PAMtutorials 15: Customising *PAM*

Adding your own monitors & alerts

PIPER-Rx *A*pplication *M*onitor – *PAM*
VIRTUAL APPS ADMINISTRATOR

PAM Version 4.0

"Blurring the line between software product and training"

May 2012

Table of Contents

1 What you'll get out of *PAMtutorials* 15 4
2 Installing the *PAM* custom examples 5
3 Creating a *PAM* custom e-mail alert 6
4 Adding your own *PAM* checks and alerts 8
4.1 How it all works 8
4.2 Setting up your *PAM* custom environment 9
4.2.1 *PAM* custom alert category 9
4.2.2 *PAM* Reports Category 9
4.3 The *PAM* configuration entry 10
4.3.1 *PAM* Config entry structure 10
4.3.2 Custom alert ids 13
4.3.3 Foreign key valid values 13
4.3.4 Alert hour of day 14
4.3.5 Alert action id 15
4.4 Associated report 16
5 Example 1: CUST-001 – Alerts that continue to notify you until a condition changes 17
5.1 Alert SQL 18
5.2 Create your PLSQL package 19
5.3 *PAM* Config Entry 20
5.4 Example e-mail alert message 21
5.5 Alert report 22
6 Example 2: CUST-002 – Alerts that notify you when a monitored item has occurred since the last *PAM* check 23
6.1 Alert SQL 24
6.2 *PAM* Config Entry 24
6.3 Create your PLSQL package 26
6.4 Example Alert 27
6.5 Alert report 28
7 Example 3: CUST-003 Alerts that notify you if a monitored item exceeds its last high point during the day 29
7.1 Alert SQL 30
7.2 *PAM* Config Entry 31
7.3 Create your PLSQL package 32
7.4 Example Alert 34
7.5 Alert report 35
8 Example 4: CUST-004 – Alerts that notify you if a monitored item occurs (once per request) 36
8.1 Alert SQL 37

8.2 *PAM* Config Entry 38
8.3 Create your PLSQL package 39
8.4 Example Alert 41
8.5 Alert Report 42
9 Other customisations 43
9.1 Changing the *PAM* alert severity colors 43
9.2 Setting the *PAM* alert e-mail background colour 44
9.3 Resetting the email colours to the default values 44
9.4 Changing the *PAM* alert e-mail links 44
9.4.1 Adding a web link 45
9.4.2 Deleting an alert e-mail link 46
9.4.3 Disabling one or more alert e-mail links 46
9.5 *PAM* alert e-mail routings 47
9.5.1 Adding a new *PAM* alert e-mail route 47
9.5.2 Deleting a *PAM* alert e-mail route 48
10 Disclaimer 49

1 What you'll get out of *PAMtutorials* 15

PAMtutorials 15 will cover how to build your own custom monitors into ***PAM***. We will walk through a series of four (4) examples.

We will also cover how to customise the look of your ***PAM*** e-mail alerts and how to configure new ***PAM*** e-mail routes.

2 Installing the *PAM* custom examples

To install the ***PAM*** Custom examples, from the ***PAM*** source directory run the file:

PIPER_RX_PAM_CUSTOM_INSTALL.sql

To remove the ***PAM*** custom examples, from the ***PAM*** source directory run the file:

PIPER_RX_PAM_CUSTOM_UNINSTALL.sql

WARNING: The uninstall process will remove all ***PAM*** configuration entries that start with the “CUST-0%”. It is therefore recommended that you do not use ‘CUST’ for your own customisations.

3 Creating a *PAM* custom e-mail alert

The simplest method for generating a ***PAM*** alert e-mail is to add a record into the ***PAM*** alerts table piper_rx_pam_alerts; ***PAM*** will generate an e-mail alert based on the entry in the table. It's that simple…. (Well almost, you do have to write your own SQL and process for inserting the record into the ***PAM*** table…)

The following SQL will add an entry into the ***PAM*** piper_rx_pam_alerts table:

```
INSERT into piper_rx_pam_alerts
       ( alert_time,
         alert_id,
         alert_severity,
         alert_details )
       VALUES ( sysdate,
                'CUST-001',
                'I',
                'Alert Message Goes Here');
```

Example *PAM* e-mail alert message

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

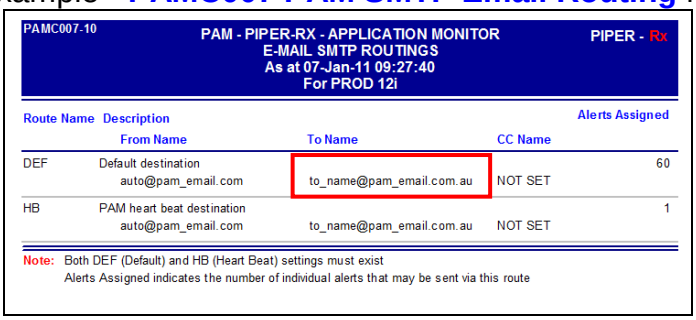
Company = Company name
Site = Site name
Alert Level = **Informational**
Detected = 06-Jan-11 (Thu) 12:32:08
Alert Frequency = Unknown

Alert Message Goes Here

Notes:

- The alert severity must be uppercase 'I' for Informational, 'W' for Warning and 'C' for Critical
- The notification will be sent to the default ***PAM*** e-mail recipient only; you cannot select a different e-mail recipient using this method. You can use ***PAMreports*** – Config PAMC007 PAM SMTP Email Routing report to identify the default e-mail account.

Example - **PAMC007 PAM SMTP Email Routing** report



PAMC007-10

PAM - PIPER-RX - APPLICATION MONITOR
E-MAIL SMTP ROUTINGS
As at 07-Jan-11 09:27:40
For PROD 12i

PIPER - Rx

Route Name	Description / From Name	To Name	CC Name	Alerts Assigned
DEF	Default destination auto@pam_email.com	to_name@pam_email.com.au	NOT SET	60
HB	PAM heart beat destination auto@pam_email.com	to_name@pam_email.com.au	NOT SET	1

Note: Both DEF (Default) and HB (Heart Beat) settings must exist
Alerts Assigned indicates the number of individual alerts that may be sent via this route

4 Adding your own *PAM* checks and alerts

In this tutorial we will show you how to add your own structured custom checks and alerts using the ***PAM*** collector to run your checks.

Step 1 - Decide:

- What are you going to check
- What will be the threshold value
- How often should that check be run
- If the alert is to be run once per day or less frequently and what hour of the day should it be run
- The e-mail message
- Do you want alert message information in the email

Step 2 – Write your check SQL, stored procedure and ***PAM*** config entry

Step 3 – Create a ***PAM*** report

In this tutorial we will cover each of these steps in detail via four (4) customisation examples.

4.1 How it all works

The ***PAM*** collector piper_rx_pam_collector.run_collections scans the ***PAM*** configuration table piper_rx_pam_config and will run any procedure where the time between the last_check_date and the current time has exceeded the ***PAM*** checks alert frequency. Where an alert has been raised the ***PAM*** e-mail package piper_rx_pam_sendmail.generate_e_mail_alert will direct an e-mail alert to the designated target.

So, all you have to do is:

- Create a ***PAM*** customisation category – One time only activity
- Create a ***PAM*** reports customisation category – One time only activity
- Create a custom package that performs the check/s and on exception writes a record into the ***PAM*** piper_rx_pam_alerts table.
- Create an entry in the ***PAM*** config piper_rx_pam_config table so the ***PAM*** collector can run your package at your defined frequency.
- Optional: Add a descriptive message to the ***PAM*** e-mail alert.

PAM does the rest….

Provided with this tutorial is an example package and a ***PAM*** report that will create a ***PAM*** reports customisation category.

4.2 Setting up your *PAM* custom environment

As with any customisations, you need to set up a customisation area within the ***PAM*** application. It is actually very simple; all you need to do is create a ***PAM*** custom alerts category and a ***PAM*** custom reports category in the TOAD® Reports Manager application to separate your custom reports from the ***PAM*** reports.

4.2.1 *PAM* custom alert category

Before you do anything you must set up a ***PAM*** alerts category reference value. This is a one off step and must be completed prior to setting up any custom ***PAM*** checks.

We recommend that you choose a category name like "**CUST"** or a few letters of your company name and add this entry to the ***PAM*** lookups table piper_rx_pam_lookups

```
INSERT into piper_rx_pam_lookups VALUES ( 'ALERT_CATEGORIES', 'CUST',
'Custom Alerts');
```

Note: Do not use any of the existing ***PAM*** alert categories

Note: The install script associated with this tutorial will have created this entry for you

4.2.2 *PAM* Reports Category

To make life simple we have provided a TOAD® Reports Manger report that will create a ***PAM*** custom reports category and will add one example report to that category.

For more information on the TOAD® Reports Manager and how to:

- Create reports and report categories
- Create charts
- Call PLSQL packages from a report
- Change report names and categories

etc…

Refer to the PIPER-Rx web site:
http://www.piper-rx.com/pages/reports_lessons.html

4.3 The *PAM* configuration entry

The ***PAM*** config table piper_rx_pam_config provides the ***PAM*** collector with information on what should be run and when. The entry also holds the ***PAM*** threshold values for the check and the ***PAM*** e-mail alert routing details, i.e. who should receive the alert email.

