realm.
Org Aware	Drop-down list	Search by whether this application switch value is specific to this organization or of it is being inherited from an ancestor organization. Options available are: • **Any** • **Yes:** inherited • **No: specific** to this organization
Organization	Chooser button	Opens the Select Organization dialog box to let you search for switches associated with a specific organization.
Results		
Object Name	Link	Application switch name. Click this link to display information in the Details section.
Feature	String	Feature to which the switch is associated.
AppSwitch Type	String	Programmatic type the switch uses as its value.

Table 8-18: Global Application Switches Page (Continued)

Name	Type	Description
Requires Restart	String	Whether or not you must restart the application server after the value of the switch is changed.
Org Aware	String	Lists whether this application switch value is specific to this organization or of it is being inherited from an ancestor organization.
Organization	String	List the organization from which this switch inherits its value.
Details		
Object Name	String	Application switch name.
Description	String	Description of the application switch.
Value	Link	Current value of the switch. Clicking on this link opens the Edit Application Switch dialog box where you can change the value and description of the switch.
Description	String	Description associated with the current value.
Inherited	String	Lists whether this application switch value is specific to this organization or of it is being inherited from an ancestor organization. The possible values are: • **Yes:** inherited • **No:** specific to this organization
Edit	Button	Lets you edit the value and description of the application switch.
Add Value	Button	Lets you add additional values to the application switch. This button is only available for list-type switches.
Remove	Button	Lets you remove additional values to the application switch. This button is only available for list-type switches.

8.11.1 Edit Application Switch Dialog Box

Navigation Path: **Configuration > Global Application Switches>** *Object Name* link **> Edit** or **Add Value**

Figure 8-15: Edit Application Switch Dialog Box

The Edit Application Switch dialog box provides access to the values and description for an application switch.

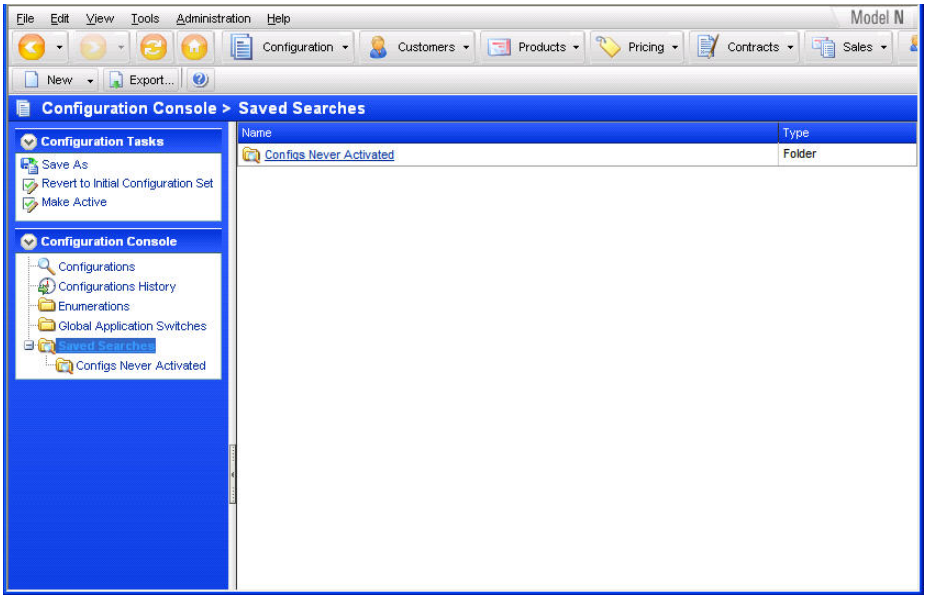
Table 8-19: Add Enumeration Dialog Box

Name	Type	Description
Value	Text box	Value of the application switch. For Boolean values, a Chooser button is available to provide access to both values.
Description	Text box	Description of the application switch value.
Save	Button	Saves any changes made.
Cancel	Button	Closes the dialog box without saving any changes.

8.12 Saved Searches Page

Navigation Path: **Configuration > Saved Searches**

Figure 8-16: Saved Searches Page



The Saved Searches page provides access to saved sets of search criteria from the Configuration Console. Clicking the **Name** link takes you to the appropriate page and performs the specified search

9 Sarbanes-Oxley

The Sarbanes-Oxley Act of 2002 required that all corporations with greater than seventy five million dollars in annual revenue be prepared to pass an external audit attesting to the effectiveness of the controls and processes that have been put in place to assure reliable financial reporting.

This chapter provides information on how the Model N application has implemented functionality that helps address Sarbanes-Oxley (SOX) compliance issues. Starting with the Model N 5.2 release, the Model N system addresses a subset of SOX functionality related to section 404 of the Sarbanes-Oxley Act.

The following sections are covered in this chapter:

9.1 Auditor Role

An auditor role is a read-only role in the application to let external auditors access the application for SOX reporting and analysis functions. An auditor cannot make changes to data or processes in the production environment.

The auditor has read-only privileges to all Model N applications except Reporting, which grants the auditor access to:

- view standard templates, custom templates, and reports
- run and save reports
- schedule report runs
- create custom templates using save as from standard or other custom templates
- share templates and reports with other users.

Note: The auditor cannot create new standard report templates, ad hoc templates or ad hoc reports.

9.2 Tagging Terms and Conditions

You can identify the type of standard and non-standard terms that have been added to contracts and report against them.

A term category is standard if the term is present in the Term Library and non-standard if the term has been uploaded. The Term Library holds the standard terms that are stored in the system. There can be more than one Term Library. When you load a non-standard term, you are required to tag it with a particular term type.

The following out-of-the-box term types are supported for standard and non-standard terms:

- Warranties
- Payment Terms
- Shipment Terms
- Pricing Programs

9.3 How to Change Approval Routing

To turn on or off approver skipping, use the com.modeln.AppSwitch.DocRoute.AllowSkip application switch. When set to true, approvers can be skipped during document routing for approval. When set to false, approvers cannot be skipped.

To configure the rules for accepting, rejecting, and delegating approvers, use the com.modeln.AppSwitch.DocRoute.approvalAuthorityOption application switch.

9.4 How to Lock a User Out of the System

You can implement an account lockout policy using the com.modeln.AppSwitch.user.maxConsecutiveInvalidLoginAttempts and the com.modeln.AppSwitch.user.invalidLoginAttemptsPeriod application switches set at deployment time. The account remains locked until the system administrator resets the account login.

9.5 How to Deactivate a User Account

You can configure a user account deactivation after a certain period of inactivity using the com.modeln.AppSwitch.user.maxDaysOfInactivity application switch. It is a global setting applicable across all users.

9.6 How to Enforce a Password Change

Using the following application switches, you can force users to change their password initially upon login (optional) and then every N (specified) number of days.

- com.modeln.AppSwitch.user.password.ForceChangeOnInitialLogin
- com.modeln.AppSwitch.user.password.passwordLifeInDays
- com.modeln.AppSwitch.user.password.passwordWarningDays
- com.modeln.AppSwitch.user.password.passwordHistoryCount

When a user is within X (specified) days of password expiration, the user is prompted at each login to change the password. If the user logs in after the password has expired, the user is prompted to change the password. If the user does not change the password, he is not allowed to log into the system. The password change every N number of days is a global option available across all users. The password change upon login option is set per user through the user interface.

9.7 How to Enforce Strong Passwords

You can enforce strong passwords across all users by using the com.modeln.AppSwitch.user.password.LengthValidator.minLength and the com.modeln.AppSwitch.user.password.CharVarietyValidator.minGroups application switches.

9.8 How to Prevent Repeating Passwords

You can prevent users from repeating passwords that they have used in the past by using the com.modeln.AppSwitch.user.password.passwordHistoryCount and the com.modeln.AppSwitch.user.password.passwordHistoryFailureReason application switches.

9.9 Relevant Application Switches

9.9.1 DocRouteSwitches.xml in Base Global Realm

Location: `app-content.jar, /base/Global/Importers/Configuration/DocRouteSwitches.xml`

9.9.1.1 com.modeln.AppSwitch.DocRoute.AllowSkip

This switch enables the skip functionality in document routing (approval routing). Possible values TRUE or FALSE. By default this is set to TRUE

9.9.1.2 com.modeln.AppSwitch.DocRoute.approvalAuthorityOption

This switch provides the options to configure the rules for accepting/rejecting/delegating approvers in document routing (approval routing). By default, this switch is set to let all users to have the ability to accept/reject/delegate approvers.

Possible values	**all**	All users have the ability to accept/reject/delegate approvers
	sysadm	Only sys admin users have the ability to accept/reject/delegate approvers
	sysadmAndSubmitter	Only sys admin plus the document submitters have the ability to accept/reject/delegate approvers

9.9.2 BaseSwitches.xml in Base Global Realm

Location: `platform-content.jar, /base/Global/Importers/Configuration/BaseSwitches.xml`

9.9.2.1 com.modeln.AppSwitch.user.password.LengthValidator.minLength

Integer-valued property defining the minimum length of passwords accepted by the `CMnLengthPasswordValidator`. Note that this application switch will be used only if that validator has been associated with the community via a community property as defined in the package documentation for the com.modeln.infr.cmty.user package.

9.9.2.2 com.modeln.AppSwitch.user.password.CharVarietyValidator.minGroups

Integer valued property defining the minimum number of character classes required in a password for it to be considered valid by the `CMnCharVarietyPasswordValidator`. To be valid, a password must contain at least one character from at least this number of the following character groups: lower-case letter (a-z), upper-case letters (A-Z), digits (0-9), and all others. Please see the Javadoc of the `CMnCharVarietyPasswordValidator` class for details

9.9.2.3 com.modeln.AppSwitch.user.password.ForceChangeOnInitialLogin

Value indicates whether the users must change their passwords on initial login. Possible values TRUE or FALSE. By default this is set to TRUE. When the application switch is set to TRUE, all users must change their passwords on initial login and the Temporary Password check box is grayed-out so that the system administrator cannot de-select the option.

When an account is activated with Temporary Password unchecked, the user should not be prompted to change the password on first login.

You must set the application switch to agree with whether the user's account is activated with the Temporary Password checked or unchecked. Otherwise, a compliance issue may arise when users do not have to change their passwords even though they were previously forced to change their passwords.

9.9.2.4 com.modeln.AppSwitch.user.password.passwordLifeInDays

Value indicates the life of a password from the day it gets set.

