

# **PAM** Pack 7 Tutorial: Managing user behavior through fear and guilt (I know what you did and when you did it)

**PIPER-Rx Application Monitor - **PAM****

*"Blurring the line between software product and training"*

**PAM** Version 1.0

April 2010

## Table of Contents

1	Before you start anything.....	4
2	What you'll get out of this <i>PAM</i> tutorial / software pack .....	5
3	Pending Requests (CM-002) .....	5
3.1	Why request wait time is not a good measure of concurrent manager health 6	
3.2	Concurrent Manager health – a better measure .....	7
3.3	What is a pending request?.....	8
3.4	<i>PAM</i> Pending requests e-mail alert .....	8
3.5	What to do with this information.....	10
3.5.1	Don't just add managers.....	11
3.5.2	Programs that spawn child concurrent programs .....	11
3.6	Terminating requests .....	12
3.7	How the pending requests alert threshold is calculated.....	13
3.8	Changing the alert threshold multiplier value .....	13
3.9	Turning the Pending requests alert off and on.....	14
3.10	Preventing multiple alerts .....	14
3.11	Changing the alert delay time .....	14
3.12	Excluding Managers from the pending requests check .....	15
3.13	Excluding additional managers from the pending requests alert .....	15
4	Long running requests (CM-003).....	16
4.1	When is a long running request a real long running request?.....	16
4.2	<i>PAM</i> Long running requests e-mail alert.....	18
4.3	What to do with this information.....	19
4.4	How are long running requests calculated?.....	20
4.4.1	Setting the minimum run parameters.....	20
4.5	How are the reference request runtime stats collected?.....	20
4.5.1	Standard deviation.....	21
4.6	Setting the standard deviations value .....	21
4.7	Listing the current <i>PAM</i> long running request run times .....	22
4.8	Resetting <i>PAM</i> request runtime statistics .....	22
4.9	Excluding a concurrent program from the long running requests check 23	
4.9.1	Viewing the <i>PAM</i> program monitored programs exceptions list	24
4.9.2	Adding a program to <i>PAM</i> monitored programs exceptions .....	24
4.9.3	Deleting a <i>PAM</i> exceptions program .....	26
4.9.4	Changing a <i>PAM</i> monitored programs items .....	26
4.9.5	Referential Integrity .....	26
5	Duplicate Requests (CP-004) .....	26

5.1	PAM Duplicate Requests Alert .....	27
5.2	What to do with this information.....	28
5.2.1	Additional uses for this report .....	29
5.2.2	Calculate wasted processing time .....	29
5.3	Multiple alerts for the same duplicates.....	30
5.4	Preventing multiple alerts on the same duplicates.....	30
5.4.1	Changing the duplicate grace period hours .....	31
5.5	Excluding a concurrent program from the duplicate program check	31
5.5.1	Viewing the PAM program monitored programs exceptions list	32
5.5.2	Adding a program to PAM monitored programs exceptions .....	33
5.5.3	Deleting a PAM exceptions program .....	34
5.5.4	Changing a PAM monitored programs items .....	35
5.5.5	Referential Integrity .....	35
6	PAM auto purging feature.....	35
7	Disclaimer.....	38

## 1 Before you start anything

**Before you start anything**, you need to have read and agreed to our terms and conditions, downloaded all documents and downloaded and installed the **PAM** software from the **PIPER-Rx.com** website using the installation guide and release notes.

You should also have a copy of the FAQs handy as they should answer most of your questions. Any further questions can be emailed to us at [pam@piper-rx.com](mailto:pam@piper-rx.com) or you can send us a [tweet](#).

## 2 What you'll get out of this **PAM** tutorial / software pack

**PAM** pack 7 tutorial and software provides a look into inappropriate user behavior that may be responsible for overall performance issues. New **PAM** alerts provided in this release are:

- ❖ Alert when the number of pending requests exceeds a threshold value
- ❖ Alert when a long running request is detected
- ❖ Alert when duplicate requests detected

We will be providing a process of excluding specific managers from the pending request check and concurrent programs from both the long running and duplicate request checks so as you can tailor these alerts to your needs.

This tutorial will also show why the industry standard of determining concurrent manager health by measuring pending request wait time is not a good measure and we provide a better alternative.

## 3 Pending Requests (CM-002)

This **PAM** check is designed to alert when there is a backlog of pending requests in any of the concurrent manager queues.

I have stated many times in the past and I still firmly believe it holds true...

***“90+% of Oracle E-Business suite activity is completed via the concurrent managers”***

...so any impact on the managers has an overall application affect. There are any number of reasons that the number of pending requests can build up and it doesn't take much for that to happen. The following are a couple of the more frequent causes:

### Scenario 1:

A user request a job but puts in the wrong arguments.. *“Oops I meant to run a report for stationary for department one but I ran a one for the whole company”*. So a report that would have taken a couple of minutes to run could now take several hours and will tie up one of the manager process for that time.

**Note:** The early detection of long running requests will be covered later in this tutorial

### Scenario 2:

A user runs a request to find that it is not on the printer when they need it, and, assuming it did not run, they run it again and again and again. I have seen twelve (12) of the same request all pending and running at the same time.

**Note:** The detection duplicate requests will be covered in a later in this tutorial

In this tutorial **PAM** will send an alert when the number of requests in any concurrent manager exceeds a threshold value, thus alerting you to the fact there may be an issue to resolve.

## 3.1 Why request wait time is not a good measure of concurrent manager health

For many years now all the big players have followed the line that the length of time a request waits to be processed (pending time) is a measure of concurrent manager health. The inference is the longer requests wait the healthier the concurrent managers are.

That's not to say that a request that has been in the queue longer than normal is not having a problem, but as for manager health, well that's a stretch! Let's use some scenarios to explain:

### Scenario 1 – Overnight managers

What if you have an Overnight manager that only wakes up a 7:00pm. Placing a request in that queue at 3:00pm will cause it to remain in a status of Pending Error until the overnight manager starts and by 7:00pm it will have a wait time of 4 hours.

## Scenario 2 – Resubmit a prior request

It is common practice for users to resubmit a prior request, may be they want the same report but don't want to type the parameters again. When the program is run the new program picks up the requested start date of the prior program. Guess what, even if the new program run immediately, it will show a wait time of the time between when the original program was run and the time the latest started. This could be days....

## Scenario 3 – On-hold request

Try this one. Submit a concurrent program and place it on-hold before it runs. Now, in 5 days time take it off hold and let it run. No surprises for guessing the wait time will be - 5 days.

## Scenario 4 – Custom manager to run slow requests

Suppose you have created a concurrent manager "SLOW MANAGER" to run all the slow, resource intensive requests, any request sent to that queue should be long running. If you get a few of these requests in the queue the wait time should be quite large. This does not indicate the slow manager is unhealthy, it is behaving normally.

All these examples are normal processing within an Oracle E-Business Suite environment. None of these scenarios of long wait times indicates a problem with the concurrent manager configuration.