4.3.1 *PAM* Config entry structure

Before creating your own customisations you should become familiar with the structure of the ***PAM*** config table piper_rx_pam_config outlined below:

alert_id – varchar2(20) - not null

> **Primary key**. This value is generally the same value as the alert_category value with an additional sequence number e.g. If the category is "CUST" then the alert id should be set to "CUST-***nnn"*** where ***nnn*** is a sequence number.

alert_category – varchar2(5) - not null

> **Foreign key (piper_rx_pam_lookups).** The alert_category is a value that logically groups like alerts together, e.g. WF – are all the workflow related alerts, UA – User activity alerts etc…

alert_description – varchar2(200) - not null

> A free format description of the ***PAM*** alert.

alert_package_name – varchar2(30) - not null

> The package name the ***PAM*** collector is to run.

alert_package_body – varchar2(30) - not null

> The package body the ***PAM*** collector is to run.

alert_active_flag – varchar2(2) - not null

The *alert_active_flag* specifies if the ***PAM*** collector is to run this ***PAM*** check (Y = Run this check N = Do not run this check)

alert_frequency – number(10) – not null

The alert_frequency in conjunction with the alert_frequency_units value determine the frequency with which the ***PAM*** check should be run.

alert_frequency_units – varchar2(5) – not null

Foreign key (piper_rx_pam_lookups). The alert_frequency_units in conjunction with the alert_frequency value determine the frequency with which the ***PAM*** check should be run.

alert_hour_of_day – number(2)

Where a ***PAM*** alert is run on a daily, weekly or monthly basis this value defines the time of day the alert check should be run. In this way we can prevent the alert check being run during the normal business day.

last_check_date – date – not null

The date and time the ***PAM*** check was last executed. This value is used by the ***PAM*** collector to determine if the check is to be run. The check will be run where the time between the last_check_date and the current time has exceeded the ***PAM*** checks alert frequency.

alert_severity – varchar2(2) – not null

Foreign key (piper_rx_pam_lookups). The alert_severity value defines the ***PAM*** alerts severity: I = Informational, W = Warning, C = Critical

threshold_value_description - varchar2(200)

The free format description of the ***PAM*** threshold values.

threshold_varchar_value – varchar2(30)

The ***PAM*** threshold value for varchar values.

threshold_date_value - date

The ***PAM*** threshold value for date values.

threshold_numeric_value – number(20)

The ***PAM*** threshold value for numeric values.

working_value_description - varchar2(200)

The free format description of the ***PAM*** working values.

working_varchar_value – varchar2(30)

The ***PAM*** working value for varchar values.

working_date_value - date

The ***PAM*** working value for date values.

working_numeric_value – number(20)

The ***PAM*** working value for numeric values.

alert_action_id - varchar2(20)

Foreign key (piper_rx_pam_actions). References the html format message to be displayed with the alert e-mail.

email_route - varchar2(10) – not null – Default 'DEF'

Foreign key (piper_rx_pam_smtp_routing). Used to determine the ***PAM*** alert e-mail route.

Example *PAM* config insert statement

```
-- *R* Indicates required attribute

INSERT INTO piper rx pam config
       (alert_id,                  -- *R* Primary Key
        alert_category,            -- *R* Foreign Key (PIPER_RX_PAM_LOOKUPS )
        alert_description,         -- *R*
        alert_package_name,        -- *R*
        alert_package_body,        -- *R*
        alert_active_flag,         -- *R* Y = Run this collection N = Do not run this
collection
        alert_frequency,           -- *R*
        alert_frequency_units,     -- *R* Foreign Key (PIPER_RX_PAM_LOOKUPS )
        alert hour of day,
        last_check_date,           -- *R* Set this to sysdate
        alert severity,            -- *R* Foreign Key (PIPER_RX_PAM_LOOKUPS )
        threshold_value_description,
        threshold_varchar_value,
        threshold_date_value,
        threshold numeric value,
        working value description,
        working varchar value,
        working_date_value,
        working_numeric_value,
        alert_action_id,           -- Foreign Key (PIPER_RX_PAM_ACTIONS)
        email_route)               -- *R* Foreign Key (PIPER_RX_PAM_SMTP_ROUTING )
      VALUES ('CUST-001',
              'CUST',
              'Alert when the sysadmin account is in use',
              'PIPER_RX_PAM_CUSTOM_MONITOR',
              'EXAMPLE_ONE',
              'N',
              5,
              'MIN',
              5,
              sysdate,
              'I',
              null,
```

```
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                'DEF');

COMMIT;
```

Note: When adding a ***PAM*** config entry, set the alert_active_flag value to 'N' (do not run) until you have written and check your alert code package

4.3.2 Custom alert ids

Each ***PAM*** custom alert you create will require a **unique** alert id. We recommend that the alert be set to the alert category you created e.g. "CUST" followed by a sequence number e.g. "CUST-***nnn"*** where ***nnn*** is a sequence number.

4.3.3 Foreign key valid values

There are a number of piper_rx_pam_config attributes that have a set of foreign key references in the pam piper_rx_pam_lookups table, these are:

4.3.3.1 ALERT_FREQUENCY_UNITS

Valid values are:

- **MIN** – Every ***n*** Minutes
- **HR** - Every ***n*** Hours
- **DAY** - Every ***n*** Days
- **WK** - Every ***n*** Weeks
- **MON** - Every ***n*** Months

4.3.3.2 ALERT_SEVERITY

Valid values are:

- **I** – Informational
- **W** – Warning
- **C** - Critical

A list of valid values can be found using ***PAMreports*** – Config PAMC003 PAM Lookups report:

Example - **PAMC003 PAM Lookups** report

PAMC003-10 PAM - PIPER-RX - APPLICATION MONITOR PIPER - Rx
LOOKUP CODES
As at 05-Jan-11 14:29:20
For PROD 12i

Lookup Code	Meaning
Lookup Type: ALERT_CATEGORIES	
AT	Auto Threshold
CM	Concurrent Manager Alerts
CP	Concurrent Program Alerts
CR	Concurrent Request Alerts
CUST	Custom Alerts
DB	Database Alerts
DBA	Database Related Alerts
GA	General Application
HB	Heart Beat
IN	Internal
PF	Performance Alerts
RM	Remote Monitor
UA	User Activity Alerts
WF	Workflow Alerts
Lookup Type: ALERT_SEVERITIES	
C	Critical
I	Informational
W	Warning
Lookup Type: FREQUENCY_UNITS	
DAY	Day
HR	Hour

4.3.3.3 EMAIL_ROUTE

Each ***PAM*** config entry must have an e-mail route. The ***PAM*** config table piper_rx_pam_config has been defined such that it will add the default route of 'DEF' if you fail to add this value

When adding your own e-mail route you should ensure the route is a valid ***PAM*** route; a list of valid e-mail routes can be found using ***PAMreports*** – Config PAMC007 PAM SMTP Email Routing report.

Example - **PAMC007 PAM SMTP Email Routing** report

PAMC007-10 PAM - PIPER-RX - APPLICATION MONITOR PIPER - Rx
E-MAIL SMTP ROUTINGS
As at 05-Jan-11 14:30:44
For PROD 12i

Route Name	Description / From Name	To Name	CC Name	Alerts Assigned
DEF	Default destination auto@pam_email.com	to_name@pam_email.com.au	NOT SET	60
HB	PAM heart beat destination auto@pam_email.com	to_name@pam_email.com.au	NOT SET	1

Note: Both DEF (Default) and HB (Heart Beat) settings must exist
Alerts Assigned indicates the number of individual alerts that may be sent via this route

4.3.4 Alert hour of day

Where an alert is run on a daily, weekly or monthly cycle you will need to set the time of day the alert will run. ***PAM*** provides this feature as generally daily, weekly or monthly checks are more resource intensive. In this way we can make sure these checks are not run during the business day. This process also prevents ***PAM*** alerts from creeping forward in time with each execution.

The “hour of day” value is set in the piper_rx_pam_config.alert_hour_of_day attribute.

Note: Where an alert is run on a minute or hour frequency the alert_hour_of_day value is ignored. However it is a good practice to populate this value just in case you change the runtime frequency at a later date.

4.3.5 Alert action id

This value links the ***PAM*** alert entry to a ***PAM*** actions entry (action_code) in the ***PAM*** piper_rx_pam_actions repository. The information held in the piper_rx_pam_actions repository is added to the ***PAM*** e-mail alert. The format of this information must be in a HTML format to be displayed correctly in the ***PAM*** e-mail alert.

Example SQL for creating a ***PAM*** Action entry – Note the HTML FORMAT:

```
INSERT INTO PIPER_RX_PAM_ACTIONS
       VALUES ( 'GA-002', 1, '<B>GA-002 - Maintenance Mode</B><P>');
INSERT INTO PIPER RX PAM ACTIONS
       VALUES ( 'GA-002', 2, '<B>THE APPLICATION HAS BEEN PLACED IN MAINTENANCE
MODE</B><P>');
INSERT INTO PIPER_RX_PAM_ACTIONS
       VALUES ( 'GA-002', 3, 'No users can connect to the application whilst the
application is in maintenance mode<P>');
INSERT INTO PIPER RX PAM ACTIONS
       VALUES ( 'GA-002', 4, '<FONT COLOR=#FF0000><B>Note 1:</B></FONT> This alert
will continue to alert until the ');
INSERT INTO PIPER RX PAM ACTIONS
       VALUES ( 'GA-002', 5, 'application is no longer in maintenance mode. Subsequent
checks will occur on a less ');
INSERT INTO PIPER RX PAM ACTIONS
       VALUES ( 'GA-002', 6, 'frequent basis during the shutdown');

COMMIT;

UPDATE piper rx pam config
   SET alert_action_id = 'GA-002'
 WHERE alert_id = 'GA-002';

COMMIT;
```

When the alert action is linked to a ***PAM*** alert, the message will be displayed in the e-mail alert as shown within the red bordered area below:

Example *PAM* GA-002 – *PAM* Maintenance Mode e-mail alert message

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

Company = Company name
Site = Site name

Alert Level = **Critical**
Detected = 08-Jan-11 (Sat) 07:45:02
Alert Frequency = 5 Minutes

Maintenance Mode = MAINT – Maintenance Mode

Alert Information:

GA-002 - Maintenance Mode

THE APPLICATION HAS BEEN PLACED IN MAINTENANCE MODE

No users can connect to the application whilst the application is in maintenance mode

Note 1: This alert will continue to alert until the application is no longer in maintenance mode. Subsequent checks will occur on a less frequent basis during the shutdown

4.4 Associated report

Make no mistake; this is where all the complexity resides. Whilst the ***PAM*** check provides a "one line" alert, and optional short description (alert action) the associated report provides all the details and should be presented in a user readable format.