9.9.2.5 com.modeln.AppSwitch.user.password.passwordWarningDays

Value indicates the number of days prior to password expiration when the user has to be warned of the impending expiration.

9.9.2.6 com.modeln.AppSwitch.user.password.passwordHistoryCount

Value indicates the number of recently used passwords that will not be allowed for current use. The maximum number of passwords maintained in history will be 16.

9.9.2.7 com.modeln.AppSwitch.user.password.passwordHistoryFailureReason

Reason for password history validation failure. The default value is:

```
Your password does not meet password policy requirements, Please
contact your system administrator for more information
```

9.9.2.8 com.modeln.AppSwitch.user.maxConsecutiveInvalidLoginAttempts

Maximum allowed consecutive invalid login attempts made within a half hour period

9.9.2.9 com.modeln.AppSwitch.user.invalidLoginAttemptsPeriod

The period within which a user account is locked if the number of invalid consecutive login attempts increases configured number. Specify the value in minutes. Default value is 30 minutes.

9.9.2.10 com.modeln.AppSwitch.user.maxDaysOfInactivity

Maximum allowed period of inactivity from the last login time.

10 Expression Language

The Model N expression language is a foundation for configurability within the application server layer of the Model N applications. This JavaScript language lets the user define flexibility in areas where data needs to be conditionalized or criteria needs to be defined at runtime.

The expression language is used in Print Templates.

Detailed documentation about the expression language is available in JSDoc as either `pharma-eldoc.zip` or `meddev-eldoc.zip`.

10.1 Language Support

The Model N expression language supports the full ECMAscript syntax and is able to fully understand Model N objects.

10.1.1 Core Libraries

The core libraries exist in any instance of the expression language runtime and are located under the class path `WEB-INF/classes/platform/expr/`. The libraries available are:

Table 10-1: Core Libraries

Name	Description
mn.array	Allows for the creation and manipulation of native arrays.
mn.conf	Provides access to the Model N configuration.
mn.describe	Provides support for runtime introspection and help.
mn.io	Provides support for input/output from the scripting environment.
mn.log	Provides access to logging.
mn.pof	Provides access to query and manipulate FGOs.
mn.sql	Provides SQL query access to the Model N database data.
mn.util	Contains miscellaneous information.

Detailed information about these libraries is available in JSDoc as either `pharma-eldoc.zip` or `meddev-eldoc.zip`.

10.2 How to Access the Expression Language

To access the expression language:

- Log into the Model N application as an admin and go to the Interact page available from the detail pages under the Admin menu
- Use the Ops tool by running the "Interact" ops tool task.

There are two levels of access available:

- restricted: provides secure access for end users. Only a subset of the Model N expression libraries are exposed.
- general: users have more access to Model N data from the expression language. It is designed for those with administration privileges.

Restricted mode has the following constraints:

- No Java API calls
- No access to Class object
- No access to specific MN objects (for example, mn.ctx, mn.env)
- No access to certain libraries (for example. mn.pof, mn.sql)

10.2.1 Commands

The following list of commands are available for use from the Interact page:

Table 10-2: Expression Language Commands

Command	Description
!	Run a command from the history Syntax: ! <cmd-id>, or !! for the last command
clear	Clears the output
exit	Exits the shell
help	Lists available shell commands
history	Lists commands that have been executed
reset	Recreates the shell. This command reloads the libraries and starts a new engine without having to bring down the server.
spool	Lets the output of the shell to be redirected to a file. Available only in the command line shell.

10.3 How to Expose Getter Methods

If you are in restricted mode and want access to an FGO "getter" method, that is, a method that can be accessed as "getFoo()" which either has no arguments or only the `CMnCtx` argument (rules defined in `CMnReflectionHelper`), then you can define access to it in the FGO definition. The syntax would be:

```
<fgo name="…">
  <property name="script.<name>" value="<attrPath>"/>
</fgo>
```

where <*name*> is the name you want to use in the scripting language and the `<attrPath>` is the attribute path (that is `x.y.z`) from the FGO. Model N recommends that you use the same <*name*> as the <*attrPath*> unless you have some real reason not to.

Example:

To expose `IMnMember.getPrintableName(ctx)`, register the following:

```
<fgo name="Member">
   …
   <property name="script.printableName" value="printableName"/>
</fgo>
```

11 Database Maintenance

This section of the document contains information relevant to the maintenance of your database.

11.1 How to Maintain Open Tables

The following SQL script needs to be executed on a periodic basis to ensure staging tables do not grow too large.

Note: This must be run with the application server shut down.

You can estimate how often this script should be run based on the data load volume. It is recommended that you execute the script after 2,000,000 submission lines are closed in the sale tables.

Code 11-1: Script for Maintaining Open Tables

```
SET SERVEROUTPUT ON SIZE 32000

BEGIN
  mn_db_util_pkg.shrink_table('MN_MCO_UTIL_OPEN');
  mn_db_util_pkg.shrink_table('MN_CUSTOM_SALE_OPEN');
  mn_db_util_pkg.shrink_table('MN_INDIR_SALE_OPEN');
```

Code 11-1: Script for Maintaining Open Tables (Continued)

```
  mn_db_util_pkg.shrink_table('MN_DIR_SALE_OPEN');
  mn_db_util_pkg.shrink_table('MN_RBTPMT_SALE_OPEN');
  mn_db_util_pkg.shrink_table('MN_WORKBOOK_INT_RESULT');
END;
/
```

11.2 How to Maintain a Partitioned Database

`MN_PARTITION_PKG` contains the set of APIs used to support partitioning:

Table 11-1: Partitioning APIs

API	Description
partition_schema	Performs the partitioning of all objects, defined in the MN_PARTITIONS table.
partition_table	Partitions a single table.
partition_index	Partitions a single index.
add_partitions	Adds the partitions for the RANGE type partitioned tables and indexes.
drop_partition	Drops a single named partition.
is_partitioned	Verifies if the object is partitioned or not.
print_partition_info	Prints the partition information about all partitioned tables in the schema (use full name).
print_table_partition_info	Prints the table and table indexes partition information.
print_index_partition_info	Prints the index partition information.

The DBA likely should access add_partitions once a month to create additional partitions.

11.2.1 Partitioning Notification

A timer job is setup to be run on a monthly basis to verify the partitioning information in the database and send a notification email, if at least one of the tables does not have the partition defined for the current date. The email contains the list of tables that don't have the current partitions and the maintenance API that has to be called.

The application switch `com.modeln.AppSwitch.DBAEmailID` should contain the email address of the DBA that should receive a notification when the partitioning maintenance API (`add_partitions`) needs to be run. If the switch contains no value, notifications will not be sent.

11.2.2 Procedure PRINT_PARTITION_INFO

This API prints out the partition information for an object. If the object name is NULL, it wi l print the information for all partitioned tables in the schema and for the tables registered in MN_PARTITIONS.

Table 11-2: Procedure PRINT_PARTITION_INFO

Parameter name	Type	Data type	Default	Description
p_object_type	IN	VARCHAR2		Object name
p_object_name	IN	VARCHAR2		Object type

11.2.3 Function PRINT_PARTITION_INFO

This is the same as `procedure PRINT_PARTITION_INFO`, but returns the information as VARCHAR2.

Table 11-3: Function PRINT_PARTITION_INFO

Parameter name	Type	Data type	Default	Description
p_output	IN	VARCHAR2	N	If the parameter is Y the same info will be printed, using dbms_output
p_object_type	IN	VARCHAR2		Object name
p_object_name	IN	VARCHAR2		Object type

11.2.4 Procedure PRINT_TABLE_PARTITION_INFO

These APIs print out the partition information for a table and all table indexes.

Table 11-4: Procedure PRINT_TABLE_PARTITION_INFO

Parameter name	Type	Data type	Default	Description
p_table_name	IN	VARCHAR2		Table name

11.2.5 Function PRINT_TABLE_PARTITION_INFO

This is the same as `procedure PRINT_TABLE_PARTITION_INFO`, but returns the information as VARCHAR2.

Table 11-5: Function PRINT_TABLE_PARTITION_INFO

Parameter name	Type	Data type	Default	Description
p_output	IN	VARCHAR2	N	If the parameter is Y the same info will be printed, using dbms_output
p_table_name	IN	VARCHAR2		Table name

11.2.6 Procedure PRINT_INDEX_PARTITION_INFO

These APIs print out the partition information for an index.

Table 11-6: Procedure PRINT_INDEX_PARTITION_INFO

Parameter name	Type	Data type	Default	Description
p_index_name	IN	VARCHAR2		Index name

11.2.7 Function PRINT_INDEX_PARTITION_INFO

This is the same as the `procedure PRINT_INDEX_PARTITION_INFO`, but returns the information as VARCHAR2.

Table 11-7: Function PRINT_INDEX_PARTITION_INFO

Parameter name	Type	Data type	Default	Description
p_output	IN	VARCHAR2	N	If the parameter is Y the same info will be printed, using dbms_output
p_index_name	IN	VARCHAR2		Index name

12 Web Services

Web Services is a software system that lets clients and servers communicate over a network using XML messages that follow the Simple Object Access Protocol (SOAP) standard, a protocol for exchanging XML-based, extensible messages over a computer network.

This chapter provides an overview of the Job Web Service APIs that are exposed by the Model N application so that you can interface to the Model N job scheduler to call it externally and incorporate it with the outcome of jobs from other applications that you deploy.

Model N exposes the following external job APIs as web service operations:

- Create and Schedule Jobs
- Search Jobs
- Query Job Status
- Cancel a Job
- Schedule Data Flows
- Search Data Flows
- Query Data Flow Status
- Reporting Scheduling

Operations that are supported by the server are written in Web Services Description Language (WSDL). The Web Services WSDL, The Job Web Service WSDL, and The Reporting Web Services WSDL for these exposed job APIs are also included in this chapter.

12.1 Create and Schedule Jobs

The job creation and scheduling web service lets you schedule new jobs asynchronously. The response to the client includes a unique identifier that you can use to query the status of a job. This service operation only supports single schedule.

Following is the input for this web service:

- Callback Class - the class name
- Job Identifier - the name of the job
- Maximum Retry Count - the maximum retry count
- Timer Information XML - contains the same attribute map that is specified in the timer store content. It is in XML format and should be wrapped in tag `<![CDATA[<AttrMap/> ]]>`.
- Job Type - the resource path for this job

Note: These input attributes are the same values used by the `TimerStorefromXml` data flow.

Following is the output for this web service:

- Job ID - the primary key in the timer table
- Job Identifier
- Status - for example, done, received, or scheduled
- Message - contains error information
- Start Date
- End Date

12.2 Search Jobs

The search jobs web service lets you query jobs with capabilities similar to those of the local command service.