## 3.2 Concurrent Manager health – a better measure

I have been using this measure for most of my career:

***"The number of requests in the queues is the best indicator of concurrent manager health"***

This approach must be implemented on a manager by manager basis as each manager has a different processing profile; "STANDARD" is different to the "SLOW" Manager and in turn is different to the "FAST" Manager.

Your "Fast" manager should have much lower throughput and faster turnaround than the standard managers, where as the "Slow" manager should

have a few, long running requests. What they all have in common is that if there is something wrong the number of pending requests for that manager will increase to a high level for **that** manager.

I.e. If you know you have a normal queue load (pending requests) for your standard manager of 50 pending requests at any time, then, if it the number of pending requests reaches 75 or 100 you know there must be something wrong. In most cases there are a few slow processing jobs holding up the managers.

If the slow queue has a normal peak of 5 requests, when it gets to 20 then there is most likely something wrong. None of this is rocket science, its just about understanding basic queue behavior.

### 3.3 What is a pending request?

**PAM** defines a pending request as any request that will run when there is a manager process available to run it. Thus the following pending statuses are not considered pending for the **PAM** pending request check:

- ❖ On-hold
- ❖ Scheduled
- ❖ Pending error

A status of Pending error can be because there are no managers to run that request. This usually occurs when the managers are down or the manager processes have not been started yet (which is normally the case for “overnight” manager configurations). In an overnight manager configuration any requests submitted for that manager will remain with a pending error status until the overnight manager is started.

### 3.4 PAM Pending requests e-mail alert

When the number of pending requests for any of your manager queues exceeds the **PAM** threshold a **PAM** alert e-mail is raised:

**Example: PAM CM-002 – PAM pending requests e-mail alert message**

**ALERT MESSAGE FROM PAM - PIPER-Rx Application Monitor - DO NOT REPLY**

Customer = DEMO  
Site = 12i  
Alert Level = **Warning**  
Detected = 07-Jan-10 (Sun) 07:46:01  
Alert Frequency = 5 Minutes

---

**The STANDARD manager has 321 pending requests which has exceeded the threshold value of 200**

---

**Alert Information:**

**CM-002 Pending Requests**

**THE NUMBER OF PENDING REQUESTS FOR THE MANAGER HAS EXCEEDED THE ALERT THRESHOLD VALUE,**

The threshold is set to **X** times the number of manager processes. (Default threshold = 20 times)

**Example:** If there are 5 standard manager processes and the alert level is set to 20, then the alert level will be 20 times 5 = 100. Thus an alert will be raised for the standard managers when the number of pending requests exceeds 100

If you want to obtain a list of pending requests for the selected manager you can use the **PAM** action report [PAMACP002-10 Pending Requests \(manager\)](#)

**Note 1:** This alert will continue to alert until the number of pending requests for the manager drops below the alert threshold

**Note 2:** The **PAM** action report will report on the pending requests at the time the report is run, not the time of the alert

**Note 3:** If you want to change the alert threshold value refer to the FAQs for more information

### 3.5 What to do with this information

The first step is to identify what is running. You can use the **PAM** Action report **PAMACP003-10 Running Requests** to list all running requests **at the time the report was run.**

#### Example - **PAMACP003-10 Running Requests** report

PAMACP003-10		PAM - PIPER-RX - APPLICATION MONITOR			PIPER - Rx	
RUNNING REQUESTS						
As at 19-Mar-10 10:32:52						
For PROD 121						
Request ID	Actual Start	Requestor	Program Name	Argument	Runtime (min)	
<b>STANDARD - Standard queue for handling requests</b>						
304635	19-Mar-10 10:29	GPIPER	PIPER-RX Pam Long running request	20	4	
304634	19-Mar-10 10:29	GPIPER	PIPER-RX Pam Long running request	20	4	

The runtime (minutes) shows the amount of time the report has been running at the time the report is generated.

The next step is to list what is in the queue that is currently backlogged. This report can also be used to show who is in the queue waiting for requests to complete (i.e. those affected by the backlog).

You can use the **PAM** Action report **PAMACP002-10 Pending Requests (manager)** to list all pending requests for a selected manager, using the manager name from the **PAM** alert.

#### Example - **PAMACP002-10 Pending Requests (manager)** report

PAMACP002-10		PAM - PIPER-RX - APPLICATION MONITOR			PIPER - Rx	
Pending Requests for Manager - STANDARD						
(Standard queue for handling requests)						
As at 19-Mar-10 10:31:40						
For PROD 121						
Request ID	Requested Start	Requestor	Program Name	Argument		
304638	19-Mar-10 10:28	GPIPER	PIPER-RX Pam Long running request	5		
304637	19-Mar-10 10:28	GPIPER	PIPER-RX Pam Long running request	5		
304636	19-Mar-10 10:28	GPIPER	PIPER-RX Pam Long running request	20		
<b>This report does not include on hold or scheduled requests</b>						

From both these reports you should be able to identify what is holding up the managers and the extent of requests waiting to be processed.

You now should have sufficient information to identify the user causing the problem and explain to them the affect of their inappropriate use of the concurrent managers... 😊

### 3.5.1 Don't just add managers

Don't think just adding more managers will solve your problems. There is a general rule of thumb that , if you have more managers that perform the bulk of the work (standard and custom managers) than two (2) times the number of CPUs then you start running the risk of overloading your CPUs.

I have seen a site that had configured 100 standard manager processes on an 8 CPU box. It was not unusual to see all CPU's at 100% during most of the processing day and no surprises to note the load was caused by concurrent manager processes.

### 3.5.2 Programs that spawn child concurrent programs

I have found at a number of sites where a customisation requires a concurrent program to spawn a child concurrent program and the parent goes into a wait state waiting for the child to complete, once the child completes its processing the parent continues running. The issue with this is that the parent holds a concurrent process for the duration of the run of all its children, thus, for a period of time limiting the number of concurrent manager processes that can process the normal load.

#### Example:

- ❖ You have three ( 3 ) manager processes
- ❖ You have a concurrent program that spawns a child program and that child spawns a child – so at any given time you can have three (3) concurrent programs running related to the one program.

#### The Issue:

The primary parent will hold one (1) concurrent manager process for the duration of its life and the first child will hold one (1) concurrent manager process for the duration of its life. Now, the first child needs to spawn its child to complete the overall process, however there are 20 pending requests in the queue when the second child is spawned, thus the 20 requests need to complete before the second child can run.

So what we have is a backlog created by two (2) of the three (3) concurrent manager processes being tied up with waiting processes leaving only 1 manager to process the normal queue load. Only when the second child is

processed will the two parents be release and free up the concurrent manager processes.

### One possible solution:

Firstly – **DON'T** just add more manager processes, this just causes more issues than its worth.

A simple solution to this issue could be to assign each child concurrent program a slightly higher request priority than it's parent:

```
Parent Program - Priority 50
----- Child Program – Priority 48
-----Child Program – Priority 46
```

In this way the children will be processed prior to all other requests in the queue, freeing up the managers sooner.

**Note:** I don't normally recommend the use of priorities as once users understand that you can change a request's priority, they all believe their requests are more important than everyone else's.