All ***PAM*** reports are written using Quest Software's TOAD® Reports Manager product which is part of the standard TOAD® product; however, there is no reason why your cannot use your own preferred reporting product.

For more information on writing TOAD® Reports Manager reports, refer to: http://www.piper-rx.com/pages/reports_lessons.html

5 Example 1: CUST-001 – Alerts that continue to notify you until a condition changes

In this first example we will build a check that will continue to send an alert at the designated frequency until the cause of the alert is corrected i.e. generate alerts whist a condition exists.

Examples of this form of alert include alerting when:

- The concurrent managers are down
- The application performance has exceeded a threshold value but has/ will decrease below the threshold when the performance issue has passed
- The number of connected users has exceeded the threshold value
- The application is in maintenance mode

***PAM* Custom Check Form**

Alert ID	CUST-001
Alert Description	Generate an alert when the **sysadmin** account is being accessed and continue to alert whilst the account is being accessed
Alert threshold	N/A
Alert Severity	I – Informational W – Warning ✓ C – Critical
Alert Frequency Value	10
Alert Frequency Units	**MIN** – Minutes ✓ **HR** – Hours **DAY** – Days **WK** – Weeks **MON** – Mon
Alert Hour of day	N/A (default to 5am)
Alert message	At 23-Jan-11 13:45 User Account SYSDAMIN was being Accessed
E-mail Route	Default
PLSQL Name	PIPER_RX_PAM_CUSTOM_MONITOR.EXAMPLE_ONE
Notes	1. The sign-on audit must be set to user or above **2.** In this example we will only be able to alert when

	the **sysadmin** account is being used at the time of the *PAM* check, if the account has been used during the period between *PAM* checks it will not alert.
Report Required	Yes
Report Details	List all **sysadmin** accesses for a selected day including the responsibilities and forms that were accessed during each individual session

The *PAM* Custom check form should contain all the information required to create the *PAM* custom check.

5.1 Alert SQL

For this custom check we need to count how many active SYSADMIN sessions there are; if the returned value is greater than zero (0) then there are active SYSADMIN sessions.

```
SELECT count(fl.login_id)
  FROM applsys.fnd_logins fl,
       applsys.fnd_user fu
 WHERE fl.user id = fu.user id
   and fu.user_name = 'SYSDAMIN'
   and fl.end_time is null
   and fl.start_time > trunc(sysdate);
```

In this case we have used the count function so as to prevent the “No Rows Returned” scenario.

Note: There are occasions where session has been terminated incorrectly and the session will not be updated with an end date and hence will be shown as active forever… To limit the number of alerts we have added the condition fl.start_time > trunc(sysdate) to show only those sessions stared today.

Yes, I know there may “better” SQL options for identifying active sessions, remember this is only one non database version specific example……

5.2 Create your PLSQL package

For this tutorial we have provided an example PLSQL package piper_rx_pam_custom_monitor.pkb with all 4 examples outlined in this tutorial. The following extract will describe how the package is structured for this alert.

Extract from piper_rx_pam_custom_monitor.pkb

```
CREATE OR REPLACE PACKAGE PIPER_RX_PAM_CUSTOM_MONITOR IS

   PROCEDURE EXAMPLE_ONE;    -- CUST-001

END PIPER_RX_PAM_CUSTOM_MONITOR;

/

CREATE OR REPLACE PACKAGE BODY PIPER_RX_PAM_CUSTOM_MONITOR AS

error_message varchar2(100);

PROCEDURE EXAMPLE_ONE IS

v_alert_id       varchar2(20) := 'CUST-001';

v user name      varchar2(30) := 'SYSADMIN';
v_user_id        number(10);
v_account_usage  number(10);

v alert severity varchar2(2);
v alert details  varchar2(200);
v_current_time   varchar2(30);

  BEGIN

    -- Check if the account currently being used
    SELECT count(fl.login id)
      INTO v_account_usage
      FROM applsys.fnd_logins fl,
           applsys.fnd user fu
     WHERE fl.user id = fu.user id
       and fu.user name = upper(v_user_name)
       and fl.end time is null
       and fl.start_time > trunc(sysdate);

    IF ( v_account_usage > 0 ) THEN

          -- Get severity value from config
          SELECT nvl(alert_severity, 'I'),
                 to_char(sysdate, 'DD-Mon-YY HH24:
            INTO v_alert_severity,
                 v current time
            FROM piper rx pam config
           WHERE alert_id = v_alert_id;

          v_alert_details := 'At '||v_current_time
was being Accessed';

         INSERT INTO piper rx pam_alerts
                    ( alert_time,
                      alert_id,
                      alert_severity,
                      alert details)
              VALUES ( sysdate,
                       v alert id,
                       v_alert_severity,
                       v_alert_details);

        COMMIT;
```

Your custom alert id

The Application account to monitor

Your check SQL

This section obtains the alert severity from the ***PAM*** config table

Build your alert message

Inserts the alert record in to the ***PAM*** alerts table

```
    END IF;

    -- UPDATE CONFIG TO SET LAST RUN TIME
    UPDATE piper_rx_pam_config
       SET last_check_date = sysdate
     WHERE alert_id = v_alert_id;

    COMMIT;

    -- ****************************
    -- **       EXCEPTION        **
    -- ****************************
    EXCEPTION
     WHEN NO DATA FOUND THEN
        error message := 'No data Found';
       dbms_output.put_line('EXAMPLE_ONE: '||error_message);
     WHEN OTHERS THEN
        error message := 'Other problem';
       dbms_output.put_line('EXAMPLE_ONE: '||error_message);
       dbms_output.put_line('Oracle Error : '|| sqlerrm);

  END EXAMPLE_ONE;

END PIPER_RX_PAM_CUSTOM_MONITOR;

/
```

Updates the *PAM* config table with the current run date

5.3 *PAM* Config Entry

Using the data from the *PAM* Custom Check Form you can now create your *PAM* config entry:

Example *PAM* config entry

```
INSERT INTO piper_rx_pam_config
       (alert_id,
        alert_category,
        alert_description,
        alert_package_name,
        alert package body,
        alert_active_flag,
        alert_frequency,
        alert_frequency_units,
        alert_hour_of_day,
        last_check_date,
        alert_severity,
        threshold_value_description,
        threshold_varchar_value,
        threshold_date_value,
        threshold_numeric_value,
        working_value_description,
        working_varchar_value,
        working_date_value,
        working_numeric_value,
        alert_action_id,
        email_route)
       VALUES ('CUST-001',
               'CUST',
               'Alert when the sysadmin account is in use',
               'PIPER_RX_PAM_CUSTOM_MONITOR',
```

```
                'EXAMPLE_ONE',
                'N',
                10,
                'MIN',
                5,
                sysdate,
                'W',
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                null,
                'DEF');

COMMIT;
```

Once you are happy you have tested your PLSQL package you can enable you custom ***PAM*** alert by setting the value for alert_active_flag to 'Y' so your custom check will now be run by the ***PAM*** collector process.

5.4 Example e-mail alert message

Once ***PAM*** is running your alert and an exception is found, you will receive the following ***PAM*** alert e-mail message:

Example *PAM* CUST-001 – *PAM* SYSADMIN account e-mail alert message

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

Company = Company name
Site = Site name
Alert Level = **Informational**
Detected = 06-Jan-11 (Thu) 16:52:28
Alert Frequency = 10 Minutes

At 05-Jan-11 13:35 User Account SYSADMIN was being Accessed

Twitter: piper-rx - *Web:* www.piper-rx.com - *Mail:* pam@piper-rx.com -

---

5.5 Alert report

This is the more important part of your customisations. The report SQL and format will be more complicated as you need to provide the details of the alert with sufficient information to take action.

For this example, ***PAM*** will have alerted us to the fact that the SYSADMIN account is being accessed and when; what we want to know is what did they do when the account was accessed…Our report in this case will list all the occurrences that the SYSADMIN account has been used for a given day including the responsibilities and forms accessed during the session. For added flexibility, we will make the user account a report variable so we can use the report to audit any user's full service access.