Following is the input for this web service:

- Callback Class
- Job Identifier
- Status
- Submitted by
- Start Date

The output is the job status.

12.3 Query Job Status

The query job status web service lets you check the status of a job (such as scheduled or failed) and job-specific information. For example, for data flow callbacks, the response should also include the number of errors and warnings.

The input for this web service is job ID.

The output is the Job Status.

12.4 Cancel a Job

The cancel a job web service lets you cancel a given job in a way similar to that of the local interface.

Following is the input for this web service:

- Job ID

The output is the job status.

12.5 Schedule Data Flows

The data flow scheduling web service lets you schedule a data flow. Data flow scheduling is an asynchronous process.

Following is the input for this web service:

- Configuration Name
- Owner Name
- Org

> **Note:** The name for the organization to be used for data flow imports can be specified as the value of the `orgName` element for the data flow web service operations `updateDataFlow` and `scheduleDataFlow`. To maintain backward compatibility for the web services, `orgName` is defined as an optional argument. If no value is specified, the web service will default to the Root org.

The output is the data flow status.

12.6 Search Data Flows

The search data flow web service lets you search data flows by data flow configuration name.

Following is the input used when you search for data flows:

- Configuration Name
- Data Flow Run Name
- Message Type
- End Date

The output is a list of data flow status.

12.7 Query Data Flow Status

The query data flow status web service lets you check the status of a job (such as scheduled or failed) and job-specific information.

The input for this web service is data flow run ID, which is given in the response when you schedule a data flow using the data flow scheduling web service. The output for this web service is the following:

- Data Flow Run ID
- Data Flow Configuration Name
- Data Flow Run Name
- Error Count
- Warning Count
- Number of Entries Processed
- Number of Entries Processed that Succeeded
- Number of Entries Processed that Failed
- Message Summary
- Start Date
- End Date

12.8 Reporting Scheduling

The reporting scheduling web services let you create and schedule a report for generation. You can also obtain the generation status, cancel generation, and search reports.

Following are the web services and their inputs:

- scheduleReport

  The input is reportName, which is the name of the report to generate. This is a required parameter and it outputs a unique identifier for this report generation that is used for other web services

- getReportStatus

Returns the current status of a report for generation. The input is identifier, which is the identifier of the report generation and is a required parameter.

- searchReports

  Returns the list of reports generated that fit the supplied criteria. Following is the input for this web service. All of the parameters are optional. Both date fields must be specified to be used in the search:

  - identifier

    the identifier of the report generation

  - status

    the current status of the report generation

  - startDate

    start date of the report generation

  - endDate

    end date of the report generation

- cancelReport:

  Cancels the report generation. The input is identifier, which is the identifier of the report generation and is a required parameter.

12.9 The Web Services WSDL

Code 12-1: Web Services WSDL

```
<complexType name="wSJobStatusResult">
<sequence>
  <element name="jobId" type="long" />
  <element name="jobIdentifier" type="string" />
  <element name="status" type="string" />
  <element name="message" type="string" />
  <element name="startDate" type="dateTime" nillable="true" />
  <element name="endDate" type="dateTime" nillable="true" />
  </sequence>
  </complexType>
<complexType name="dfSearchCriteria">
<sequence>
  <element name="confName" type="string" />
  <element name="dfRunName" type="string" />
  <element name="msgType" type="string" />
  <element name="startDate" type="dateTime" nillable="true" />
  <element name="endDate" type="dateTime" nillable="true" />
  </sequence>
  </complexType>
<complexType name="wSDataFlowStatusResult">
<sequence>
```

Code 12-1: Web Services WSDL (Continued)

```
  <element name="dfRunId" type="long" />
  <element name="confName" type="string" />
  <element name="dfRunName" type="string" />
  <element name="errorCount" type="integer" />
  <element name="warningCount" type="integer" />
  <element name="numEntriesProcessed" type="integer" />
  <element name="numEntriesProcessedSuccess" type="integer" />
  <element name="numEntriesProcessedFailed" type="integer" />
  <element name="msgSummary" type="string" />
  <element name="startDate" type="dateTime" nillable="true" />
  <element name="endDate" type="dateTime" nillable="true" />
  </sequence>
  </complexType>
<complexType name="job">
<sequence>
  <element name="callBackClass" type="string" />
  <element name="jobIdentifier" type="string" />
  <element name="maxRetryCount" type="integer" />
  <element name="timerInfoXml" type="string" />
  <element name="jobType" type="string" />
  </sequence>
  </complexType>
<complexType name="df">
<sequence>
  <element name="confName" type="string" />
  <element name="ownerName" type="string" />
  </sequence>
  </complexType>
<complexType name="searchCriteria">
<sequence>
  <element name="callbackClass" type="string" />
  <element name="jobIdentifier" type="string" />
  <element name="status" type="string" />
  <element name="submittedBy" type="string" />
  <element name="startDate" type="dateTime" nillable="true" />
  </sequence>
  </complexType>
<!-- scheduleJob Request Type-->
<element name="scheduleJob">
<complexType>
<sequence>
  <element name="job" type="tns:job" />
  </sequence>
  </complexType>
  </element>
<!-- scheduleJob Response Type-->
  <element name="scheduleJobResponse" type="tns:wSJobStatusResult" />
<!-- searchJobs Request Type-->
```

Code 12-1: Web Services WSDL (Continued)

```
<element name="searchJobs">
<complexType>
<sequence>
  <element name="searchCriteria" type="tns:searchCriteria" />
  </sequence>
  </complexType>
  </element>
<!-- searchJobs Response Type-->
<element name="searchJobsResponse">
<complexType>
<sequence>
  <element name="wSJobStatusResult" type="tns:wSJobStatusResult"
nillable="true" minOccurs="0" maxOccurs="unbounded" />
  </sequence>
  </complexType>
  </element>
<!-- getJobStatus Request Type-->
<element name="getJobStatus">
<complexType>
<sequence>
  <element name="jobId" type="long" nillable="false" />
  </sequence>
  </complexType>
  </element>
<!-- getJobStatus Response Type-->
  <element name="getJobStatusResponse" type="tns:wSJobStatusResult" />
<!-- cancelJob Request Type-->
<element name="cancelJob">
<complexType>
<sequence>
  <element name="jobId" type="long" nillable="false" />
  </sequence>
  </complexType>
  </element>
<!-- cancelJob Response Type-->
  <element name="cancelJobResponse" type="tns:wSJobStatusResult" />
<!-- scheduleDataFlow Request Type-->
<element name="scheduleDataFlow">
<complexType>
<sequence>
  <element name="df" type="tns:df" />
  </sequence>
  </complexType>
  </element>
<!-- scheduleDataFlow Response Type-->
  <element name="scheduleDataFlowResponse"
type="tns:wSDataFlowStatusResult" />
<!-- getDataFlowStatus Request Type-->
<element name="getDataFlowStatus">
```

Code 12-1: Web Services WSDL (Continued)

```
<complexType>
<sequence>
  <element name="dfRunId" type="long" nillable="false" />
  </sequence>
  </complexType>
  </element>
<!-- getDataFlowStatus Response Type-->
  <element name="getDataFlowStatusResponse"
type="tns:wSDataFlowStatusResult" />
<!-- searchDataFlows Request Type-->
<element name="searchDataFlows">
<complexType>
<sequence>
  <element name="dfSearchCriteria" type="tns:dfSearchCriteria" />
  </sequence>
  </complexType>
  </element>
<!-- searchDataFlows Response Type-->
<element name="searchDataFlowsResponse">
<complexType>
<sequence>
  <element name="wSDataFlowStatusResult"
type="tns:wSDataFlowStatusResult" nillable="true" minOccurs="0"
maxOccurs="unbounded" />
  </sequence>
  </complexType>
  </element>
  </schema>
```

12.10 The Job Web Service WSDL

Code 12-2: Job Web Service WSDL

```
<definitions name="MnJobWebSvc"
             targetNamespace="http://jobws.intg.modeln.com"
             xmlns:tns="http://jobws.intg.modeln.com"
             xmlns="http://schemas.xmlsoap.org/wsdl/"
             xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
             xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
             xmlns:ns1="http://jobws.intg.modeln.com/types">

    <types>
        <schema targetNamespace="http://jobws.intg.modeln.com"
                xmlns="http://www.w3.org/2001/XMLSchema">

            <import id="ns1"
                    schemaLocation="schema/MnJobWebSvc.xsd"
```

Code 12-2: Job Web Service WSDL (Continued)

```
                    namespace="http://jobws.intg.modeln.com/types"/>

        </schema>
    </types>



    <!--message Request-->
    <message name="getDataFlowStatusRequest">
        <part name="parameters" element="ns1:getDataFlowStatus"/>
    </message>

    <!--message Response-->
    <message name="getDataFlowStatusResponse">
       <part name="parameters" element="ns1:getDataFlowStatusResponse"/
>
    </message>



    <!--message Request-->
    <message name="searchDataFlowsRequest">
        <part name="parameters" element="ns1:searchDataFlows"/>
    </message>

    <!--message Response-->
    <message name="searchDataFlowsResponse">
        <part name="parameters" element="ns1:searchDataFlowsResponse"/>
    </message>



    <!--message Request-->
    <message name="getJobStatusRequest">
        <part name="parameters" element="ns1:getJobStatus"/>
    </message>

    <!--message Response-->
    <message name="getJobStatusResponse">
        <part name="parameters" element="ns1:getJobStatusResponse"/>
    </message>



    <!--message Request-->
    <message name="scheduleJobRequest">
        <part name="parameters" element="ns1:scheduleJob"/>
    </message>
```

Code 12-2: Job Web Service WSDL (Continued)

```
<!--message Response-->
<message name="scheduleJobResponse">
    <part name="parameters" element="ns1:scheduleJobResponse"/>
</message>



<!--message Request-->
<message name="cancelJobRequest">
    <part name="parameters" element="ns1:cancelJob"/>
</message>

<!--message Response-->
<message name="cancelJobResponse">
    <part name="parameters" element="ns1:cancelJobResponse"/>
</message>



<!--message Request-->
<message name="searchJobsRequest">
    <part name="parameters" element="ns1:searchJobs"/>
</message>

<!--message Response-->
<message name="searchJobsResponse">
    <part name="parameters" element="ns1:searchJobsResponse"/>
</message>



<!--message Request-->
<message name="scheduleDataFlowRequest">
    <part name="parameters" element="ns1:scheduleDataFlow"/>
</message>

<!--message Response-->
<message name="scheduleDataFlowResponse">
   <part name="parameters" element="ns1:scheduleDataFlowResponse"/>
</message>



<portType name="MnJobWebSvcPortType">

    <!--operations-->
```