## 3.6 Terminating requests

Never kill the request unless you really have to.

When you terminate a user's request they will blame you, when in fact they caused the problem in the first place. The best action is to explain there is a major backlog caused by their concurrent request/s and get them to terminate the offending request/s; that way they know they have caused the problem. Also, for any comments about performance from other users, you can politely let them know who caused the problem...

If you find you have something very important in the queue you really need to get through but it is way down the list, you could place all the requests that will run prior on-hold and then release them after the "important" request has completed. Care should be taken with this approach as you may cause issues with request sets and custom programs that submit sub processes as concurrent requests. Another alternative is you could assign the program to your special request type and ask the user to terminate the original and resubmit the request again which will run the program in your special queue.

All you need to know about request types is in a paper I wrote back in 2002 - **Concurrent Managers 101 and 202** – which paper can be found on the **PIPER-Rx** website at: [www.piper-rx.com/pages/papers/cm101.html](http://www.piper-rx.com/pages/papers/cm101.html)

### 3.7 How the pending requests alert threshold is calculated

The **PAM** alert threshold is calculated as the number of running processes for a manager times a multiplier value (Default value = 20).

Example: If you set the **PAM** multiplier value to 10 then:

- ❖ If the standard manager has 8 processes then **PAM** will alert when the number of pending requests for the standard manager exceeds 80 requests (  $8 * 10$  )
- ❖ If the custom slow queue manager has 2 processes then **PAM** will alert when the number of pending requests for the standard manager exceeds 20 requests (  $2 * 10$  )

**Note:** An alert will not be generated when the managers are shut down, however if requests are submitted whilst the managers are down, and the number of requests in the queues exceeds the **PAM** alert value, when the managers are restarted **PAM** will generate an alert.

The **PAM** pending request check multiplier is applied to all managers

### 3.8 Changing the alert threshold multiplier value

The **PAM** multiplier value can be set using the following **PAM** API:

```
exec PIPER_RX_PAM_API_2.PAM_SET_DELAY_PENDING_CHECK (20);
```

Parameter: The pending requests multiplier value

### 3.9 Turning the Pending requests alert off and on

The **PAM** pending requests alert can be turned off using the following **PAM** API:

```
exec PIPER_RX_PAM_API.PAM_ALERT_ENABLE ( 'CM-002', 'N' );
```

The alert can be re-enabled using the following **PAM** API:

```
exec PIPER_RX_PAM_API.PAM_ALERT_ENABLE ( 'CM-002', 'Y' );
```

### 3.10 Preventing multiple alerts

The default **PAM** pending request check (**PAM** Alert ID CM-002) frequency is every 5 minutes. In order to prevent an annoying e-mail alert every 5 minutes that the managers backlogged with requests whilst the backlog is being cleared the **PAM** alerts delay feature has been added to this check. After the first alert is raised **PAM** adds a delay time to the next pending request check.

The current delay time (minutes) can be found using the following SQL:

```
SELECT working_value_description,  
       working_numeric_value  
FROM   piper_rx_pam_config  
WHERE  alert_id = 'CM-002';
```

Thus after the first alert is raised, **PAM** will continue to send an e-mail alerts but on a less frequent basis. The alert delay is removed when the backlog is cleared.

### 3.11 Changing the alert delay time

The delay time can be changed using the following **PAM** API:

```
exec PIPER_RX_PAM_API_2.PAM_SET_DELAY_PENDING_CHECK_CHECK ( 10 );
```

Parameter: The number of minutes to delay before the next check and alert.

**Note:** The minimum delay is 10 minutes; setting the parameter to anything smaller than 10 minutes will default to 10 minutes

### 3.12 Excluding Managers from the pending requests check

One or more managers can be excluded from the **PAM** pending requests check by adding the manager to be excluded to the **PAM** Manager Exceptions table [piper\\_rx\\_pam\\_conc\\_mgr\\_ex](#)

A list of excluded managers can be found using the **PAM** config report **PAMC021-10 PAM Concurrent Manager Exceptions**.

Example - **PAMC021-10 PAM Concurrent Manager Exceptions** report

PAMC021-10 PIPER-RX - APPLICATION MONITOR Concurrent Manager Exceptions List As at 19-Mar-10 10:16:08 For PROD 12i	
Manager Name	Pending Request Check
FNDCRM	Excluded

**Note:** By default the conflict resolution manager FNDCRM is excluded from the pending requests check.

### 3.13 Excluding additional managers from the pending requests alert

Additional managers can be excluded from the pending request check by adding the manager you wish to exclude to the **PAM** manager exceptions table [piper\\_rx\\_pam\\_conc\\_mgr\\_ex](#) using the following **PAM** API:

```
exec PIPER_RX_PAM_API_2.PAM_PENDING_CHECK_EX_ADD ( 'MRPMGR' );
```

Parameter: The Concurrent manager name you wish to exclude from the pending request alert

A manager can be removed from the **PAM** manager exceptions list using the following **PAM** API:

```
exec PIPER_RX_PAM_API_2.PAM_PENDING_CHECK_EX_DEL ( 'MRPMGR' );
```

Parameter: The Concurrent manager name you wish to remove from the **PAM** pending request alert exceptions list.

After a period of time you may have a number of managers in the **PAM** manager exceptions list that no longer exist. The following **PAM** API performs a **PAM** referential cleanup which will remove any managers from the **PAM** exceptions list that no longer exist in the OEBS application.

```
exec PIPER_RX_PAM_API.PAM_REFERENTIAL_CLEANUP;
```

## 4 Long running requests (CM-003)

What is a long running request?

I once heard a DBA define that a long running request as any request that takes longer than 3 hours. The DBA went on to state that they kill any request that takes longer than 3 hours.

How did this definition come about? From a technical perspective it is easy to find any request that has been running for longer than 3 hours:

```
SELECT request_id
FROM applsys.fnd_concurrent_requests fcr
WHERE (sysdate - fcr.actual_start_date ) > 3/24
and phase_code = 'R';
```

However there are in many cases requests that will take longer the 3 hours to complete and killing one of these could be a career limiting move.

### 4.1 When is a long running request a real long running request?

*“When it has been running longer than it normally does...”*

One of the great things about Oracle E-Business suite that is so often misunderstood by the technical people is its predictability. The same business processes are run each business cycle by usually the same people.

More often than not this means that the same reports with the same parameters are run on or about the same time each business cycle and they generally take about the same time to run.

Example: For your site it may be normal for a trial balance to have an average run time of 15 minutes so if a user enters the wrong account range then the report could take several hours to run. For this example lets say it would take 5 hours. If we could be notified that after 1 hour of running this program was a long running request and then terminate that program we would have saves 4 hours of unnecessary processing and free up that manager process for other processing

Detecting long running requests is not about hard and fast rules, its about identifying those requests that are running longer than normal that should be investigated.