PAMreports – General PAMRAA001 FS Activity Audit By User (day) shows the following result:

Example **PAMRAA001 FS Activity Audit By User (day)** report

PAMRAA001.10

PAM - PIPER-RX - APPLICATION MONITOR
FULL SERVICE SESSION AUDIT
(SYSADMIN) System administrator
For 05-Jan-11 (Wed)
as at 05-Jan-11 14:59
for PROD 12i

PIPER - Rx

Activity (Login ID)	Activity Start	Activity End	DD:HH:MI:SS
Connection (314715)	05-Jan-11 14:39	Active	00:00:19:57
System Administrator	05-Jan-11 14:39	05-Jan-11 14:41	00:00:01:10
Define Application User	05-Jan-11 14:39	05-Jan-11 14:40	00:00:00:14
Update System Profile Values	05-Jan-11 14:40	05-Jan-11 14:40	00:00:00:30
Application Developer	05-Jan-11 14:41	Active	00:00:18:41
Define Key Flexfield Segments	05-Jan-11 14:41	05-Jan-11 14:42	00:00:00:37
Run Reports	05-Jan-11 14:42	Active	00:00:17:30
Connection (314714)	05-Jan-11 14:35	05-Jan-11 14:37	00:00:01:13
System Administrator	05-Jan-11 14:35	05-Jan-11 14:37	00:00:01:09
Monitor Application Users	05-Jan-11 14:36	05-Jan-11 14:36	00:00:00:14
Run Reports	05-Jan-11 14:36	05-Jan-11 14:37	00:00:00:40

In this example you can see the user SYSADMIN connected at 14:35 (login id 314714) as "Systems Administrator" and used two screens "Monitor Application Users" and "Run Reports" and then logged out. SYSADMIN logged in again at 14:39 (login id 314715) connected to the "Systems Administrator" responsibility used two screens "Define Application User" and "Update System Profile Values" and then switched responsibility to "Application Developer", used the screen "Define Key Flexfield Segments" and was still in the "Run Reports" screen at the time of the report.

6 Example 2: CUST-002 – Alerts that notify you when a monitored item has occurred since the last *PAM* check

In this second example we will build a check that will send an alert if the SYSADMIN account has been accessed since the last ***PAM*** alert check.

Unlike the first check where the account has to be in use when ***PAM*** runs its check, this check will provide an alert if the SYSADMIN account has been used at any time since the last ***PAM*** check.

Examples of this form of alert include alerting when:

- A specific user has connected to the application
- A specific responsibility or form has been accessed
- A specific concurrent program has been run

***PAM* Custom Check Form**

Alert ID	CUST-002
Alert Description	Generate an alert when the **sysadmin** account has been accessed
Alert threshold	N/A
Alert Severity	I - Informational: ; W – Warning: ✓ ; C - Critical:
Alert Frequency Value	10
Alert Frequency Units	**MIN** - Minutes: ✓ ; **HR** – Hours: ; **DAY** - Days: ; **WK** - Weeks: ; **MON** - Mon:
Alert Hour of day	N/A (default to 5am)
Alert message	The SYSDAMIN has been accessed 2 time/s since 08-Jan-11 13:45
E-mail Route	Default
PLSQL Name	PIPER_RX_PAM_CUSTOM_MONITOR.EXAMPLE_TWO
Notes	The sign-on audit must be set to user or above
Report Required	Yes
Report Details	List all **sysadmin** accesses for a selected day including

	the responsibilities and forms that were accessed during each individual session

The *PAM* Custom Check Form should contain all the information required to create the *PAM* custom check.

6.1 Alert SQL

For this custom check we need to count how many active SYSADMIN sessions there have been since the last time the *PAM* check was run; if the returned value is greater than zero (0) then the SYSADMIN account has been accessed.

```
SELECT count(fl.login_id)
  FROM applsys.fnd_logins fl,
       applsys.fnd_user fu
 WHERE fl.user_id = fu.user_id
   and fu.user_name = 'SYSDAMIN'
   and start_time >= v_last_check_date;
```

Again, in this case we have used the count function so as to prevent the “No Rows Returned” scenario.

The value for v_last_check_date is the date and time of the last time *PAM* ran. As we update the *PAM* config entry last_check_date with the date and time the *PAM* check was run, we can use this value as our reference point.

The last_check_date can be found using the following SQL:

```
SELECT last_check_date
  FROM piper_rx_pam_config
 WHERE alert_id = 'CUST-002';
```

6.2 *PAM* Config Entry

In this case the config entry must exist prior to the package build as we need to reference the last_check_date so as to pick up any new access to the SYSADMIN account since the prior *PAM* run.

Where the *PAM* config entry is created before the package it is **very** important to set the value for alert_active_flag to ‘N’ so as *PAM* does not attempt to run the package until you have fully tested your package.

If *PAM* does run the package prior to the package existing or whilst the package is in an invalid state, *PAM* will record that the package is causing

errors and as part of the *PAM* internal protection feature (IN-012), after a number of errors have been recorded *PAM* will disable the check for the remainder of the day.

Using the data from the *PAM* Custom Check Form you can now create your *PAM* config entry:

Example *PAM* config entry

```
INSERT INTO piper_rx_pam_config
       (alert_id,
        alert_category,
        alert_description,
        alert_package_name,
        alert_package_body,
        alert_active_flag,
        alert_frequency,
        alert_frequency_units,
        alert_hour_of_day,
        last_check_date,
        alert_severity,
        threshold_value_description,
        threshold_varchar_value,
        threshold_date_value,
        threshold_numeric_value,
        working_value_description,
        working_varchar_value,
        working_date_value,
        working_numeric_value,
        alert_action_id,
        email_route)
       VALUES ('CUST-002',
               'CUST',
               'Alert when the sysadmin has been used since prior
check',
               'PIPER_RX_PAM_CUSTOM_MONITOR',
               'EXAMPLE_TWO',
               'N',
               10,
               'MIN',
               5,
               sysdate,
               'W',
               null,
               null,
               null,
               null,
               null,
               null,
               null,
               null,
               null,
               'DEF');

COMMIT;
```

Once you are happy you have tested your PLSQL package you can enable you custom *PAM* alert by setting the value for alert_active_flag to 'Y' so your custom check will now be run by the *PAM* collector process.

6.3 Create your PLSQL package

For this tutorial we have provided an example PLSQL package piper_rx_pam_custom_monitor.pkb with all 4 examples outlined in this tutorial. The following extract will describe how the package is structured for this alert.

Extract from piper_rx_pam_custom_monitor.pkb

```
CREATE OR REPLACE PACKAGE PIPER_RX_PAM_CUSTOM_MONITOR IS

     PROCEDURE EXAMPLE_TWO;    -- CUST-002

END PIPER_RX_PAM_CUSTOM_MONITOR;

/

CREATE OR REPLACE PACKAGE BODY PIPER_RX_PAM_CUSTOM_MONITOR AS

error_message varchar2(100);

PROCEDURE EXAMPLE_TWO IS

v_alert_id        varchar2(20) := 'CUST-002';

v user name       varchar2(30) := 'SYSADMIN';
v_account_usage   number(10);

v_alert_severity  varchar2(2);
v_alert_details   varchar2(200);
v_last_check_date date;
v last check date formatted varchar2(30);

  BEGIN

   -- Get severity value and last check date from config
    SELECT nvl(alert_severity, 'I'),
           last check date,
           to char(last check date, 'DD-Mon-YY HH24:M
      INTO v alert severity,
           v_last_check_date,
           v_last_check_date_formatted
      FROM piper_rx_pam_config
     WHERE alert_id = v_alert_id;

    -- Check if the account has been used since the last check
    SELECT count(fl.login_id)
      INTO v_account_usage
      FROM applsys.fnd logins fl,
           applsys.fnd user fu
     WHERE fl.user id = fu.user id
       and fu.user_name = upper(v_user_name)
       and start_time >= v_last_check_date;

    IF ( v_account_usage > 0 ) THEN

           v alert details := 'The '||v user name||' account has been accessed
'||v_account_usage||' time/s since '||v_last_check_date_formatted;

           INSERT INTO piper_rx_pam_alerts
                    ( alert_time,
```

Your custom alert id

The Application account to monitor

This section obtains the last checked dated alert severity from the *PAM* config table

Your check SQL

Build your alert message

Inserts the alert record in to the *PAM* alerts table

```
                        alert id,
                        alert severity,
                        alert details)
                VALUES ( sysdate,
                          v_alert_id,
                          v_alert_severity,
                          v_alert_details);

            COMMIT;

    END IF;

    -- UPDATE CONFIG TO SET LAST RUN TIME
    UPDATE piper rx pam config
       SET last check date = sysdate
     WHERE alert_id = v_alert_id;

    COMMIT;

    -- ****************************
    -- **       EXCEPTION        **
    -- ****************************
    EXCEPTION
     WHEN NO_DATA_FOUND THEN
        error message := 'No data Found';
       dbms_output.put_line('EXAMPLE_TWO: '||error_message);
     WHEN OTHERS THEN
        error_message := 'Other problem';
       dbms_output.put_line('EXAMPLE_TWO: '||error_message);
       dbms_output.put_line('Oracle Error : '|| sqlerrm);

  END EXAMPLE_TWO;

END PIPER_RX_PAM_CUSTOM_MONITOR;

/
```

Updates the *PAM* config table with the current run date

6.4 Example Alert

Once *PAM* is running your alert and an exception is found you will receive the following *PAM* alert e-mail message:

Example *PAM* CUST-002 – *PAM* SYSADMIN account e-mail alert message

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

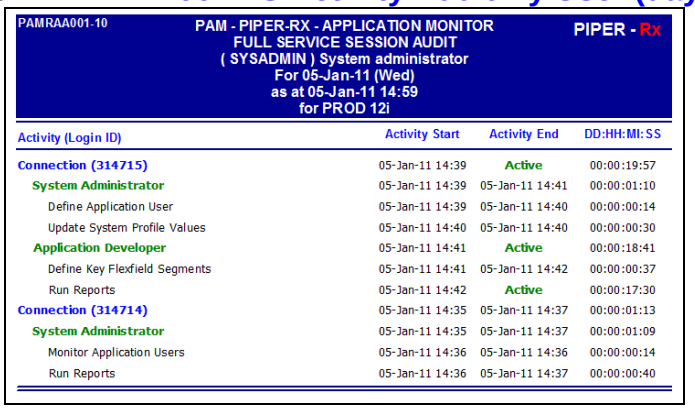
Company = Company name
Site = Site name
Alert Level = **Informational**
Detected = 06-Jan-11 (Thu) 16:52:28
Alert Frequency = 10 Minutes