Code 12-2: Job Web Service WSDL (Continued)

```
        <operation name="getDataFlowStatus">
            <input message="tns:getDataFlowStatusRequest"/>
            <output message="tns:getDataFlowStatusResponse"/>
        </operation>

        <operation name="searchDataFlows">
            <input message="tns:searchDataFlowsRequest"/>
            <output message="tns:searchDataFlowsResponse"/>
        </operation>

        <operation name="getJobStatus">
            <input message="tns:getJobStatusRequest"/>
            <output message="tns:getJobStatusResponse"/>
        </operation>

        <operation name="scheduleJob">
            <input message="tns:scheduleJobRequest"/>
            <output message="tns:scheduleJobResponse"/>
        </operation>

        <operation name="cancelJob">
            <input message="tns:cancelJobRequest"/>
            <output message="tns:cancelJobResponse"/>
        </operation>

        <operation name="searchJobs">
            <input message="tns:searchJobsRequest"/>
            <output message="tns:searchJobsResponse"/>
        </operation>

        <operation name="scheduleDataFlow">
            <input message="tns:scheduleDataFlowRequest"/>
            <output message="tns:scheduleDataFlowResponse"/>
        </operation>


    </portType>

    <binding name="MnJobWebSvcBinding" type="tns:MnJobWebSvcPortType">
        <soap:binding style="document" transport="http://
schemas.xmlsoap.org/soap/http"/>

        <!--operations-->

         <operation name="getDataFlowStatus">
            <soap:operation soapAction="http://jobws.intg.modeln.com/
types"/>
            <input>
                <soap:body use="literal"/>
```

Code 12-2: Job Web Service WSDL (Continued)

```
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>

         <operation name="searchDataFlows">
            <soap:operation soapAction="http://jobws.intg.modeln.com/
types"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>

         <operation name="getJobStatus">
            <soap:operation soapAction="http://jobws.intg.modeln.com/
types"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>

         <operation name="scheduleJob">
            <soap:operation soapAction="http://jobws.intg.modeln.com/
types"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>

         <operation name="cancelJob">
            <soap:operation soapAction="http://jobws.intg.modeln.com/
types"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>
```

Code 12-2: Job Web Service WSDL (Continued)

```
         <operation name="searchJobs">
            <soap:operation soapAction="http://jobws.intg.modeln.com/
types"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>

         <operation name="scheduleDataFlow">
            <soap:operation soapAction="http://jobws.intg.modeln.com/
types"/>
            <input>
                <soap:body use="literal"/>
            </input>
            <output>
                <soap:body use="literal"/>
            </output>
        </operation>



    </binding>

    <service name="MnJobWebSvcService">
        <port name="MnJobWebSvcPort" binding="tns:MnJobWebSvcBinding">
            <soap:address location="REPLACE_WITH_ACTUAL_URL"/>
        </port>
    </service>

</definitions>
```

12.11 The Reporting Web Services WSDL

Code 12-3: Reporting Web Services WSDL

```
<?xml version="1.0" encoding="UTF-8"?><!-- Published by JAX-WS RI at
http://jax-ws.dev.java.net. RI's version is hudson-jaxws-2.1.x-nightly-
push-402. --><definitions xmlns:tns="http://reporting.ws.modeln.com"
xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:wsdl="http://
schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/
soap/" xmlns:ns1="http://reporting.ws.modeln.com/types"
name="MnScheduleReportWebSvc" targetNamespace="http://
reporting.ws.modeln.com">


    <types>
```

Code 12-3: Reporting Web Services WSDL (Continued)

```
        <schema xmlns="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://reporting.ws.modeln.com">


            <import id="ns1" schemaLocation="http://pdvhuynh:8049/
webservices/app/MnScheduleReportWebSvc?xsd=1" namespace="http://
reporting.ws.modeln.com/types"></import>


        </schema>
    </types>




    <!--message Request-->
    <message name="scheduleReportRequest">
        <part name="parameters" element="ns1:scheduleReport"></part>
    </message>

    <!--message Response-->
    <message name="scheduleReportResponse">
        <part name="parameters" element="ns1:scheduleReportResponse"></
part>
    </message>




    <!--message Request-->
    <message name="searchReportsRequest">
        <part name="parameters" element="ns1:searchReports"></part>
    </message>

    <!--message Response-->
    <message name="searchReportsResponse">
        <part name="parameters" element="ns1:searchReportsResponse"></
part>
    </message>




    <!--message Request-->
    <message name="cancelReportRequest">
        <part name="parameters" element="ns1:cancelReport"></part>
    </message>

    <!--message Response-->
    <message name="cancelReportResponse">
        <part name="parameters" element="ns1:cancelReportResponse"></
part>
    </message>
```

Code 12-3: Reporting Web Services WSDL (Continued)

```
    <!--message Request-->
    <message name="getReportStatusRequest">
        <part name="parameters" element="ns1:getReportStatus"></part>
    </message>

    <!--message Response-->
    <message name="getReportStatusResponse">
       <part name="parameters" element="ns1:getReportStatusResponse"></
part>
    </message>



    <portType name="MnScheduleReportWebSvcPortType">

        <!--operations-->

        <operation name="scheduleReport">
            <input message="tns:scheduleReportRequest"></input>
            <output message="tns:scheduleReportResponse"></output>
        </operation>

        <operation name="searchReports">
            <input message="tns:searchReportsRequest"></input>
            <output message="tns:searchReportsResponse"></output>
        </operation>

        <operation name="cancelReport">
            <input message="tns:cancelReportRequest"></input>
            <output message="tns:cancelReportResponse"></output>
        </operation>

        <operation name="getReportStatus">
            <input message="tns:getReportStatusRequest"></input>
            <output message="tns:getReportStatusResponse"></output>
        </operation>


    </portType>

    <binding name="MnScheduleReportWebSvcBinding"
type="tns:MnScheduleReportWebSvcPortType">
        <soap:binding style="document" transport="http://
schemas.xmlsoap.org/soap/http"></soap:binding>

        <!--operations-->

         <operation name="scheduleReport">
```

Code 12-3: Reporting Web Services WSDL (Continued)

```
            <soap:operation soapAction="http://reporting.ws.modeln.com/
types"></soap:operation>
            <input>
                <soap:body use="literal"></soap:body>
            </input>
            <output>
                <soap:body use="literal"></soap:body>
            </output>
        </operation>

         <operation name="searchReports">
            <soap:operation soapAction="http://reporting.ws.modeln.com/
types"></soap:operation>
            <input>
                <soap:body use="literal"></soap:body>
            </input>
            <output>
                <soap:body use="literal"></soap:body>
            </output>
        </operation>

         <operation name="cancelReport">
            <soap:operation soapAction="http://reporting.ws.modeln.com/
types"></soap:operation>
            <input>
                <soap:body use="literal"></soap:body>
            </input>
            <output>
                <soap:body use="literal"></soap:body>
            </output>
        </operation>

         <operation name="getReportStatus">
            <soap:operation soapAction="http://reporting.ws.modeln.com/
types"></soap:operation>
            <input>
                <soap:body use="literal"></soap:body>
            </input>
            <output>
                <soap:body use="literal"></soap:body>
            </output>
        </operation>




    </binding>

    <service name="MnScheduleReportWebSvcService">
        <port name="MnScheduleReportWebSvcPort"
binding="tns:MnScheduleReportWebSvcBinding">
```

Code 12-3: Reporting Web Services WSDL (Continued)

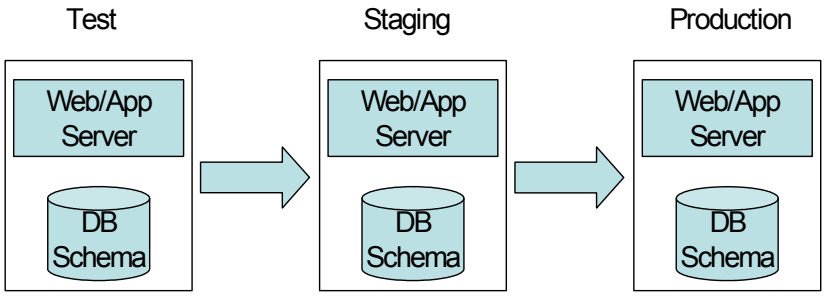
```
            <soap:address location="http://pdvhuynh:8049/webservices/
app/MnScheduleReportWebSvc"></soap:address>
        </port>
    </service>


</definitions>
```

A Promotions

This section covers the deployment promotion process. Promotion means to deploy the same update to multiple environments at one time with the goal of eventually updating the production environment.

Following are some examples of environment patterns:

- Staging > Production
- Test > Staging > Production
- Dev > Test > Staging > Production



A.1 Single Promotion Path

The deployment promotion process for a single promotion path consists of the following steps:

1. Define the required promotion path.

   For example, Test > Staging > Production

2. Determine whether the Test environment can be skipped when an emergency deployment is needed.

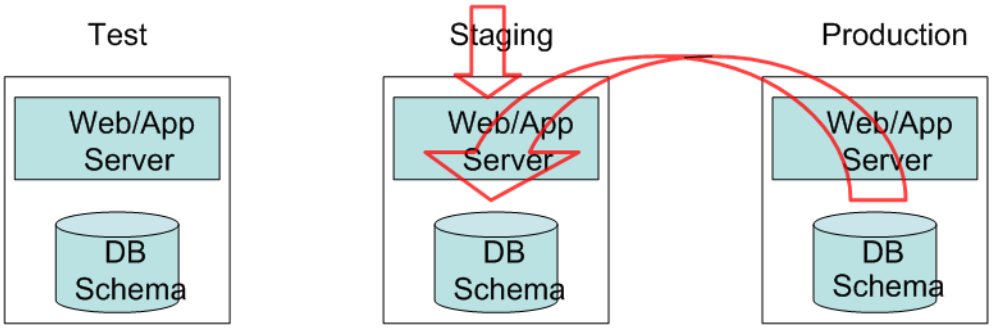
3. Deploy the new update to the Test environment.

   a. Remove the current application from the Test environment.

   b. Sync the Production database to the Test database if needed.

   c. Deploy the new update to the Test environment.

   The following diagram illustrates this process.