## 4.2 PAM Long running requests e-mail alert

When PAM detects one or more long running requests a PAM alert e-mail is raised:

**Example: PAM CM-003 – PAM long running requests e-mail alert message**

**ALERT MESSAGE FROM PAM - PIPER-Rx Application Monitor - DO NOT REPLY**

Customer = DEMO  
Site = 12i  
Alert Level = **Warning**  
Detected = 21-Jan-10 (Sun) 07:05:01  
Alert Frequency = 10 Minutes

---

**Program PIPER-RX Pam Long running (Req id = 304655) has been running for 22 Min - (Avg = 10) and could be long running**

---

**Alert Information:**

**CM-003 Long Running Requests**

**PAM HAS DETECTED A REQUEST THAT HAS BEEN RUNNING SIGNIFICANTLY LONGER THAN THE NORMAL (AVERAGE) RUNTIME FOR THAT PROGRAM**

If you want to obtain a list of all running requests you can use the PAM action report [PAMACP003-10 Running Requests](#)

**Note 1:** The alert action report will report on the running requests at the time the report is run, not the time of the alert

**Note 2:** Report set and report set stages have been excluded from this alert

**Note 3:** It is good practice to not kill a long running program before getting the users permission to terminate it. This way, the user is aware of the issue they caused. If you kill their program you will instead be the villain.

### 4.3 What to do with this information

The issue is that a long running request will tie up a manager process for the duration of its life, reducing the number of manager processes available to processes other requests by one (1).

The first step is to run the **PAM** action report **PAMACP003-10 Running Requests** to list all running requests.

Example - **PAMACP003-10 Running Requests** report

PAMACP003-10		PAM - PIPER-RX - APPLICATION MONITOR				PIPER - Rx
RUNNING REQUESTS						
As at 21-Mar-10 07:20:21						
For PROD 121						
Request ID	Actual Start	Requestor	Program Name	Argument	Runtime (min)	
<b>STANDARD - Standard queue for handling requests</b>						
304655	21-Mar-10 06:51	GPIPER	PIPER-RX Pam Long running request	30	29	
304654	21-Mar-10 06:51	GPIPER	PIPER-RX Pam Long running request	30	29	

If the request has completed by the time you run this report the potential long running request will not appear. However all is not lost...

You can use the **PAM** report **PAMRGA005-10 Application Activity (period)** entering a to and from date range that covers the long running period and view all activity that was occurring during that time.

Example - **PAMRGA005-10 Application Activity (period)** report

PAMRGA005-10		PAM - PIPER-RX - APPLICATION MONITOR				PIPER - Rx
ISSUE AFFECT						
Issue Period - From 21-Mar-10 06:50 To 22-Mar-10 07:30						
As at 21-Mar-10 07:26						
For PROD 121						
FS - Login ID	SS - Session ID	CR - Request ID	User Name	FS - User Description	SS - User Description	CR - Program Name
				Session Start	Session End	Account Type
<b>Full Service Users:</b>						
314050	GPIPER		Gary Piper	21-Mar-10 06:50	21-Mar-10 06:51	(--)
314057	GPIPER		Gary Piper	21-Mar-10 07:15	21-Mar-10 07:16	(--)
<b>Total Affected: 2</b>						
<b>Concuurent Requests:</b>						
304654	GPIPER		PIPER-RX Pam Long running request	21-Mar-10 06:51	21-Mar-10 07:21	(--)
			<i>Argument Text: 30</i>			
304655	GPIPER		PIPER-RX Pam Long running request	21-Mar-10 06:51	21-Mar-10 07:21	(--)
			<i>Argument Text: 30</i>			
<b>Total Affected: 2</b>						

You can now seek out the guilty party.. 😊

## 4.4 How are long running requests calculated?

The first part of the calculation is to exclude any request that runs in less than **X** minutes, as we don't want to be notified of every little request. The aim is to catch the real long runners. Next, exclude any programs that do not have sufficient historical runs to provide useful runtime data.

The **PAM** long running request check periodically (every 10 minutes) checks each running request against **PAM** collected runtime statistics which are collected / updated on a weekly basis.

### 4.4.1 Setting the minimum run parameters

In order for a program's runtime to be included in the **PAM** runtime history a concurrent program must have been run more than **X** times in the available history and must have a minimum run time of **X** minutes.

The minimum runs and runtime can be set using the following **PAM** API:

```
exec PIPER_RX_PAM_API.PAM_LR_REQUEST_SETTINGS_SET ( 5, 10 );
```

Parameter 1:

The minimum number of historical runs before a program is included in the long running check

Parameter 2:

The minimum runtime (minutes) before a program is included in the long running check

## 4.5 How are the reference request runtime stats collected?

Once per week (default) **PAM** evaluates via the **PAM** IN-007 collector the request runtimes in the **fnd\_concurrent\_request** table and updates the runtime values in the **PAM piper\_rx\_pam\_cr\_runtimes** table. That way if a program is "fixed", or new hardware is added the change in **PAM** runtimes will be reflected in the **PAM** runtime information.

**Note:** If your site selectively purges programs from the **fnf\_concurrent\_requests** table or you have a short purge cycle i.e. less than 32 days, the runtime information recorded by **PAM** may be unsuitable.

Concurrent requests that complete with any other status than normal i.e. terminated, error, etc... are not included in the **PAM** statistics.

### 4.5.1 Standard deviation

When **PAM** collects its request runtime statistics it collects the average run time plus the standard deviation for each program. If we were to assume a normal distribution then 95% of all results will fall within 2 standard deviations of the mean and 99.7% will fall between 3 standard deviations of the mean.

Thus if a program's runtime is recorded as 20 minutes with a standard deviation of 4 minutes then:

95% of runs would fall within the average + 2 standard deviations

$$20 + (2 * 4) = 28 \text{ minutes}$$

95% of runs would fall within the average + 2 standard deviations

$$20 + (3 * 4) = 32 \text{ minutes}$$

**PAM** sets the default standard deviation to 2, thus in this example if a program has been running longer than 28 minutes it will be alerted as a candidate for review.

**Note:** To the purest statistician; yes, I am misusing the standard deviation, and I do understand that runtimes are not the best example of a normal distribution and that the sample sizes are too small. That being said, what we need is an indicator to let us know a request is **probably** running longer than it should and is a candidate for investigation. Over the past 20 years I have found this to be one of the best indicators for our purpose.

## 4.6 Setting the standard deviations value

The **PAM** long running request standard deviation value can be set using the following **PAM** API:

```
exec PIPER_RX_PAM_API.PAM_LR_REQUEST_ALERT_STDDEV ( 3 );
```

Parameter: The number of standard deviations to use in the calculation of a long running request

## 4.7 Listing the current **PAM** long running request run times

You can use the **PAM** report **PAMRCP003-11 Request Runtime Stats** to show the current **PAM** collected request runtime statistics:

### Example - **PAMRCP003-11 Request Runtime Stats** report

PAMRCP003-11									
PAM - PIPER-RX - APPLICATION MONITOR									
REQUEST RUNTIME STATS									
As at 22-Mar-10 08:13									
For PROD 121									
Long Running Request Limits - Minimum Runs 5 : Minimum Minutes 10 : STDDEVS 3									
Appn Id	Application	Program Id	Program	Last Stats Update	Run Count	Minimum	Average	Maximum	Runtime Minutes
0	FND	37815	PIPER-RX Pam Long running request	19/03/2010 11:47:27 a.m.	19	0	10	10	

The title bar lists the current minimum runs, minimum runtime and standard deviation currently being used by the **PAM** long running request check.