The SYSADMIN account has been accessed 2 time/s since 06-Jan-11 (Thu) 16:40:03

6.5 Alert report

The same report can be used as described in Example 1 above:

Example - **PAMRAA001 FS Activity Audit By User (day)** report



PAMRAA001-10

PAM - PIPER-RX - APPLICATION MONITOR
FULL SERVICE SESSION AUDIT
(SYSADMIN) System administrator
For 05-Jan-11 (Wed)
as at 05-Jan-11 14:59
for PROD 12i

PIPER - Rx

Activity (Login ID)	Activity Start	Activity End	DD:HH:MI:SS
Connection (314715)	05-Jan-11 14:39	Active	00:00:19:57
System Administrator	05-Jan-11 14:39	05-Jan-11 14:41	00:00:01:10
Define Application User	05-Jan-11 14:39	05-Jan-11 14:40	00:00:00:14
Update System Profile Values	05-Jan-11 14:40	05-Jan-11 14:40	00:00:00:30
Application Developer	05-Jan-11 14:41	Active	00:00:18:41
Define Key Flexfield Segments	05-Jan-11 14:41	05-Jan-11 14:42	00:00:00:37
Run Reports	05-Jan-11 14:42	Active	00:00:17:30
Connection (314714)	05-Jan-11 14:35	05-Jan-11 14:37	00:00:01:13
System Administrator	05-Jan-11 14:35	05-Jan-11 14:37	00:00:01:09
Monitor Application Users	05-Jan-11 14:36	05-Jan-11 14:36	00:00:00:14
Run Reports	05-Jan-11 14:36	05-Jan-11 14:37	00:00:00:40

7 Example 3: CUST-003 Alerts that notify you if a monitored item exceeds its last high point during the day

In this example we want to report if the number of concurrent requests that complete with a status of warning exceeds a threshold value and **not** continue to report until more warning requests have been found during the day.

In this example we will use the config entry threshold value for the base threshold value and the working_value to hold the last number of warning requests found (in that way we can use the working value as the threshold value). At the beginning of each day we will set the daily level back to the starting threshold level.

Examples of this form of alert include alerting when:

- A value increases during the day
- A performance indictor exceeds its last high value for the day

***PAM* Custom Check Form**

Alert ID	CUST-003
Alert Description	Generate an alert when the number of warning requests exceeds the threshold value and report again when more warning requests are found
Alert threshold	Base threshold = 5
Alert Severity	**I** - Informational ✓ **W** – Warning **C** - Critical
Alert Frequency Value	10
Alert Frequency Units	**MIN** - Minutes ✓ **HR** – Hours **DAY** - Days **WK** - Weeks **MON** - Mon
Alert Hour of day	N/A (default to 5am)
Alert message	***nn*** requests with a status of completed warning have exceeded the threshold value of ***nn***

E-mail Route	Default
PLSQL Name	PIPER_RX_PAM_CUSTOM_MONITOR.EXAMPLE_THREE
Notes	
Report Required	Yes
Report Details	List all Warning requests including the requestor for a given day

The ***PAM*** Custom Check Form should contain all the information required to create the ***PAM*** custom check.

7.1 Alert SQL

For this custom check we need to count how many requests have completed with a status of Warning (status_code = 'G') today.

```
SELECT count(*)
  FROM applsys.fnd_concurrent_requests fcr
 WHERE fcr.status_code = 'G'
   and fcr.actual_completion_date > trunc(sysdate);
```

The trick is at the beginning of each day to set the working value to the threshold value and then use the working value as the check threshold value incrementing the working value as the number warning requests increases during the day.

Firstly, at the beginning of each day we need to reset the working value to the threshold value:

```
  UPDATE piper_rx_pam_config
     SET working_numeric_value = threshold_numeric_value
   WHERE trunc(last_check_date) != trunc(sysdate)
     and alert_id = 'CUST-003';
```

The next step is to compare the current number of warning requests to the working value:

```
  SELECT threshold_numeric_value,
         working_numeric_value,
         alert_severity
    FROM piper_rx_pam_config
    WHERE alert_id = 'CUST-003';
```

If the working value has been exceeded then raise an alert.

Finally, set the working value to the current number of warning requests found:

```
    UPDATE piper_rx_pam_config
       SET last_check_date = sysdate,
           working_numeric_value =
                   greatest( v_threshold_varchar_value,
                             v_warning_request_count )
     WHERE alert_id = v_alert_id; WHERE alert_id = 'CUST-003';
```

With this method you will only be alerted when the number of warning requests exceeds the working value, and you will be alerted again throughout the day when more warning requests are found.

7.2 *PAM* Config Entry

Again in this case the config entry must exist prior to the package build as we need to reference the threshold_numeric_value and working_numeric_value etc…

Where the ***PAM*** config entry is created before the package it is **very** important to set the value for alert_active_flag to 'N' so as ***PAM*** does not attempt to run the package until you have fully tested your package.

If ***PAM*** does run the package prior to the package existing or whilst the package is in an invalid state, ***PAM*** will record that the package is causing errors and as part of the ***PAM*** internal protection feature (IN-012), after a number of errors have been recorded ***PAM*** will disable the check for the remainder of the day.

Using the data from the ***PAM*** Custom check form you can now create your ***PAM*** config entry:

Example *PAM* config entry

```
INSERT INTO piper_rx_pam_config
       (alert_id,
        alert_category,
        alert_description,
        alert_package_name,
        alert_package_body,
        alert_active_flag,
        alert_frequency,
        alert_frequency_units,
        alert_hour_of_day,
        last_check_date,
        alert_severity,
        threshold_value_description,
        threshold_varchar_value,
        threshold_date_value,
```

```
         threshold_numeric_value,
         working_value_description,
         working_varchar_value,
         working_date_value,
         working_numeric_value,
         alert_action_id,
         email_route)
       VALUES ('CUST-003',
               'CUST',
               'Alert when the number of complete warning requests
exceeds the threshold value',
               'PIPER_RX_PAM_CUSTOM_MONITOR',
               'EXAMPLE_THREE',
               'N',
               10,
               'MIN',
               5,
               sysdate,
               'W',
               'Competed Warning',
               null,
               null,
               5,
               'Intraday threshold',
               null,
               null,
               5,
               null,
               'DEF')
COMMIT;
```

Once you are happy you have tested your PLSQL package you can enable you custom **PAM** alert by setting the value for alert_active_flag to 'Y' so your custom check will now be run by the **PAM** collector process.

7.3 Create your PLSQL package

For this tutorial we have provided an example PLSQL package piper_rx_pam_custom_monitor.pkb with all 4 examples outlined in this tutorial. The following extract will describe how the package is structured for this alert:

Extract from piper_rx_pam_custom_monitor.pkb

```
CREATE OR REPLACE PACKAGE PIPER_RX_PAM_CUSTOM_MONITOR IS

   PROCEDURE EXAMPLE_THREE; -- CUST-003

END PIPER_RX_PAM_CUSTOM_MONITOR;

/

CREATE OR REPLACE PACKAGE BODY PIPER_RX_PAM_CUSTOM_MONITOR AS

error_message varchar2(100);

PROCEDURE EXAMPLE_THREE IS
```

```
v_alert_id                varchar2(20) := 'CUST-003';

v_warning_request_count   number(10);

v_alert_severity          varchar2(2);
v_threshold_numeric_value varchar2(30);
v working numeric value   varchar2(30);
v alert details           varchar2(200);

  BEGIN

   --Update the working value with the threshold value at the beginning of each day
   UPDATE piper_rx_pam_config
      SET working numeric value = threshold numeric_value
    WHERE trunc(last check date) != trunc(sysdate)
      and alert_id = v_alert_id;


    --Get severity and threshold values
    SELECT nvl(alert severity, 'I'),
           threshold numeric value,
           working numeric value
      INTO v_alert_severity,
           v_threshold_numeric_value,
           v working numeric value
      FROM piper rx pam config
     WHERE alert_id = v_alert_id;

    --Get the number of completed warning request generated today
    SELECT count(*)
      INTO v warning request count
      FROM applsys.fnd concurrent_requests fcr
     WHERE fcr.status code = 'G'
       and fcr.actual_completion_date > trunc(sysdate);

    IF (v warning request count > greatest( v_threshold_numeric_value,
v_working_numeric_value ) ) THEN

           v_alert_details := v_warning_request_count||' requests with a status of
completed warning have exceeded the threshold value of '||v_threshold_numeric_value;

           INSERT INTO piper rx pam_alerts
                     ( alert time,
                       alert id,
                       alert_severity,
                       alert_details)
                VALUES ( sysdate,
                         v alert id,
                         v alert severity,
                         v_alert_details);

            COMMIT;

    END IF;

    -- UPDATE CONFIG TO SET LAST RUN TIME
    UPDATE piper_rx_pam_config
       SET last_check_date = sysdate,
           working numeric_value = greatest( v_threshold_numeric_value,
v warning request count )
     WHERE alert_id = v_alert_id;

    COMMIT;

    -- ****************************
    -- **       EXCEPTION        **
    -- ****************************
    EXCEPTION
     WHEN NO_DATA_FOUND THEN
        error_message := 'No data Found';
       dbms_output.put_line('EXAMPLE_THREE: '||error_message);
     WHEN OTHERS THEN
        error_message := 'Other problem';
```

```
        dbms_output.put_line('EXAMPLE_THREE: '||error_message);
        dbms_output.put_line('Oracle Error : '|| sqlerrm);

  END EXAMPLE_THREE;

END PIPER_RX_PAM_CUSTOM_MONITOR;

/
```