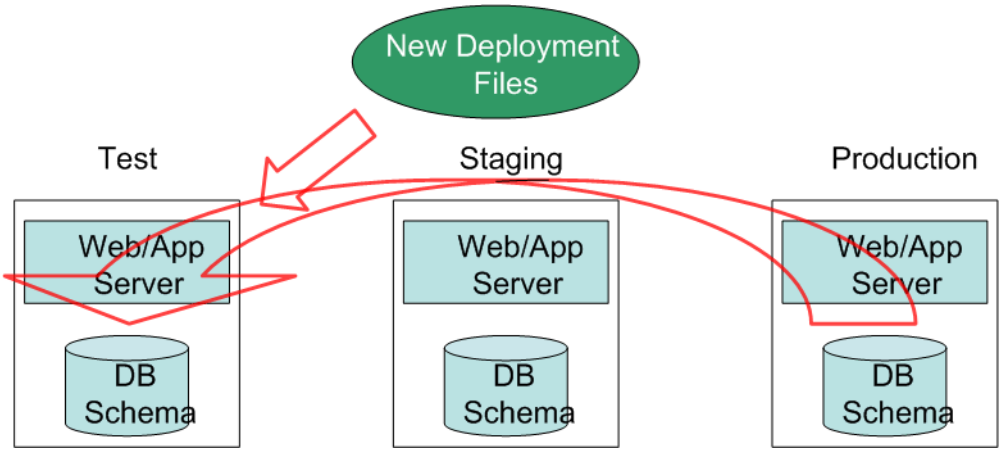
4. Deploy the new update to the Staging environment.

   a. Remove the current application from the Staging environment.

   b. Sync the Production database to the Staging database (export, import, data clean).

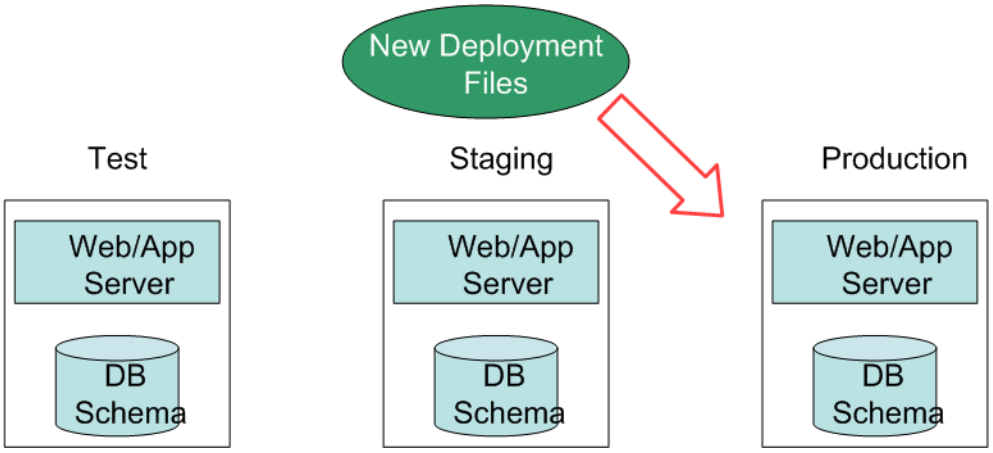
   c. Deploy the new update to the Staging environment.

   The following diagram illustrates this process.



5. Deploy the new update to the Production environment.
    a. Remove the current application from the Production environment.
    b. Back up the existing production data.
    c. Deploy the new update to the Production environment.

The following diagram illustrates this process.

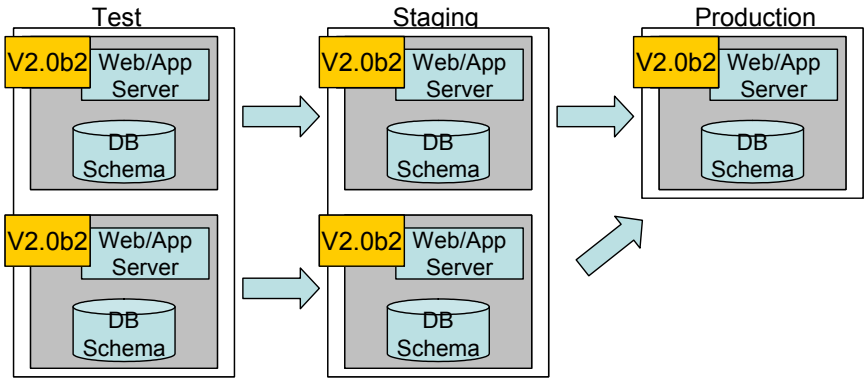


A.2 Multiple Promotion Paths

Multiple instances of the application on the same environment are often necessary to facilitate parallel testing. For example, you may want to test data loading and perform user testing at the same time. Multiple instances are also required to test multiple versions of the application. For example, you may want to test Model N 5.0 P3 and Model N 5.1 simultaneously.

When you promote a current release from Test1 to Staging1 to Production, Test1 and Staging1 will contain the same point release as production. If you deploy a new patch to Test1 and then to Staging1, you can deploy it to Production once testing is done. Once, Test1, Staging1, and Production all have the same point and or patch release, they are in sync.

If you then want to deploy a new release, you would set up another Test environment (Test2) and another Staging environment (Staging2). After setting up Test2 and Staging2, you can deploy the new release candidate to Test2 and then to Staging2. Once the new release is tested and released, it is deployed to Production and then deployed to Test1 and Staging1. Now, all five environments are in sync as illustrated in the following diagram:



B Performance Monitoring Application SQL

This appendix provides information on how to use SQL to retrieve and analyze data using the Performance Monitoring Application (PMA). For information on how to use Cognos Reports to analyze PMA data, see Using Cognos Reports to Analyze PMA Data.

B.1 PMA Tables

The following tables are used by the Performance Monitoring Application:

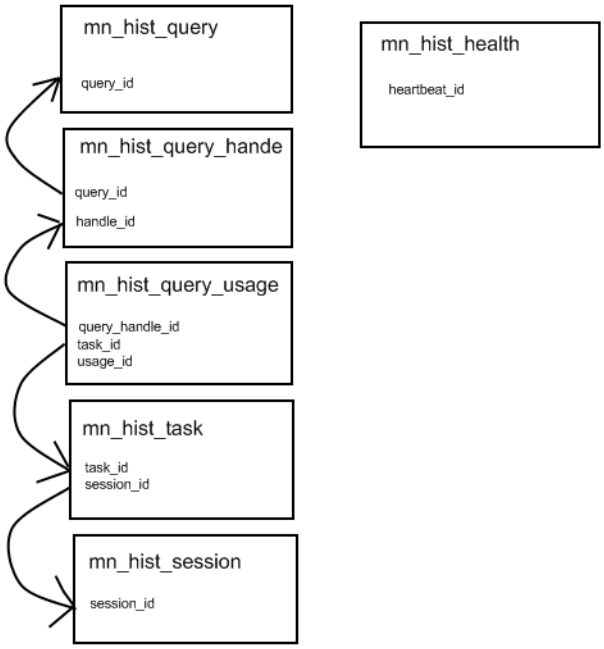
Table B-1: Tables Used by the PMA

Database Table	Description
`mn_hist_session`	Table used to track user sessions.
`mn_hist_task`	Table used to track user requests.
`mn_hist_query_usage`	Table used to track problem queries associated with a historical task.

Table B-1: Tables Used by the PMA (Continued)

Database Table	Description
mn_hist_query_handle	Table used as a buffer between usage and query so that there is no need to ever run a scan on a table with a CLOB, and instead only a need to look by primary key.
mn_hist_query	Table used to track historical query SQL.
mn_hist_health	Table used to track the overall system health including memory usage, number of active user sessions, database connection usages, number of back-end tasks, and different types of queries run.

The following diagram shows the PMA tables, which columns control the relationships, and which direction the references go:



The data analyst retrieves and analyzes the data directly from the database using SQL. This section is intended to be an aid to the analyst, providing an overview of what is in the database, suggestions on what to look for, how to determine the causes of any issues detected, and some helpful SQL queries to use as a starting point.

The central table for analysis is `mn_hist_task`. It contains an entry for every task the server undertakes, which can be a backend thread, a backend command, or a user request. Each task saves the details of what it has done into a row of `mn_hist_task`. To flesh out the data in task, there is `mn_hist_session`, which links together the user request tasks into sessions, and `mn_hist_query_usage`, which contains details on queries that ran repeatedly or slowly inside a single task.

B.2 Performance Report Query

You can get a basic overview of the state of the system from the following query:

Code B-1: Performance Report Query

```
select count (*) requests, sum(server_time) total_server_time,
round(sum(server_time)/count(*)) average_server_time

, sum(components) components, sum(component_time) component_time,
round(sum(component_time)/count(*)) average_comp_time

, sum(pobjs) pobjs, sum(pobjs_time) pobjs_time, round(sum(pobjs_time)/
count(*)) average_pobjs_time

, sum(query_count) queries, sum(query_time) query_time,
round(sum(query_time)/count(*)) average_query_time

from mn_hist_task where Thread_Type like 'UserRequest';
```

The output of this query appears as follows:

Table B-2: Performance Report Query Output

REQUESTS	107
TOTAL_SERVER_TIME	429479
AVERAGE_SERVER_TIME	4014
COMPONENTS	172918
COMPONENT_TIME	148942
AVERAGE_COMP_TIME	1392
POBJS	21025
POBJS_TIME	10677
AVERAGE_POBJS_TIME	100
QUERIES	10241
QUERY_TIME	20774
AVERAGE_QUERY_TIME	194

B.3 Finding Slow Queries

To find the slowest queries that were run by the application, use the following query and make sure to specify a value for `avg_time` in milliseconds:

Code B-2: Finding Slow Queries

```
select usage.*, query.sql from

(select sum (time), sum (uses), sum(time)/sum(uses) avg_time, count(*)
cnt_tasks, query_handle_id from mn_hist_query_usage usage

group by query_handle_id) usage, mn_hist_query_handle handle,
mn_hist_query query

where usage.query_handle_id=handle.handle_id

and query.query_id=handle.query_id

and avg_time > ?

order by avg_time desc;
```

Table B-3: Finding Slow Queries

SUM(TIME)	343	958	851	320
SUM(USES)	38	180	223	96
AVG_TIME	9.03	5.32	3.82	3.33
CNT_TASKS	2	3	12	6
QUERY_ HANDLE_ID	204	225	205	211
SUBSTR (QUERY.SQL, 1,30)	SELECT mn_mcd_ claim_line. date	SELECT mn_ structured_ contract	SELECT mn_member. ver_num, mn_m	SELECT mn_mcd_ claim_line. date