**Note:** This report does not show all concurrent request runtime information, it shows only those programs that are within the **PAM** thresholds of minimum runs and minimum run time.

## 4.8 Resetting **PAM** request runtime statistics

The **PAM** long running request reference statistics can be reset using the following **PAM** API:

```
exec PIPER_RX_PAM_API_2.PAM_RUNTIME_STATS_RESET;
```

This API will **delete** all existing **PAM** collected statistics from the **PAM** **piper\_rx\_pam\_cr\_runtimes** table and repopulate with current information from the application **fnd\_concurrent\_requests** table.

## 4.9 Excluding a concurrent program from the long running requests check

Concurrent programs can be excluded from the long running program check. The most obvious programs to be excluded are report sets and report set stages; both these are excluded by default.

**PAM** uses a central concurrent programs exceptions table `piper_rx_pam_cp_monitor_tl` to hold all **PAM** concurrent program exceptions. The exceptions maintained in this table are:

### Exclude from duplicates

Program to be excluded from the **PAM** duplicate requests check

### Exclude from long running

Program to be excluded from the **PAM** long running requests check

### Check program exists \*

The program must exist in the `fnl_concurrent_requests` table on a pending or running state – designed to alert if a normal maintenance program does not exist, has been deleted or placed on-hold

### Check completed error\*

Alert if the concurrent program completes with a status of error

### Check completed warning\*

Alert if the concurrent program completes with a status of warning

### Check submitted\*

Alert when this program has been submitted. This could be used to alert you to the existence of a program with a known performance issue

\* These items will be covered in future tutorials

### 4.9.1 Viewing the PAM program monitored programs exceptions list

You can use the PAM config report **PAMC009-10 PAM Program Monitor List** to list all PAM concurrent program exceptions:

Example - **PAMC009-10 PAM Program Monitor List** report

PAMC009-10		PIPER-RX - APPLICATION MONITOR PROGRAM MONITOR LIST As at 23-Mar-10 09:01:44 For PROD 121					PIPER - Rx			
Application	Prog Id	Program Name	Check Status	Exists	Completed		Submitted	Exclude Duplicates	Exclude Long Running	
					Error	Warning				
0 (FND)	0	Activate Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes	
0 (FND)	1	Deactivate Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes	
0 (FND)	3	Restart Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes	
0 (FND)	4	Abort Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes	
0 (FND)	5	Shutdown Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes	
0 (FND)	6	Startup Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes	
0 (FND)	31659	Report Set	Enabled	No	No	No	No	No	Yes	
0 (FND)	32263	Purge Concurrent Request and/or Manager Data	Enabled	Yes	No	No	No	No	Yes	
0 (FND)	32592	Purge Signon Audit data	Enabled	Yes	No	No	No	No	Yes	
0 (FND)	36034	Request Set Stage	Enabled	No	No	No	No	Yes	Yes	
0 (FND)	36888	Workflow Background Process	Enabled	No	No	No	No	Yes	Yes	
101 (GL)	32045	Program - Optimizer	Enabled	Yes	No	No	No	No	Yes	
178 (ICX)	36662	Delete data from temporary table	Enabled	Yes	No	No	No	No	Yes	

### 4.9.2 Adding a program to PAM monitored programs exceptions

Concurrent programs can be added to the PAM monitored programs exception list using the following PAM API:

```

BEGIN

PIPER_RX_PAM_API.PAM_MONITORED_PROGRAM_ADD
  ( 101,      -- Program Application ID
    101,      -- Program ID
    'Y',      -- Alert status - alert on or off
    'N',      -- Alert if program does not exist
    'Y',      -- Alert if completed error
    'Y',      -- Alert if completed earning
    'N',      -- Alert if submitted
    'N',      -- Exclude from duplicates check
    'Y' );   -- Exclude from long running check

END;
```

In this example we have added the GL posting program alerting if the program completes with a status of either error or warning (this actual alert will be implemented in a later tutorial) and to exclude the program from the PAM long running requests check.

When a program is first added the program name will be displayed as “unknown”. This value will be updated with the program name when the PAM check program exists check (CP-001) is run.

PAMC009-10		PIPER-RX - APPLICATION MONITOR PROGRAM MONITOR LIST As at 23-Mar-10 09:16:19 For PROD 121					PIPER - Rx		
Application	Prog Id	Program Name	Check Status	Exists	Completed			Exclude Duplicates	Exclude Long Running
					Error	Warning	Submitted		
0 (FND)	0	Activate Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes
0 (FND)	1	Deactivate Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes
0 (FND)	3	Restart Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes
0 (FND)	4	Abort Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes
0 (FND)	5	Shutdown Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes
0 (FND)	6	Startup Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes
0 (FND)	31659	Report Set	Enabled	No	No	No	No	No	Yes
0 (FND)	32263	Purge Concurrent Request and/or Manager Data	Enabled	Yes	No	No	No	No	Yes
0 (FND)	32592	Purge Signon Audit data	Enabled	Yes	No	No	No	No	Yes
0 (FND)	36034	Request Set Stage	Enabled	No	No	No	No	Yes	Yes
0 (FND)	36888	Workflow Background Process	Enabled	No	No	No	No	Yes	Yes
101 (GL)	101	Unknown	Enabled	No	Yes	Yes	No	No	Yes
101 (GL)	32045	Program - Optimizer	Enabled	Yes	No	No	No	No	Yes
178 (CX)	36662	Delete data from temporary table	Enabled	Yes	No	No	No	No	Yes

**Note:** The PAM check program exists alert will be implemented in a later tutorial

To validate your entry prior to the PAM check program exists check (CP-001) being run, you can use the PAM config report PAMC010-10 PAM Program Monitor Validate:

Example - PAMC010-10 PAM Program Monitor Validate report

PAMC010-10		PIPER-RX - APPLICATION MONITOR PROGRAM MONITOR LIST - VALIDATION As at 23-Mar-10 09:20:24 For PROD 121			PIPER - Rx	
Prog Appn	Prog Id	Program Name	Validation			
0	0	Activate Concurrent Manager	Program Exists			
0	1	Deactivate Concurrent Manager	Program Exists			
0	3	Restart Concurrent Manager	Program Exists			
0	4	Abort Concurrent Manager	Program Exists			
0	5	Shutdown Concurrent Manager	Program Exists			
0	6	Startup Concurrent Manager	Program Exists			
0	31659	Report Set	Program Exists			
0	32263	Purge Concurrent Request and/or Manager	Program Exists			
0	32592	Purge Signon Audit data	Program Exists			
0	36034	Request Set Stage	Program Exists			
0	36888	Workflow Background Process	Program Exists			
101	101	* Posting	Program Exists			
101	32045	Program - Optimizer	Program Exists			
178	36662	Delete data from temporary table	Program Exists			

\* Indicates the program name has not been updated by the program monitor  
This will occur the next time the program monitor runs

### 4.9.3 Deleting a PAM exceptions program

A concurrent program can be removed from the PAM program exception list by using the following PAM API:

```
exec PIPER_RX_PAM_API.PAM_MONITORED_PROGRAM_DEL ( 101, 101);
```

Parameter 1: The program application\_id

Parameter 2: The concurrent program id

### 4.9.4 Changing a PAM monitored programs items

The PAM monitored programs settings can be changed by first deleting the existing entry and then adding the program with the revised parameters.