7.4 Example Alert

Once *PAM* is running your alert and an exception is found you will receive the following *PAM* alert e-mail message:

Example *PAM* CUST-003 – *PAM* High point exceeded e-mail alert message

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

Company = Company name
Site = Site name
Alert Level = **Informational**
Detected = 06-Jan-11 (Thu) 17:00:14
Alert Frequency = 10 Minutes

---

23 requests with a status of completed warning have exceeded the threshold value of 10

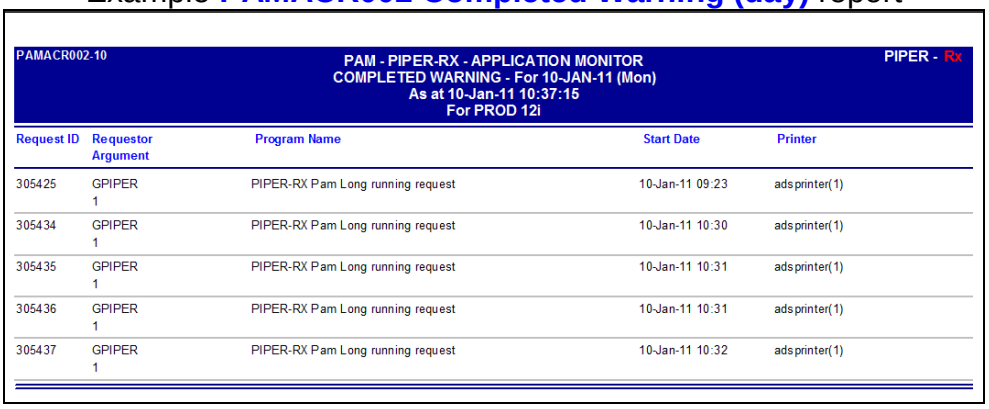
Twitter: piper-rx - *Web:* www.piper-rx.com - *Mail:* pam@piper-rx.com -

---

7.5 Alert report

PAMreports – Actions provides a Completed Warning Report as per the following example:

Example **PAMACR002 Completed Warning (day)** report



PAMACR002-10

PAM - PIPER-RX - APPLICATION MONITOR
COMPLETED WARNING - For 10-JAN-11 (Mon)
As at 10-Jan-11 10:37:15
For PROD 12i

PIPER - Rx

Request ID	Requestor Argument	Program Name	Start Date	Printer
305425	GPIPER 1	PIPER-RX Pam Long running request	10-Jan-11 09:23	adsprinter(1)
305434	GPIPER 1	PIPER-RX Pam Long running request	10-Jan-11 10:30	adsprinter(1)
305435	GPIPER 1	PIPER-RX Pam Long running request	10-Jan-11 10:31	adsprinter(1)
305436	GPIPER 1	PIPER-RX Pam Long running request	10-Jan-11 10:31	adsprinter(1)
305437	GPIPER 1	PIPER-RX Pam Long running request	10-Jan-11 10:32	adsprinter(1)

8 Example 4: CUST-004 – Alerts that notify you if a monitored item occurs (once per request)

In this example we want to alert when specific concurrent programs have been submitted but only alert once per request.

Whilst for this example we have hard coded the concurrent requests of interest, you would most likely create a table to hold the programs of interest as we have done with the *PAM* piper_rx_pam_cp_monitor_tl table check_exists attribute.

In order to prevent generating alerts on the same requests, when an alert is raised, the request id is stored in the piper_rx_pam_alerts.alert_id_value attribute for the alert. When the custom check is run the piper_rx_pam_alerts table is scanned and any requests that have already been generated are excluded.

Examples of this form of alert include alerting when:

- An invoice or PO etc… is over a designated value (storing the PO number)
- A credit note over a defined value has been created

In this example we will be using a cursor and as such it is possible to generate a large number of alert e-mails. If you do happen to generate a large number of *PAM* alerts, remember *PAM* has an e-mail alert grouping feature so as to prevent e-mail storms.

***PAM* Custom Check Form**

Alert ID	CUST-004
Alert Description	Generate an alert when specific concurrent programs have been submitted. Only report once per program submission.
Alert threshold	N/A
Alert Severity	I - Informational ✓ W – Warning C - Critical
Alert Frequency Value	10
Alert Frequency Units	MIN - Minutes ✓ HR – Hours DAY - Days

	WK - Weeks	
	MON - Mon	
Alert Hour of day	N/A (default to 5am)	
Alert message	Program PROG_NAME submitted by USER_NAME to start at START_DATE	
E-mail Route	Default	
PLSQL Name	PIPER_RX_PAM_CUSTOM_MONITOR.EXAMPLE_FOUR	
Notes		
Report Required	Yes	
Report Details	List all monitored requests for the specified day	

8.1 Alert SQL

For this custom check we need is identify if a monitored concurrent Program has been submitted.

```
SELECT fcr.request_id,
       fu.user_name,
       to_char(fcr.requested_start_date, 'DD-Mon-YY HH24:MI')
requested_start,
       substr(fcp.concurrent_program_name, 1,30)
  FROM applsys.fnd_concurrent_requests fcr,
       applsys.fnd_concurrent_programs fcp,
       applsys.fnd_user fu
 WHERE fcr.program_application_id = fcp.application_id
   and fcr.concurrent_program_id = fcp.concurrent_program_id
   and fcr.requested_by = fu.user_id
   and fcp.concurrent_program_name in ('DEACTIVATE', 'PAM_LONG_RUN');
```

The trick here is to exclude any requests that have already been alerted. Whenever we generate an alert we add the request id to the alert record in the alert_id_value attribute so we can then use this value to exclude any requests that have already been alerted.

```
   and fcr.request_id not in ( SELECT alert_id_value
                                 FROM piper_rx_pam_alerts
                                WHERE alert_id = 'CUST-004')
```

For performance, we want to limit the number of requests we access when scanning the fnd_concurrent_requests table. At the beginning of each day we set the ***PAM*** config entries working_numeric_value for the alert to the maximum request_id of the prior day. We then use this value to limit the search of the fnd_concurrent_requests table:

```
       UPDATE piper_rx_pam_config
          SET working_numeric_value =
```

```
                    ( SELECT max(fcr.request_id)
                        FROM applsys.fnd_concurrent_requests fcr
                       WHERE fcr.requested_by > -1
                         and fcr.actual_completion_date > sysdate - 1)
        WHERE alert_id = v_alert_id;
```

I have used fcr.requested_by > -1 so as to access the completion date index.

8.2 *PAM* Config Entry

Again, in this case the config entry must exist prior to the package build as we need to reference the threshold_numeric_value and working_numeric_value etc…

Where the ***PAM*** config entry is created before the package it is **very** important to set the value for alert_active_flag to 'N' so as ***PAM*** does not attempt to run the package until you have fully tested your package.

If ***PAM*** does run the package prior to the package existing or whilst the package is in an invalid state, ***PAM*** will record that the package is causing errors and as part of the ***PAM*** internal protection feature (IN-012), after a number of errors have been recorded ***PAM*** will disable the check for the remainder of the day.

Using the data from the ***PAM*** Custom Check Form you can now create your ***PAM*** config entry:

Example *PAM* config entry

```
INSERT INTO piper_rx_pam_config
       (alert_id,
        alert_category,
        alert_description,
        alert_package_name,
        alert_package_body,
        alert active flag,
        alert_frequency,
        alert_frequency_units,
        alert_hour_of_day,
        last_check_date,
        alert_severity,
        threshold_value_description,
        threshold_varchar_value,
        threshold_date_value,
        threshold_numeric_value,
        working_value_description,
        working_varchar_value,
        working_date_value,
        working_numeric_value,
        alert_action_id,
        email_route)
INSERT INTO piper_rx_pam_config
```

```
        VALUES ('CUST-004',
                'CUST',
                'Alert when selected programs are submitted',
                'PIPER_RX_PAM_CUSTOM_MONITOR',
                'EXAMPLE_FOUR',
                'N',
                10,
                'MIN',
                5,
                sysdate,
                'I',
                null,
                null,
                null,
                null,
                'Request id to limit search for perfromnace',
                null,
                null,
                0,
                null,
                'DEF');
COMMIT;
```

Once you are happy you have tested your PLSQL package you can enable you custom ***PAM*** alert by setting the value for alert_active_flag to 'Y' so your custom check will now be run by the ***PAM*** collector process.