B.4 Finding Tasks that Used a Slow Query

To find the tasks that used a given slow query, use the handle_id field as follows:

Code B-3: Handle_ID Field

```
select task.task_id, task.thread_type,  task.member_name,
task.resource_key, task.id request_number, task.server_time,
task.query_time, hist_usage.time this_query, q.sql

from mn_hist_task task, mn_hist_query q, mn_hist_query_usage
hist_usage, mn_hist_query_handle handle where

hist_usage.query_handle_id = handle.handle_id and

hist_usage.task_id = task.task_id and

q.query_id=handle.query_id and

handle.handle_id = ?;
```

Table B-4: Handle_ID Field

TASK_ID	15771	15784	15801
THREAD_TYPE	UserRequest	UserRequest	UserRequest
MEMBER_NAME	sadm	sadm	sadm
RESOURCE_KEY	root.app.root.main.bodyComp.documentFrame.noOp	root.app.root.main.bodyComp.documentFrame.noOp	root.app.root.main.bodyComp.documentFrame.noOp
REQ_NUM	2	4	4
SERVER_TIME	2264	6772	1733
QUERY_TIME	909	665	521
THIS_QUERY	255	466	237
SUBSTR(Q.SQL,1,30)	SELECT mn_structured_contract	SELECT mn_structured_contract	SELECT mn_structured_contract

B.5 Finding User Session Details

To get details on sessions and how much server resources they consumed, use the following query:

Code B-4: Finding User Session Details

```
select sess.member_name, sess.session_id, task.requests,
(nvl(sess.end_date, sess.last_request_date) - sess.start_date)
duration, task.svr_time, task.comps, task.comp_time, task.objects,
task.obj_time, task.queries, task.query_time, sess.start_date

from mn_hist_session sess, (select count(*) requests, session_id,
sum(server_time) svr_time, sum(components) comps, sum(component_time)
comp_time, sum(pobjs) objects, sum(pobjs_time) obj_time,
sum(query_count) queries, sum(query_time) query_time from mn_hist_task
group by session_id) task

where sess.session_id = task.session_id

order by svr_time desc;
```

Table B-5: Finding User Session Details

MEMBER_NAME	sadm	sadm	sadm
SESSION_ID	1821	1801	1822
REQUESTS	20	4	6
DURATION	+000000000 01:31:11.523	+000000000 00:34:03.798	+000000000 01:09:08.372
SVR_TIME	87466	61145	7352
COMPS	15571	9358	7835
COMP_TIME	18002	14604	5460
OBJECTS	8993	1896	1720
OBJ_TIME	1648	492	393
QUERIES	3428	850	611
QUERY_TIME	4481	2335	1029
START_DATE	30-JUL-07 01.21.30.630 PM	30-JUL-07 10.44.29.098 AM	30-JUL-07 02.36.03.911 PM

B.6 Retrieving Session Details

You can use session IDs from the details that you get to retrieve details on the sessions that are troublesome as follows:

Code B-5: Retrieving Session Details

```
select task_id, id request, server_time, query_time, resource_key from
mn_hist_task where task_id in (select task_id from mn_hist_task where
session_id = ?) order by task_id;
```

Table B-6: Retrieving Session Details

TASK_ID	REQUEST	SERVER_TIME	QUERY_TIME	RESOURCE_KEY
15781	1	49788	342	root.app.root.main.noOp
15782	2	95	4	root.app.root.main.noOp
15783	3	8866	450	root.app.root.main.bodyComp.login.btLogin.submitAction
15784	4	6772	665	root.app.root.main.bodyComp.documentFrame.noOp
15785	5	1715	261	root.app.root.main.bodyComp.navMenuBar.contractsMenu.rightMenu.Contracts.submitAction
15786	6	3226	110	root.app.root.main.bodyComp.documentFrame.contracts.searcher.viewContainer.contract.BODY-0-0
15787	7	1302	199	root.app.root.main.bodyComp.offer.computePricingAlertsButton.submitAction
15788	8	788	85	root.app.root.main.bodyComp.browserControls.refresh.submitAction

Table B-6: Retrieving Session Details (Continued)

TASK_ID	REQUEST	SERVER_TIME	QUERY_TIME	RESOURCE_KEY
15789	9	6223	1409	root.app.root.main.bodyComp.navMenuBar.productsMenu.rightMenu.Catalog.Global Catalog View.
15790	10	1458	123	root.app.root.main.bodyComp.navMenuBar.contractsMenu.rightMenu.Contracts.submitAction
15791	11	196	17	root.app.root.main.keepAliveLazyRequestComp.requestComp.submitAction
15792	12	917	90	root.app.root.main.bodyComp.documentFrame.contracts.searcher.viewContainer.contract.BODY-0-
15795	13	373	34	root.app.root.main.bodyComp.clientRefreshMgtComp.requestComp.submitAction
15796	14	253	18	root.app.root.main.bodyComp.clientRefreshMgtComp.requestComp.submitAction
15797	15	245	18	root.app.root.main.bodyComp.clientRefreshMgtComp.requestComp.submitAction
15798	16	1492	285	root.app.root.main.bodyComp.documentFrame.contracts.offer.saveButton.submitAction
15799	17	1297	166	root.app.root.main.bodyComp.documentFrame.contracts.offer.computePricingAlertsButton.submitA

Table B-6: Retrieving Session Details (Continued)

TASK_ID	REQUEST	SERVER_TIME	QUERY_TIME	RESOURCE_KEY
15804	18	323	17	root.app.root.main.bodyComp.clientRefreshMgtComp.requestComp.submitAction
15806	19	1035	169	root.app.root.main.bodyComp.navMenuBar.customersMenu.rightMenu.Accounts.submitAction
15807	20	1102	19	root.app.root.main.bodyComp.globalMenuBar.fileMenu.logout.submitAction

B.7 Using the Usage Report Query

To find the time spent in various sections of the application by `RESOURCE_KEY`, use the Usage Report query as follows:

Code B-6: Usage Report Query

```
select count(*) requests, sum(server_time) time,
round(sum(server_time)/count(*)) average_time, sum(query_time)
query_time, round(sum(query_time)/count(*)) avg_query_time,
resource_key

from mn_hist_task where Thread_Type like 'UserRequest'

group by resource_key;
```

Table B-7: Usage Report Query

REQUESTS	TIME	AVERAGE_ TIME	QUERY_ TIME	AVG_ QUERY_ TIME	RESOURCE_KEY
1	788	788	85	85	root.app.root.main.bodyComp.browserControls.refresh.submitAction

Table B-7: Usage Report Query (Continued)

REQUESTS	TIME	AVERAGE_ TIME	QUERY_ TIME	AVG_ QUERY_ TIME	RESOURCE_KEY
6	1773	296	166	28	root.app.root.main.bodyComp.clientRefreshMgtComp.requestComp.submitAction
2	2599	1300	365	183	root.app.root.main.bodyComp.documentFrame.contracts.offer.computePricingAlertsButton.submitAction
1	1492	1492	285	285	root.app.root.main.bodyComp.documentFrame.contracts.offer.saveButton.submitAction
2	4143	2072	200	100	root.app.root.main.bodyComp.documentFrame.contracts.searcher.viewContainer.contract.BODY-0-0.submitAction
1	6881	6881	224	224	root.app.root.main.bodyComp.documentFrame.home.docsTable.BODY-0-0.submitAction
4	11467	2867	2114	529	root.app.root.main.bodyComp.documentFrame.noOp
1	1102	1102	19	19	root.app.root.main.bodyComp.globalMenuBar.fileMenu.logout.submitAction

Table B-7: Usage Report Query (Continued)

REQUESTS	TIME	AVERAGE_ TIME	QUERY_ TIME	AVG_ QUERY_ TIME	RESOURCE_KEY
2	13238	6619	843	422	root.app.root.main.bodyComp.login. btLogin.submitAction
2	3173	1587	384	192	root.app.root.main.bodyComp.navMenuBar.contractsMenu.rightMenu.Contracts.submitAction
1	1035	1035	169	169	root.app.root.main.bodyComp.navMenuBar.customersMenu.rightMenu.Accounts.submitAction
1	580	580	18	18	root.app.root.main.bodyComp.navMenuBar.pricingMenu.rightMenu.Price Lookup.submitAction
1	6223	6223	1409	1409	root.app.root.main.bodyComp.navMenuBar.productsMenu.rightMenu.Catalog.Global Catalog View.submitAction
2	284	142	35	18	root.app.root.main.keepAliveLazy RequestComp.requestComp.submitAction
9	107486	11943	1682	187	root.app.root.main.noOp

B.8 Finding Tasks Running Between Two Times

The following is a query to find tasks that were running between two times, which are specified explicitly in the query:

Code B-7: Finding Tasks Running Between Two Times

```
select task_id, thread_type, server_time, task_start_date,
(task_start_date + server_time/(1000*60*60*24)) from mn_hist_task where
(task_start_date + server_time/(1000*60*60*24)) < to_date('07/25/2007
15:21:00', 'MM/dd/yyyy HH24:MI:SS') and task_start_date > to_date('07/
25/2007 15:20:00', 'MM/dd/yyyy HH24:MI:SS')
```

B.9 Finding Slowest Query Usages

The following is a query to find the slowest query usages by average run time with a cutoff to be specified as a parameter. This query has one parameter (the average time in milliseconds a query must take to show up in the list), and four "like" clauses which are given as match-everything wildcards. You can use these clauses to narrow the desired range of tasks returned, by area of the application, user, thread type, and query text:

Code B-8: Finding Slowest Query Usages

```
select task.task_id, task.thread_type,  task.member_name member,
task.resource_key, task.id req_num, task.server_time svr_time,
task.query_time q_time, hist_usage.time this_q_time,
round(aggregation.avg, 1) avg, handle.query_id q_id, q.sql

from mn_hist_task task, mn_hist_query q, mn_hist_query_usage
hist_usage, mn_hist_query_handle handle, (select query_handle_id,
sum(time)/sum(uses) avg from mn_hist_query_usage group by
query_handle_id) aggregation where

hist_usage.query_handle_id = handle.handle_id and

hist_usage.task_id = task.task_id and

q.query_id = handle.query_id and

handle.handle_id in aggregation.query_handle_id and

aggregation.avg > ?

and q.sql like '%'

and task.resource_key like '%'

and task.member_name like '%'

and task.thread_type like '%'

order by aggregation.avg desc;
```

B.10 Finding User Requests

Use the following query to find user requests:

Code B-9: Finding User Requests

```
select sess.member_name, task.session_id, task.id request,
task.resource_key, task.server_time, task.query_time from mn_hist_task
task, mn_hist_session sess

where

task.Thread_Type like 'UserRequest'

and task.session_id = sess.session_id;
```

B.11 Finding the Latest Tasks

Use the following query to find tasks for the last X number of minutes:

Code B-10: Finding the Latest Tasks

```
select task_id, thread_type, server_time, task_start_date,
resource_key, session_id

from mn_hist_task, (select sysdate ct from dual) curr_time where
task_start_date + (:1/(60*24)) > curr_time.ct;
```

B.12 Finding Active User Sessions

Use the following query to find all active user sessions:

Code B-11: Finding Active User Sessions

```
select * from mn_hist_session where end_date is not null
```

B.13 Clearing History Data

Use the following query to delete history data:

Code B-12: Clearing History Data

```
delete from mn_hist_query_usage;

delete from mn_hist_query_handle;

delete from mn_hist_query;

delete from mn_hist_task;

delete from mn_hist_session;

commit;
```

C Monitoring Guidelines

This appendix describes the types of metrics and mechanisms for monitoring the Model N system. Once a system is in production, those responsible for managing that system, typically called the run team, need regularly review metrics that can provide a health indication. These metrics must be measurable through automated means that do not significantly impact the performance of the system.