### 4.9.5 Referential Integrity

After a period of time you may have a number of programs in the PAM monitored program list that no longer exist. The following PAM API performs a PAM referential cleanup which will remove any program from the PAM exceptions list that no longer exists in the OEBS application.

```
exec PIPER_RX_PAM_API.PAM_REFERENTIAL_CLEANUP;
```

## 5 Duplicate Requests (CP-004)

PAM defines a duplicate request as the same program submitted by the same user with the same arguments with either a status of running or pending.

**Note:** Both on-hold and scheduled requests are not included in the duplicate request check.

Ever had the situation when the users are under a heavy work load, usually at peak processing times like month end and they submit a request and then run to the printer only to find the report has not printed. Then they go back and resubmit the same request.

The first report is still in the queue, by resubmitting again and again the reports are all in the queue and when they eventually get run they will

compete with each other as they are accessing the same information... I have personally seen twelve (12) of the same report by the same user.

In the main duplicate requests are a waste of processing resources.

Normally you would be alerted to the existence of duplicate requests when users complain about performance, or the number of pending requests increases to an unacceptable level and users complain their requests are running slow and you look at the running and pending requests list.

## 5.1 PAM Duplicate Requests Alert

When PAM detects a duplicate request PAM will generate the following e-mail alert:

**Example: PAM CP-004 – PAM Duplicate request alert e-mail alert message**

**ALERT MESSAGE FROM PAM - PIPER-Rx Application Monitor - DO NOT REPLY**

Customer = DEMO  
Site = 12i  
Alert Level = **Warning**  
Detected = 21-Mar-10 (Sun) 08:50:01  
Alert Frequency = 15 Minutes

---

**User GPIPER has submitted 3 PIPER-RX Pam Long running requests with arguments – 30**

---

**Alert Information:**

**CP-004 - Duplicate Requests**

**ONE OR MORE DUPLICATE REQUESTS HAVE BEEN DETECTED.**

**Note 1:** Specific programs can be excluded from the duplicate requests check by adding an entry in the **PAM piper\_rx\_pam\_cp\_monitor\_tl** table setting the exclude duplicates check to [Y]. Refer to the FAQs for more information

**Note 2:** If you want to obtain a list of programs excluded from the duplicate requests check you can use the **PAM** config report [PAMC009-10 PAM Program Monitor List](#)

## 5.2 What to do with this information

When alerted to the existence you need to identify the duplicates, and assess the extent of the issue. Normally with duplicates timing is off the essence. If you don't investigate the duplicates when they occur you have missed the opportunity. Trying to identify them after the fact is a very complex task and rarely done. All is not lost however; with **PAM** we have introduced two reports:

The **PAM** report [PAMRGA006-10 Application Activity \(current\)](#) lists the current activity within the application:

### Example - [PAMRGA006-10 Application Activity \(current\)](#) report

PAMRGA006-10		PAM - PIPER-RX - APPLICATION MONITOR		PIPER - Rx	
As at 21-Mar-10 06:54					
For PROD 121					
FS - Login ID	FS - User Description		Session Start	Account Type	
SS - Session ID	SS - User Description				
CR - Request ID User Name	CR - Program Name				
<b>Concuurent Requests:</b>					
303929	PRJUMFG	Pending - Process transaction interface	05-Jan-06 09:29	(E-)	
304656	<i>Argument Text (None)</i>	Pending - PIPER-RX Pam Long running request	21-Mar-10 06:51	(-)	
304654	<i>Argument Text 30</i> ←	Running - PIPER-RX Pam Long running request	21-Mar-10 06:51	(-)	
304657	<i>Argument Text 30</i> ←	Pending - PIPER-RX Pam Long running request	21-Mar-10 06:51	(-)	
304658	<i>Argument Text 5</i>	Pending - PIPER-RX Pam Long running request	21-Mar-10 06:51	(-)	
304659	<i>Argument Text 5</i>	Pending - PIPER-RX Pam Long running request	21-Mar-10 06:51	(-)	
304655	<i>Argument Text 30</i> ←	Running - PIPER-RX Pam Long running request	21-Mar-10 06:51	(-)	

In this report you can clearly see the three (3) Piper-Rx long running requests mentioned in the example alert above with an argument of 30.

However in the real world, you are not always around to run this report when you receive the alert message. So **PAM** has provided a second report [PAMRGA005-10 Application Activity \(period\)](#) that is designed to show the activity that was occurring in the past. With this report you enter a start time and an end time and the report will list the activity that was occurring during that time:

Example - **PAMRGA005-10 Application Activity (period)** report

PAMRGA005-10		PAM - PIPER-RX - APPLICATION MONITOR			PIPER - Rx
ISSUE AFFECT					
Issue Period - From 21-Mar-10 06:50 To 21-Mar-10 07:30					
As at 23-Mar-10 10:04					
For PROD 121					
FS - Login ID	FS - User Description				
SS - Session ID	SS - User Description		Session Start	Session End	Account Type
CR - Request ID	CR - Program Name				
<b>Full Service Users:</b>					
314050	GPIPER	Gary Piper	21-Mar-10 06:50	21-Mar-10 06:51	(--)
314057	GPIPER	Gary Piper	21-Mar-10 07:15	21-Mar-10 07:16	(--)
<b>Total Affected:</b>		2			
<b>Concurrent Requests:</b>					
304654	GPIPER	PIPER-RX Pam Long running request	21-Mar-10 06:51	21-Mar-10 07:21	(--)
	<i>Argument Text 30</i>				
304655	GPIPER	PIPER-RX Pam Long running request	21-Mar-10 06:51	21-Mar-10 07:21	(--)
	<i>Argument Text 30</i>				
304656	GPIPER	PIPER-RX Pam Long running request	21-Mar-10 07:21	21-Mar-10 07:51	(--)
	<i>Argument Text 30</i>				
304657	GPIPER	PIPER-RX Pam Long running request	21-Mar-10 07:21	21-Mar-10 07:26	(--)
	<i>Argument Text 5</i>				
304658	GPIPER	PIPER-RX Pam Long running request	21-Mar-10 07:26	21-Mar-10 07:31	(--)
	<i>Argument Text 5</i>				
<b>Total Affected:</b>		5			

Looking at the report header information you can see this report was run on the 23-Mar-10 showing activity that was occurring between 21-Mar-10 06:50 and 22-Mar-10 07:30 some two days prior.