8.3 Create your PLSQL package

For this tutorial we have provided an example PLSQL package piper_rx_pam_custom_monitor.pkb with all 4 examples outlined in this tutorial. The following extract will describe how the package is structured for this alert:

Extract from piper_rx_pam_custom_monitor.pkb

```
CREATE OR REPLACE PACKAGE PIPER_RX_PAM_CUSTOM_MONITOR IS

   PROCEDURE EXAMPLE_FOUR;  -- CUST-004

END PIPER_RX_PAM_CUSTOM_MONITOR;

/

CREATE OR REPLACE PACKAGE BODY PIPER_RX_PAM_CUSTOM_MONITOR AS

error_message varchar2(100);

PROCEDURE EXAMPLE_FOUR IS

v_alert_id           varchar2(20)  := 'CUST-004';
v_date_change        number(5);
v_request_id_limiter number(10);

v_alert_severity     varchar2(2);
v_alert_details      varchar2(200);

c_request_id         number(30);
c_user_name          varchar2(30);
```

```
c requested start     varchar2(30);
c_program_name        varchar2(30);

CURSOR EXAMPLE FOUR C IS
SELECT fcr.request_id,
       fu.user_name,
       to char(fcr.requested start date, 'DD-Mon-YY HH24:MI') requested_start,
       substr(fcp.concurrent program name, 1,30)
  FROM applsys.fnd concurrent requests fcr,
       applsys.fnd concurrent_programs fcp,
       applsys.fnd_user fu
 WHERE fcr.program_application_id = fcp.application_id
   and fcr.concurrent_program_id = fcp.concurrent_program_id
   and fcr.requested by = fu.user id
   and fcr.request id not in ( SELECT alert id value
                                 FROM piper rx pam alerts
                                WHERE alert_id = 'CUST-004')
   and fcp.concurrent_program_name in ('DEACTIVATE', 'PAM_LONG_RUN')
   and fcr.request id >  v request_id_limiter;

  BEGIN

   -- Check if date has changed if so reset the request_id to limit search
   SELECT sign(trunc(last_check_date) - trunc(sysdate))
     INTO v date change
     FROM piper rx pam config
    WHERE alert_id = v_alert_id;

    IF ( v_date_change != 0 ) THEN

       UPDATE piper rx pam config
          SET working numeric value =
                   ( SELECT max(fcr.request id)
                       FROM applsys.fnd_concurrent_requests fcr
                      WHERE fcr.requested_by > -1
                        and fcr.actual_completion_date > sysdate - 1)
        WHERE alert_id = v_alert_id;

       COMMIT;

    END IF;

     --Get alert severity
    SELECT nvl(alert severity, 'I'),
           working_numeric_value
      INTO v_alert_severity,
           v_request_id_limiter
      FROM piper rx pam config
     WHERE alert_id = v_alert_id;


    OPEN EXAMPLE_FOUR_C;

        LOOP
           FETCH EXAMPLE FOUR C into c request id,
                                     c_user_name,
                                     c_requested_start,
                                     c_program_name;

           EXIT WHEN EXAMPLE_FOUR_C%NOTFOUND;

            v_alert_details := 'Program '||c_program_name ||
            ' submitted by  '||c_user_name||' to start at '||c_requested_start;

             INSERT INTO piper rx pam_alerts
                       ( alert time,
                         alert id,
                         alert_id_value,
                         alert_severity,
                         alert_details)
                  VALUES ( sysdate,
                           v alert id,
                           c_request_id,
```

```
                          v_alert_severity,
                          v_alert_details);

            COMMIT;

        END LOOP;

     CLOSE EXAMPLE_FOUR_C;

    -- UPDATE CONFIG TO SET LAST RUN TIME AND MAX REQUEST ID
    UPDATE piper_rx_pam_config
       SET last_check_date = sysdate
     WHERE alert_id = v_alert_id;

    COMMIT;

    -- ****************************
    -- **       EXCEPTION        **
    -- ****************************
    EXCEPTION
     WHEN NO_DATA_FOUND THEN
        error_message := 'No data Found';
       dbms_output.put_line('EXAMPLE_FOUR: '||error_message);
     WHEN OTHERS THEN
        error_message := 'Other problem';
       dbms_output.put_line('EXAMPLE_FOUR: '||error_message);
       dbms_output.put_line('Oracle Error : '|| sqlerrm);

  END EXAMPLE_FOUR;

END PIPER_RX_PAM_CUSTOM_MONITOR;

/
```

8.4 Example Alert

Once ***PAM*** is running your alert and an exception is found you will receive the following ***PAM*** alert e-mail message:

Example *PAM* CUST-004 – *PAM* High point exceeded e-mail alert message

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

Company = Company Name
Site = APPS 12i
Alert Level = **Informational**
Detected = 06-Jan-11 (Thu) 18:21:05
Alert Frequency = 10 Minutes

Program PAM_LONG_RUN submitted by sysadmin to start at 06-Jan-11 18:45

Twitter: piper-rx - *Web:* www.piper-rx.com - *Mail:* pam@piper-rx.com -

8.5 Alert Report

The following is an example of the SQL you could use to generate a report for monitored requests that have been requested:

```
SELECT fcr.request id,
       fu.user_name requested_by,
       fcp.concurrent_program_name,
       to_char(fcr.requested_start_date, 'DD-Mon-YY HH24:MI') requested_start,
       to char(fcr.actual start date, 'DD-Mon-YY HH24:MI') actual start,
       to char(fcr.actual completion date, 'DD-Mon-YY HH24:MI') actual_completion,
       nvl(ltrim(to char(mod(trunc((fcr.actual completion_date -
fcr.actual_start_date)*24),24),    '00')),0) ||':'||
          nvl(ltrim(to_char( mod(trunc((fcr.actual_completion_date -
fcr.actual start date)*1440),60), '00')),0) run_time_hh_mm
  FROM applsys.fnd concurrent requests fcr,
       applsys.fnd concurrent_programs fcp,
       applsys.fnd_user fu
 WHERE fcr.program_application_id = fcp.application_id
   and fcr.concurrent_program_id = fcp.concurrent_program_id
   and fcr.requested by = fu.user id
   and fcp.concurrent program name in ('DEACTIVATE', 'PAM_LONG_RUN')
   and fcr.request id in ( SELECT alert id value
                             FROM piper_rx_pam_alerts
                            WHERE alert_id = 'CUST-004')
 ORDER by fcr.request_id DESC;
```

Example output

REQUES...	REQUESTED_BY	CONCURRENT_PROGRAM_NAME	REQUESTED_START	ACTUAL_START	ACTUAL_COMPLETION	RUN_TIME_HH_MM
305479	SYSADMIN	PAM_LONG_RUN	12-Jan-11 11:34	12-Jan-11 11:35	12-Jan-11 11:36	00:01
305477	SYSADMIN	PAM_LONG_RUN	12-Jan-11 11:27	12-Jan-11 11:27	12-Jan-11 11:28	00:01
305475	SYSADMIN	PAM_LONG_RUN	12-Jan-11 11:13	12-Jan-11 11:13	12-Jan-11 11:14	00:01

9 Other customisations

9.1 Changing the *PAM* alert severity colors

The colours used for the ***PAM*** informational, warning and critical severity levels can be changed using the following ***PAM*** API:

```
exec PIPER_RX_PAM_API.PAM_EMAIL_SEVERITY_COLOURS_SET ( 'I', '#0000FF);
```

Parameter 1: The severity level and valid values are:

I	**I**nformational - (Default #0000FF)
W	**W**arning - (Default #FF9900)
C	**C**ritical - (Default #FF0000)

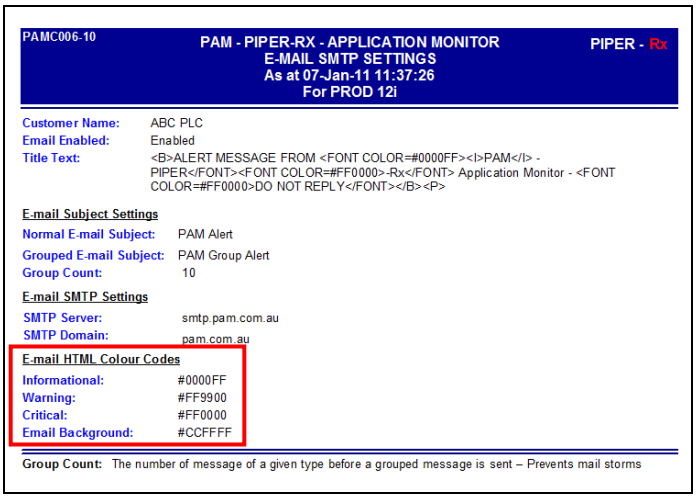
Parameter 2: The hex colour code for the desired colour.

Once the colour has been changed send a test e-mail to test the colour you have chosen using the following procedure call:

```
exec PIPER_RX_PAM_SENDMAIL.send_test_email ('DEF');
```

The current severity colours can be found using ***PAMreports*** – Config **PAMC006 PAM SMTP Settings** report:

Example **PAMC006 PAM SMTP Settings** report



PAMC006-10

PAM - PIPER-RX - APPLICATION MONITOR
E-MAIL SMTP SETTINGS
As at 07-Jan-11 11:37:26
For PROD 12i

PIPER - Rx

Customer Name: ABC PLC
Email Enabled: Enabled
Title Text: <B>ALERT MESSAGE FROM <FONT COLOR=#0000FF><I>PAM</I> - PIPER</FONT><FONT COLOR=#FF0000>-Rx</FONT> Application Monitor - <FONT COLOR=#FF0000>DO NOT REPLY</FONT></B><P>

E-mail Subject Settings
Normal E-mail Subject: PAM Alert
Grouped E-mail Subject: PAM Group Alert
Group Count: 10

E-mail SMTP Settings
SMTP Server: smtp.pam.com.au
SMTP Domain: pam.com.au

E-mail HTML Colour Codes
Informational: #0000FF
Warning: #FF9900
Critical: #FF0000
Email Background: #CCFFFF

Group Count: The number of message of a given type before a grouped message is sent – Prevents mail storms

9.2 Setting the *PAM* alert e-mail background colour

The background colour of the ***PAM*** e-mail alerts can be set using the following ***PAM*** API:

```
exec PIPER_RX_PAM_API.PAM_EMAIL_BG_COLOUR_SET ( '#CCFFFF' );
```

Parameter 1: The hex colour code

This feature could be used to differentiate between different application instances.