It is recommend that the following measurements be made periodically. Included with the measurement definition is a description of how the measurement can be made. When the data already resides in Model N tables, a query is described that can be used to provide the measurement. Otherwise, recommendations are given on how the data can be gathered.

C.1 Assumptions

The following assumptions are made regarding the monitoring guidelines:

- These metrics are monitored frequently enough to detect changes in the system before they become a serious problem, but with a large enough period to provide statistical significance to the results.

  As a practical matter, the period should not be too frequent to provide a significant burden, so a period in the range of a week to a month seems reasonable.

Note: For the metrics in this document, it is assumed a period of a week is used. In the queries, you can identify this because the `task_start_date` column will have a restriction between `SYSTIMESTAMP - 7` and `SYSTIMESTAMP`.

- The Performance Monitoring Application (PMA) is available and enabled.

  PMA Task Purging

  Since the PMA purges task data at regular intervals, the PMA must be configured to hold the task data at least as long at the monitoring period. The default PMA purging time is seven days.

  Resource Keys

  For each task, which could be a UI request or background activity, a resource key is provided to identify what that task is doing. For the UI request, the resource key identifies the location within the user interface as well as the specific action taken. For background activities, the resource key identifies the timer or command being run.

- All times are in seconds and rounded to two decimal places unless explicitly stated otherwise.

C.2 Monitoring User Response Times

C.2.1 Response Time Levels

To provide a summary of the performance of the application for the users, the response time percentiles should be measured. A percentile indicates the percentage of responses that were lower than the given value. For example, if the 90th percentile value was 1.25 seconds, this means that 90% of the requests had a response time of 1.25 seconds or less.

A useful set of numbers is the 50th (median), 90th, and 100th (maximum) percentiles in the past period.

C.2.1.1 Query

The following query lists the number of user requests, 50th, 90th, and 100th percentiles over the past week.

Code C-1: Query for Response Time Levels

```
SELECT
  COUNT(*) AS cnt,
ROUND(PERCENTILE_CONT(0.50)
    WITHIN GROUP (ORDER BY server_time ASC)/1000, 2) AS pct50,
ROUND(PERCENTILE_CONT(0.90)
    WITHIN GROUP (ORDER BY server_time ASC)/1000, 2) AS pct90,
  ROUND(MAX(server_time)/1000, 2) AS pct100
FROM
  mn_hist_task
WHERE
```

Code C-1: Query for Response Time Levels (Continued)

```
  thread_type = 'UserRequest'
  AND task_start_date BETWEEN SYSTIMESTAMP - 7 AND SYSTIMESTAMP
;
```

C.2.2 Top Response Times by User Action

The longest user requests may indicate performance problems in the application. The top user requests over a given threshold time should be measured for the given monitoring period.

C.2.2.1 Query

The following query lists the longest user requests greater than 10 seconds during the past week.

Code C-2: Query for Top Response Times by User Action with product_area

```
SELECT
  *
FROM
 (SELECT
    REGEXP_REPLACE(
      REGEXP_REPLACE(
        REGEXP_REPLACE(
          resource_key,
          '(root.app.)?root.main.bodyComp(.documentFrame)?.',''),
        '^([^\.]+)\..*', '\1'),
      '(.*)\[.*', '\1', 1, 1, 'n')     AS product_area,
    resource_key                       AS resource_key,
    COUNT(*)                           AS cnt,
    ROUND(MIN(server_time)/1000, 2)    AS min_server_time,
    ROUND(MAX(server_time)/1000, 2)    AS max_server_time,
    ROUND(AVG(server_time)/1000, 2)    AS avg_server_time,
    ROUND(STDDEV(server_time)/1000, 2) AS stddev_server_time
  FROM
    mn_hist_task
  WHERE
    thread_type = 'UserRequest'
    AND task_start_date BETWEEN SYSTIMESTAMP - 7 AND SYSTIMESTAMP
  GROUP BY
    resource_key
  ORDER BY
    ROUND(MAX(server_time)/1000, 2) DESC
  ) temp
WHERE
  max_server_time > 10
;
```

The product_area column provides an indication of which area of the application the request comes from (such as contracts, rebates, or admin). The resource_key provides the specific detail on exactly where in that application the request originates.

C.3 Monitoring Background Activity

Background activity is executed through timers and commands. These activities can be scheduled to be run periodically (such as nightly data flows) or executed immediate as the result of a user action (such as calculate GP workbook).

C.3.1 Response Time Levels by Activity

As with the user requests, the background activity response time levels should be monitored. Since each background activity can have significantly different durations, the response times should be segmented by the activity type (such as GP calculation versus membership publishing).

C.3.1.1 Query

The following query returns the number of runs, 50th, 90th, and 100th percentiles for each background activity over the past week. The results are ordered such that the longest duration activity is at the top.

Code C-3: Query for Response Time Levels by Activity

```
SELECT
  resource_key AS resource_key,
  class        AS class,
  COUNT(*)     AS cnt,
ROUND(PERCENTILE_CONT(0.50)
    WITHIN GROUP (ORDER BY server_time ASC)/1000, 2) AS pct50,
ROUND(PERCENTILE_CONT(0.90)
    WITHIN GROUP (ORDER BY server_time ASC)/1000, 2) AS pct90,
  ROUND(MAX(server_time)/1000, 2) AS pct100
FROM
  mn_hist_task
WHERE
  thread_type IN ('TimerService','Command')
  AND task_start_date BETWEEN SYSTIMESTAMP - 7 AND SYSTIMESTAMP
GROUP BY
  resource_key,
  class
ORDER BY
  ROUND(MAX(server_time)/1000, 2) DESC
;
```

Note: The structure of the resource_key column is different than for UI requests, so the product_area column should not be used here.

C.3.2 Top Response Times by Activity

The longest running background activities may provide insight into where the performance bottlenecks in the system reside. This metric is similar to that for user requests except that the results are segmented by activity.

C.3.2.1 Query

The following query lists the background activities during the past week grouped by activity. Since background activities are generally longer running than user requests, we look only for activities that have a maximum duration of at least 60 seconds. The results are ordered such that the longest duration activity returns first.

Code C-4: Query for Top Response Times by Activity

```
SELECT
  *
FROM
 (SELECT
    resource_key                       AS resource_key,
    class                              AS class,
    COUNT(*)                           AS cnt,
    ROUND(MIN(server_time)/1000, 2)    AS min_server_time,
    ROUND(MAX(server_time)/1000, 2)    AS max_server_time,
    ROUND(AVG(server_time)/1000, 2)    AS avg_server_time,
    ROUND(STDDEV(server_time)/1000, 2) AS stddev_server_time
  FROM
    mn_hist_task
  WHERE
    thread_type IN ('TimerService','Command')
    AND task_start_date BETWEEN SYSTIMESTAMP - 7 AND SYSTIMESTAMP
  GROUP BY
    resource_key,
    class
  ORDER BY
    ROUND(MAX(server_time)/1000, 2) DESC
  ) temp
WHERE
  max_server_time > 60
;
```

C.3.2.2 Response Time Trend

To understand if there are activities in the system that have degraded significantly since the system went live, the response time trend for each background activity is important to monitor. For example, a large difference between the Go-Live and current response times could indicate queries that have slowed down as the volume of data in the system has increased.

Since the PMA does not currently support persisting this information for longer than the purging time period, the Top Response Times by Activity data should be stored per monitoring period (for example, a week) and reviewed for increasing response times.

C.4 Monitoring Application Instances

The application JVM instances should be monitored to identify general issues that may significantly degrade an application node. Potential issues include running out of memory or uneven workload across the cluster.

C.4.1 CPU Utilization Trend

The CPU utilization for each application server should be measured over time. The utilization is the load on the CPU. The common measurements are either the CPU load or the % utilization.

In most production systems, we usually see the CPU utilization under 50% for most periods, but can climb to higher values during short periods of time (for example, when data flows are running).

C.4.1.1 Tools

Most monitoring tools provide some CPU monitoring capability. At Model N, we use an open source tool called internally to monitor our servers. For more information, see http://www.zabbix.com.

C.4.2 JVM Heap Usage Trend

Java applications allocate memory in the JVM heap and not directly from the operating system. Therefore, we should monitor the JVM heap usage and not the memory allocated by the operating system to the JVM itself.

C.4.2.1 Periods of Excessive Garbage Collection Activity

A well-behaved system should show a regular "saw tooth" pattern in terms of memory usage. The "saw tooth" shape indicates that memory is being used up, but periodically being freed and reclaimed by the garbage collector. The peaks of the "saw tooth" pattern are typically near the total heap size, but you should see a significant drop during each GC cycle.

If you get into a low-memory condition, then the valleys in the "saw tooth" will increase as less memory is available to be freed. If this continues too long, then you will see more major GC cycles. Once you get low enough in memory, the JVM will get to the point where it can't free up enough memory and will start continuously issuing major GC cycles.