The account type in this report (---) will detail if the account is an:

- Employee (E--)
- Customer (-C-)
- Supplier (--S)

In this way you can identify if any customers where affected by the performance hit related to duplicate requests.

### 5.2.1 Additional uses for this report

This report can be used to assess the impact of a performance issue. By setting the start and end times to a known performance issue time, you will be provided a list of all full and self service users connected at the time of the issue. You can then use the account type indicator in the report to identify if any customers were affected.

### 5.2.2 Calculate wasted processing time

One of the things I have done with duplicates in the past is to record the number of duplicates and sum up the total run time for the duplicates over a

period of a month. For this example, let's say I found a total of 10 hours of processing duplicates for the month.

The next step is to place an arbitrary value on the processing of these duplicates. For this example let's say \$1000 per hour. It does not have to be accurate as you are only proving a point.

Ten (10) hours at \$1,000 per hour is \$10,000 of unnecessary processing. Once you put this into a dollar value you have the attention of business management.

The next step is to gain approval to manage duplicates by terminating duplicates when they are found. Now in this situation, if you terminate a user's request you have done so with the sanction of business management.

### 5.3 Multiple alerts for the same duplicates

**Issue:** If there are five (5) duplicates, when one completes there will still be four (4) duplicates in the queues. What we don't want is **PAM** to alert every 15 minutes (default) that there are the same duplicates, but we do want to be notified if there are more duplicates submitted or duplicates submitted by other users.

**PAM** has introduced a duplicate grace period (Hours). That is, when a set of duplicates has been detected (user, program, arguments), **PAM** will not report on these duplicates again for **X** hours. This should provide enough time for the duplicates to clear or be manually cleared (cancelled etc...).

### 5.4 Preventing multiple alerts on the same duplicates

The nature of duplicates is that they may take some time to clear.

Example: User A submits 3 of the same program with the same arguments.

At time 1:

**PAM** detects three pending duplicate requests for user A and sends an alert

At time 2:

One request has completed, one is running and the third is pending, on the next check cycle **PAM** detects the two remaining duplicates and would send an alert if not for the grace period feature.

In effect with this scenario **PAM** will have sent two alerts for the same duplicates which is what we wanted to prevent.

The **PAM** duplicate request check introduces the duplicate grace period default 1 hour. That is once an alert is sent for a set of duplicate requests (same user, same program, same augments) **PAM** will not alert on these duplicates again until the duplicate grace period has passed. In this way we prevent multiple alerting by allowing sufficient time to pass for the duplicates to clear or be cleared.

#### 5.4.1 Changing the duplicate grace period hours

The **PAM** duplicate request grace period can be changed using the following **PAM** API:

```
exec PIPER_RX_PAM_API_2.PAM_DUPLICATES_GRACE_SET ( 1 );
```

Parameter: The number of hours to suspend a duplicate check for detected duplicates.

#### 5.5 Excluding a concurrent program from the duplicate program check

Concurrent programs can be excluded from the duplicate program check.

**PAM** uses a central concurrent programs exceptions table **piper\_rx\_pam\_cp\_monitor\_tl** to hold all **PAM** concurrent program exceptions. The exceptions maintained in this table are:

### Exclude from duplicates

Program to be excluded from the **PAM** duplicate requests check

### Exclude from long running

Program to be excluded from the **PAM** long running requests check

### Check program exists \*

The program must exist in the **find\_concurrent\_requests** table on a pending or running state – designed to alert if a normal maintenance program does not exist, has been deleted or placed on-hold

### Check completed error\*

Alert if the concurrent program completes with a status of error

### Check completed warning\*

Alert if the concurrent program completes with a status of warning

### Check submitted\*

Alert when this program has been submitted. This could be used to alert you to the existence of a program with a known performance issue

\* These items will be covered in future tutorials

## 5.5.1 Viewing the **PAM** program monitored programs exceptions list

You can use the **PAM** config report **PAMC009-10 PAM Program Monitor List** to list all **PAM** concurrent program exceptions:

Example - PAMC009-10 PAM Program Monitor List report

PAMC009-10		PIPER-RX - APPLICATION MONITOR PROGRAM MONITOR LIST As at 23-Mar-10 09:01:44 For PROD 121					PIPER - Rx		
Application	Prog Id	Program Name	Check Status	Exists	Completed			Exclude Duplicates	Exclude Long Running
					Error	Warning	Submitted		
0 (FND)	0	Activate Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes
0 (FND)	1	Deactivate Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes
0 (FND)	3	Restart Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes
0 (FND)	4	Abort Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes
0 (FND)	5	Shutdown Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes
0 (FND)	6	Startup Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes
0 (FND)	31659	Report Set	Enabled	No	No	No	No	No	Yes
0 (FND)	32263	Purge Concurrent Request and/or Manager Data	Enabled	Yes	No	No	No	No	Yes
0 (FND)	32592	Purge Signon Audit data	Enabled	Yes	No	No	No	No	Yes
0 (FND)	36034	Request Set Stage	Enabled	No	No	No	No	Yes	Yes
0 (FND)	36898	Workflow Background Process	Enabled	No	No	No	No	Yes	Yes
101 (GL)	32045	Program - Optimizer	Enabled	Yes	No	No	No	No	Yes
178 (ICX)	36662	Delete data from temporary table	Enabled	Yes	No	No	No	No	Yes

### 5.5.2 Adding a program to PAM monitored programs exceptions

Concurrent programs can be added to the PAM monitored programs exception list using the following PAM API:

```

BEGIN

  PIPER_RX_PAM_API.PAM_MONITORED_PROGRAM_ADD
    ( 101,      -- Program Application ID
      101,      -- Program ID
      'Y',      -- Alert status - alert on or off
      'N',      -- Alert if program does not exist
      'Y',      -- Alert if completed error
      'Y',      -- Alert if completed earning
      'N',      -- Alert if submitted
      'Y',      -- Exclude from duplicates check
      'N' );    -- Exclude from long running check

END;
```

In this example we have added the GL posting program alerting if the program completes with a status of either error or warning (the actual alert will be implemented in a later tutorial) and to exclude the program from the PAM long running requests check.

When a program is first added the program name will be displayed as “unknown”. This value will be updated with the program name when the PAM check program exists check (CP-001) is run.

PAMC009-10		PIPER-RX - APPLICATION MONITOR PROGRAM MONITOR LIST As at 23-Mar-10 14:32:12 For PROD 12i					PIPER - Rx		
Application	Prog Id	Program Name	Check Status	Exists	Completed			Exclude Duplicates	Exclude Long Running
					Error	Warning	Submitted		
0 (FND)	0	Activate Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes
0 (FND)	1	Deactivate Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes
0 (FND)	3	Restart Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes
0 (FND)	4	Abort Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes
0 (FND)	5	Shutdown Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes
0 (FND)	6	Startup Concurrent Manager	Enabled	No	Yes	Yes	Yes	No	Yes
0 (FND)	31659	Report Set	Enabled	No	No	No	No	No	Yes
0 (FND)	32263	Purge Concurrent Request and/or Manager Data	Enabled	Yes	No	No	No	No	Yes
0 (FND)	32592	Purge Signon Audit data	Enabled	Yes	No	No	No	No	Yes
0 (FND)	36034	Request Set Stage	Enabled	No	No	No	No	Yes	Yes
0 (FND)	36888	Workflow Background Process	Enabled	No	No	No	No	Yes	Yes
101 (GL)	101	Unknown	Enabled	No	Yes	Yes	No	Yes	No
101 (SQ)	32045	Program - Optimizer	Enabled	Yes	No	No	No	No	Yes
178 (CX)	36662	Delete data from temporary table	Enabled	Yes	No	No	No	No	Yes

**Note:** The **PAM** check program exists alert will be implemented in a later tutorial

To validate your entry prior to the **PAM** check program exists check (CP-001) being run you can use the **PAM** config report **PAMC010-10 PAM Program Monitor Validate:**

Example - **PAMC010-10 PAM Program Monitor Validate** report

PAMC010-10		PIPER-RX - APPLICATION MONITOR PROGRAM MONITOR LIST - VALIDATION As at 23-Mar-10 09:20:24 For PROD 12i			PIPER - Rx	
Prog Appn	Prog Id	Program Name	Validation			
0	0	Activate Concurrent Manager	Program Exists			
0	1	Deactivate Concurrent Manager	Program Exists			
0	3	Restart Concurrent Manager	Program Exists			
0	4	Abort Concurrent Manager	Program Exists			
0	5	Shutdown Concurrent Manager	Program Exists			
0	6	Startup Concurrent Manager	Program Exists			
0	31659	Report Set	Program Exists			
0	32263	Purge Concurrent Request and/or Manager	Program Exists			
0	32592	Purge Signon Audit data	Program Exists			
0	36034	Request Set Stage	Program Exists			
0	36888	Workflow Background Process	Program Exists			
101	101	* Posting	Program Exists			
101	32045	Program - Optimizer	Program Exists			
178	36662	Delete data from temporary table	Program Exists			

\* Indicates the program name has not been updated by the program monitor  
This will occur the next time the program monitor runs

### 5.5.3 Deleting a **PAM** exceptions program

A concurrent program can be removed from the **PAM** program exception list by using the following **PAM** API:

```
exec PIPER_RX_PAM_API.PAM_MONITORED_PROGRAM_DEL ( 101, 101);
```

Parameter 1: The program application\_id

Parameter 2: The concurrent program id

### 5.5.4 Changing a PAM monitored programs items

The PAM monitored programs settings can be changed by first deleting the existing entry and then adding the program with the revised parameters.

### 5.5.5 Referential Integrity

After a period of time you may have a number of programs in the PAM monitored program list that no longer exist. The following PAM API performs a PAM referential cleanup which will remove any program from the PAM exceptions list that no longer exists in the OEBS application.

```
exec PIPER_RX_PAM_API.PAM_REFERENTIAL_CLEANUP;
```

## 6 PAM auto purging feature

PAM provides an auto purge feature to remove PAM collected data after a defined period of time. This is based on the understanding that historical monitoring data's value diminishes over time so there is no real need to keep that data. What we don't want is the PAM repository growing too large and interfering with the "self managing" and "set and forget" design principles of PAM.

The following PAM tables are purged as part of the PAM auto purge process:

### PIPER\_RX\_PAM\_DEBUG

This table holds PAM debug information. Once PAM debug has been enabled this table will be populated at a rate of one record per individual PAM check. The default number of day's history to be held online is set to 5 days.

The number of days PAM debug history to be held on-line can be changed using the following PAM API:

```
exec PIPER_RX_PAM_API.PAM_DEBUG_PURGE_DAYS_SET ( 5 );
```

### PIPER\_RX\_PAM\_ERRORS

This table holds **PAM** runtime error information. This table will be populated at a rate of one record per **PAM** error encountered when running a **PAM** stored procedure. The default number of day's history to be held online is set to 10 days.

The number of days **PAM** error history to be held on-line can be changed using the following **PAM** API:

```
exec PIPER_RX_PAM_API.PAM_ERROR_SET_PURGE_DAYS ( 10 );
```

### PIPER\_RX\_PAM\_ALERTS

This table holds the alerts generated by **PAM** and currently defaults to 365 days.

The number of days **PAM** alert history to be held on-line can be changed using the following **PAM** API:

```
exec PIPER_RX_PAM_API.PAM_ALERT_PURGE_DAYS_CHANGE ( 365 );
```

### PIPER\_RX\_PAM\_IDWA\_MONITOR\_TL

This table holds the **Intraday Workflow Activity** (IDWA) data which is populated at the rate of one (1) record every 15 minutes (default) which equated to approximately 35,000 records per year (default purge data older than 1 year).

The number of days **PAM** intraday workflow history is to be held on-line can be changed using the following **PAM** API:

```
exec PIPER_RX_PAM_API_2.PAM_IDWA_HIST_PURGE_DAYS_SET ( 365 );
```

### PIPER\_RX\_PAM\_DAILY\_ACTIVITY

This table holds the daily application activity data which is populated at a rate of one (1) record per day which equates to 365 records per year (default purge data older than 1.5 years).

The number of days **PAM** daily activity history to be held on-line can be changed using the following **PAM** API:

```
exec PIPER_RX_PAM_API.PAM_SETTINGS_DA_PURGE_DAY_SET ( 540 );
```

### PIPER\_RX\_PAM\_IDA\_ACTIVITY

This table holds the application intraday activity which is populated at the rate of one (1) record every 15 minutes (default) which equates to approximately 35,000 records per year (default purge data older than 1 year).

The number of days **PAM** intraday application activity history to be held on-line can be changed using the following **PAM** API:

```
exec PIPER_RX_PAM_API_2.PAM_IDAA_PURGE_DAYS_SET ( 180 );
```

### PIPER\_RX\_PAM\_RT\_HISTORY

This table holds the intra day application response time data which is populated at a rate of one (1) record every 10 minutes which equates to 144 records per day (default purge data older than 6 months).

The number of days **PAM** application response time history to be held on-line can be changed using the following **PAM** API:

```
exec PIPER_RX_PAM_API_2.PAM_RT_HISTORY_PURGE_DAYS_SET ( 180 );
```

### PIPER\_RX\_PAM\_AT\_HISTORY

This table holds the historical auto threshold data which is populated at a rate of one (1) record per week which equates to 52 records per year (default purge data older than 2 years).

The number of days **PAM** auto threshold history to be held on-line can be changed using the following **PAM** API:

```
exec PIPER_RX_PAM_API_2.PAM_AT_PURGE_DAYS_SET ( 365 );
```

## 7 Disclaimer

All material contained in this document is provided by the author "as is" and any express or implied warranties, including, but not limited to, any implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the author be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of any content or information, even if advised of the possibility of such damage. It is always recommended that you seek independent, professional advice before implementing any ideas or changes to ensure that they are appropriate.

*Oracle®, Oracle Applications® & Oracle E-Business Suite® are registered trademarks of  
Oracle Corporation  
TOAD® is a registered trademark of Quest Software*