C.4.2.2 Average and Maximum Memory Usage

One question that is frequently asked of Model N is how many JVMs may be required in any particular deployment. The required JVM heap memory is one of the primary considerations to answer this question. By monitoring the GC activity, over time the average and total memory consumption can be determined. This can be used to determine if more JVMs are required to support the load or if potentially the JVMs are not being used fully.

C.4.2.3 Tools

Some monitoring tools provide Java heap monitoring capability. The JVM support for monitoring is evolving rapidly. Java SE 5 introduced significant monitoring and management support. The `jconsole` application provides a straightforward means for monitoring the JVM heap usage as well as other metrics, such as threads. For details on jconsole, see http://java.sun.com/developer/technicalArticles/J2SE/jconsole.html.

Sun JVM GC Logging

To turn on garbage collection logging within the Sun JVM, use the JVM arguments:

```
-verbose:gc -XX:+PrintGCTimeStamps -XX:+PrintGCDetails
-Xloggc:<logfile>
```

where `<logfile>` is where the GC log entries will go.

IBM JVM GC Logging

To turn on garbage collection logging within the IBM JVM, use the JVM arguments:

```
-verbose:gc
```

and the output goes to either the `native_stderr.log` (Windows/Linux) or `native_stdout.log` (Solaris). This can also be accomplished through the WebSphere management console.

The IBM GC and Memory Visualizer tool supports the new IBM GC logging format. This tool provides a good visualization and supports monitoring for more than just garbage collection. For details on the IBM GC and Memory Visualizer, see http://www.ibm.com/developerworks/java/library/j-ibmtools2/index.html.

C.5 Monitoring the Database Server

C.5.1 CPU Utilization Trend

The CPU utilization for the database server should be measured over time. The utilization is the load on the CPU. The common measurements are either the CPU load or the % utilization.

In most production systems, we usually see the CPU utilization around 50% for most periods, but can climb to higher values resulting from background activities, such as rebate bucketing, price monitoring, membership publishing, GP calculations, Medicaid URA calculations, etc.

C.5.1.1 Tools

Most monitoring tools provide some CPU monitoring capability. At Model N, we use an open source tool called internally to monitor our servers. For more information, see http://www.zabbix.com.

C.5.2 Slowest Queries

The response time for queries provides a good indicator on the overall application performance, since the Model N system relies heavily on the database. Queries can slow down for a variety of reasons. The most common reason is data growth.

The following query lists the slowest queries by average response time. The full SQL string of the query, average response time, and number of executions is provided. The task type indicates whether the query is associated with a UI request (UI) or background activity (BE). A threshold of 10 seconds is given for UI queries and 60 seconds for background activity queries.

Code C-5: Query for Slowest Queries

```
SELECT
  usage.thread_type AS thread_type,
  usage.avg_time    AS avg_time,
  usage.total_time  AS total_time,
  usage.query_count AS query_count,
  usage.task_count  AS task_count,
  query.sql         AS sql_string
FROM
  (SELECT
     ROUND(SUM(u.time) / 1000, 2)                AS total_time,
     SUM(u.uses)                                 AS query_count,
     ROUND(SUM(u.time) / (1000*SUM(u.uses)), 2)  AS avg_time,
     COUNT(*)                                    AS task_count,
     u.query_handle_id                           AS query_handle_id,
     t.thread_type                               AS thread_type
   FROM
     mn_hist_query_usage u
     INNER JOIN mn_hist_task t ON (
       u.task_id = t.task_id
     )
   WHERE
    t.task_start_date BETWEEN SYSTIMESTAMP - 7 AND SYSTIMESTAMP
   GROUP BY
     u.query_handle_id,
     t.thread_type
  ) usage
  INNER JOIN mn_hist_query_handle handle ON (
    usage.query_handle_id = handle.handle_id
  )
  INNER JOIN mn_hist_query query ON (
    query.query_id = handle.query_id
  )
WHERE
  (usage.thread_type = 'UserRequest'
     AND usage.avg_time > 10)
  OR
  (usage.thread_type IN ('TimerService','Command')
     AND usage.avg_time > 60)
```

Code C-5: Query for Slowest Queries (Continued)

```
ORDER BY
  usage.thread_type ASC,
  usage.avg_time DESC
;
```

C.5.3 Top Tables by Size

There are hundreds of tables in the Model N system with a variety of growth patterns. In the past, we have seen unusual growth patterns in customer deployments based on differing usages of the product as well as from customizations separate from the out of the box product. This may also provide information whether some areas may need to adjust the purging frequency (such as timers).

The following query takes a snapshot of all the tables within the Model N schema and includes sizing information (such as average row length) and statistics information (such as when the table was last analyzed). The result set is sorted such that the largest table (defined by the number of rows) is returned first.

Code C-6: Query for Top Tables by Size

```
SELECT
  table_name,
  tablespace_name,
  temporary,
  num_rows,
  blocks,
  avg_row_len,
  sample_size,
  last_analyzed
FROM
  user_tables
ORDER BY
   num_rows DESC NULLS LAST
;
```

A similar query can be used to rank the tables by storage size:

Code C-7: Query for Tables by Storage Size

```
SELECT
  table_name,
  tablespace_name,
  temporary,
  num_rows,
  blocks,
  avg_row_len,
  sample_size,
  last_analyzed
FROM
  user_tables
ORDER BY
```

Code C-7: Query for Tables by Storage Size (Continued)

```
    blocks DESC NULLS LAST
;
```

These queries assume that they are being run within the Model N user (that is, you logged into the database as the same user as the application would use).

These queries should be run periodically and tracked so that a trend can a growth trend can be determined.

C.5.4 Top Indexes by # Rows

As with the Top Tables by # Rows, the indexes should also be monitored for growth.

The following query takes a snapshot of all the indexes within the Model N schema and includes structural information (such as B-Tree Level, # Distinct Keys, Clustering Factor) and statistics information (such as when the index was last analyzed). The result set is sorted such that the largest index (defined by the number of rows) is returned first.

Code C-8: Query for Top Indexes by # Rows

```
SELECT
    index_name,
    table_name,
    tablespace_name,
    blevel,
    distinct_keys,
    clustering_factor,
    leaf_blocks,
    avg_data_blocks_per_key,
    avg_leaf_blocks_per_key,
    num_rows,
    sample_size,
    last_analyzed
FROM
    user_indexes
ORDER BY
    num_rows DESC NULLS LAST
;
```

A similar query can be used to rank the indexes by storage size as determined by the number of leaf blocks that are used:

Code C-9: Query for Top Indexes by Storage Size

```
SELECT
    index_name,
    table_name,
    tablespace_name,
    blevel,
    distinct_keys,
    clustering_factor,
    leaf_blocks,
```

Code C-9: Query for Top Indexes by Storage Size

```
    avg_data_blocks_per_key,
    avg_leaf_blocks_per_key,
    num_rows,
    sample_size,
    last_analyzed
FROM
    user_indexes
ORDER BY
    leaf_blocks DESC NULLS LAST
;
```

These queries assume that they are being run within the Model N user.

Note that if the `blevel` for any index grows beyond a value of three, this may indicate that the index is becoming imbalanced and should be rebuilt.

C.6 Index Monitoring

There are many indexes within the Model N schema that are used for a variety of purposes. The usage of the Model N application may vary from deployment to deployment. Therefore, some indexes may become more or less important based on the specific usage. We recommend monitoring the indexes on a weekly basis.

Note that Oracle always recommends that foreign keys be indexed except in the case when the matching unique or primary key is never updated or deleted.

C.6.1 Oracle 10g

In Oracle 10g, the Automatic Workload Repository (AWR) mechanism provides statistics that include index usage. By default, the AWR collects statistics hourly and preserved for one week. As such, there is no other setup that is required to extract index statistics.

The following query returns the usage of indexes in a given schema:

Code C-10: Query for Usage of Indexes in a Given Schema

```
SELECT
  i.table_name  AS table_name,
p.object_name AS index_name,
p.operation   AS operation,
p.options     AS options,
COUNT(1)      AS usage_count
FROM
dba_hist_sql_plan p
INNER JOIN dba_hist_sqlstat s ON (
  p.sql_id = s.sql_id
)
INNER JOIN dba_indexes I ON (
  p.object_owner = i.owner
  AND p.object_name = i.index_name
)
```

Code C-10: Query for Usage of Indexes in a Given Schema (Continued)

```
WHERE
p.object_owner = '<SCHEMA>'
AND p.operation LIKE '%INDEX%'
GROUP BY
  i.table_name,
p.object_name,
p.operation,
p.options
ORDER BY 1,2,3,4
;
```

Replace <SCHEMA> with the schema owner that contains the indexes to monitor. The options column provides information on how the index was used (such as RANGE SCAN, UNIQUE SCAN). The usage_count column indicates how many times a particular index was used. If an index is used multiple times within a single query, then this count would include the number of times it was referenced.

The following query returns the indexes that were not used in the given monitoring period:

Code C-11: Query for Indexes Not Used in the Given Monitoring Period

```
SELECT
  i.table_name,
  i.index_name
FROM
  dba_indexes i
WHERE
  i.owner = '<SCHEMA>'
  AND NOT EXISTS (
    SELECT
      1
    FROM
      dba_hist_sql_plan p
      INNER JOIN dba_hist_sqlstat s ON (p.sql_id = s.sql_id)
    WHERE
      p.object_owner = i.owner
      AND p.object_name = i.index_name
      AND p.operation LIKE '%INDEX%'
  )
ORDER BY 1,2
;
```

C.7 Monitoring the Reporting Server

C.7.1 CPU Utilization Trend

The CPU utilization for the reporting servers should be measured over time. The utilization is the load on the CPU. The common measurements are either the CPU load or the % utilization.

Running reports within Cognos seem to be CPU intensive on the reporting server as well as having an impact on the database through the underlying queries. By reviewing the reporting server CPU load, we can see if the reports need to be tuned or the reporting server configuration may need to be modified (such as the maximum number of concurrently running reports).

C.7.1.1 Tools

Most monitoring tools provide some CPU monitoring capability. At Model N, we use an open source tool called internally to monitor our servers. For more information, see http://www.zabbix.com.

C.7.2 Top Execution Times by Report

Cognos reports may be intensive for both the Cognos Report Server as well as the database. We have seen the Cognos Report Server take significant CPU cycles to build a report. We have also seen some report queries be complex and database intensive, potentially adding to the contention in the database. The report execution times should provide an indicator of potential problems from reports.

D Revisions

This appendix covers the revisions made to this document based on changes in the Model N application suite, or expansion or inclusion of the content. It does not address editorial changes.

D.1 Changes in 5.5

No updates were made in this release.

Index