9.3 Resetting the email colours to the default values

The values can be reset to the default values using the following ***PAM*** API:

```
exec PIPER_RX_PAM_API.PAM_EMAIL_COLOUR_CODES_RESET;
```

Using this API will set the ***PAM*** e-mail colours back to the default values of:

- Informational - '#0000FF' - Blue
- Warning - '#FF9900' - Orange
- Critical - '#FF0000' - Red
- E-mail background colour - '#CCFFFF' - Light blue

9.4 Changing the *PAM* alert e-mail links

At the bottom of each ***PAM*** alert e-mail is a section where you can add a number of links:

Example Web links section

Twitter: piper-rx - *Web:* www.piper-rx.com - *Mail:* pam@piper-rx.com -

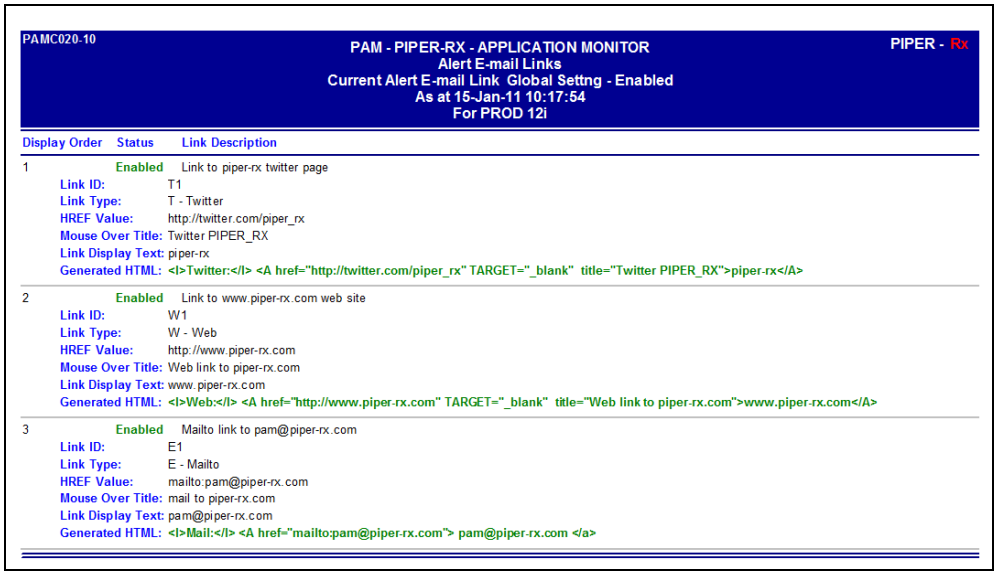
This feature can be turned on or off using the following ***PAM*** API:

```
exec PIPER_RX_PAM_API.PAM_EMAIL_LINK_DISPALY_SET ( 'N' );
```

Parameter 1: 'Y' = Enable web links section, 'N' = turn off web links section

You can view the current web links data using ***PAMreports*** – Config PAMC020 PAM Alert Email Links report:

Example **PAMC020 PAM Alert Email Links** report



PAMC020-10

PAM - PIPER-RX - APPLICATION MONITOR
Alert E-mail Links
Current Alert E-mail Link Global Settng - Enabled
As at 15-Jan-11 10:17:54
For PROD 12i

PIPER - Rx

Display Order	Status	Link Description
1	Enabled	Link to piper-rx twitter page
	Link ID:	T1
	Link Type:	T - Twitter
	HREF Value:	http://twitter.com/piper_rx
	Mouse Over Title:	Twitter PIPER_RX
	Link Display Text:	piper-rx
	Generated HTML:	<I>Twitter:</I> <A href="http://twitter.com/piper_rx" TARGET="_blank" title="Twitter PIPER_RX">piper-rx</A>
2	Enabled	Link to www.piper-rx.com web site
	Link ID:	W1
	Link Type:	W - Web
	HREF Value:	http://www.piper-rx.com
	Mouse Over Title:	Web link to piper-rx.com
	Link Display Text:	www.piper-rx.com
	Generated HTML:	<I>Web:</I> <A href="http://www.piper-rx.com" TARGET="_blank" title="Web link to piper-rx.com">www.piper-rx.com</A>
3	Enabled	Mailto link to pam@piper-rx.com
	Link ID:	E1
	Link Type:	E - Mailto
	HREF Value:	mailto:pam@piper-rx.com
	Mouse Over Title:	mail to piper-rx.com
	Link Display Text:	pam@piper-rx.com
	Generated HTML:	<I>Mail:</I> <A href="mailto:pam@piper-rx.com"> pam@piper-rx.com

9.4.1 Adding a web link

A web link entry can be added using the following ***PAM*** API:

```
DECLARE

BEGIN

PIPER_RX_PAM_API.PAM_EMAIL_LINK_ADD( 'W2',
                                     'W',
                                     'Example entry',
                                     'Y',
                                     5,
                                     'http://www.piper-rx.com',
                                     'Mouse over text here',
                                     'Test entry' );

COMMIT;

END;
```

Parameter 1:

A unique web link ID. The standard used by ***PAM*** is that all web link id's start with the letter W followed by a sequence number, e-mail links start with the letter E followed by a sequence number…

Parameter 2:

Link Type: Valid entries are **T** (Twitter) **W** (Web) **E** (E-mail)

Parameter 3:

A free format description of the link - varchar2(100)

Parameter 4:

The display status defines if the link is to be displayed in the ***PAM*** e-mail alerts 'Y' = Display the link, 'N' = Do not display the link

Parameter 5:

A numeric value that determines the order the links will be displayed (left to right)

Parameter 6:

The link title that will appear in the ***PAM*** e-mail alert i.e. Web link to piper-rx.com i.e. http://www.piper-rx.com

Parameter 7:

Mouse over text

Parameter 8:

The displayed value that will appear in the link section

9.4.2 Deleting an alert e-mail link

An alert email link can be deleted using the following ***PAM*** API:

```
exec PIPER_RX_PAM_API.PAM_EMAIL_LINK_DEL    ( 'W2' );
```

Parameter 1: Is the e-mail link ID

9.4.3 Disabling one or more alert e-mail links

A ***PAM*** alert e-mail link can be enabled / disabled using the following ***PAM*** API:

```
exec PIPER_RX_PAM_API.PAM_EMAIL_LINK_ENABLE_LINK ('W2', 'N' );
```

Parameter 1: Is the e-mail link ID

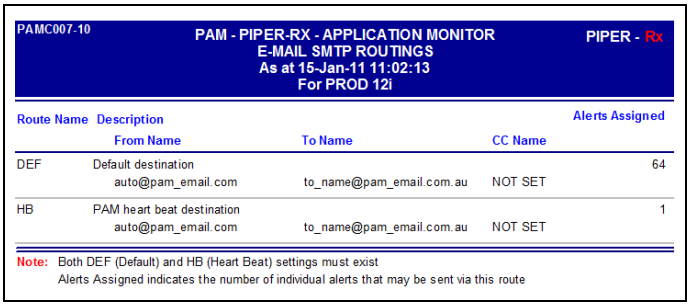
Parameter 2: Y = Enable the check
N = Disable check

9.5 *PAM* alert e-mail routings

The ***PAM*** e-mail routings define where ***PAM*** e-mail alerts are to be sent.

The current ***PAM*** e-mail routings can be found using ***PAMreports*** – Config PAMC007 PAM SMTP Email **Routing** report:

Example **PAMC007 PAM SMTP Email Routing** report



PAMC007-10 — PAM - PIPER-RX - APPLICATION MONITOR — PIPER - Rx

E-MAIL SMTP ROUTINGS
As at 15-Jan-11 11:02:13
For PROD 12i

Route Name	Description / From Name	To Name	CC Name	Alerts Assigned
DEF	Default destination auto@pam_email.com	to_name@pam_email.com.au	NOT SET	64
HB	PAM heart beat destination auto@pam_email.com	to_name@pam_email.com.au	NOT SET	1

Note: Both DEF (Default) and HB (Heart Beat) settings must exist
Alerts Assigned indicates the number of individual alerts that may be sent via this route

Note: Both the 'DEF' and 'HB' routings **must** exist and be valid addresses

The "Assigned Alerts" value is the number of ***PAM*** alerts that have been assigned to the ***PAM*** e-mail route.

9.5.1 Adding a new *PAM* alert e-mail route

Assuming you want all ***PAM*** workflow alerts to go to the workflow administrator you will need to add a new ***PAM*** e-mail route.

A new e-mail route can be added using the following ***PAM*** API:

```
DECLARE

BEGIN

   PIPER_RX_PAM_API.PAM_EMAIL_ROUTING_ADD
               ( 'WF',
                 'Workflow Alert to be sent to Workflow admin',
                 'pam@abc.com.au',
                 'wf_admin@abc.com.au' ) ;
COMMIT;

END;
```

Parameter 1:

The unique e-mail rout identifier, this value can be any value you wish as long as it is unique and descriptive.

Parameter 2:

Free format description of the route varchar2(100)

Parameter 3:

The e-mail address to appear in the “From” section of the e-mail, in most cases this must be a valid e-mail address.

Parameter 4:

The e-mail address of the recipient.

9.5.2 Deleting a *PAM* alert e-mail route

The first thing to do is to make sure all config entries e-mail routes have been changed to a valid e-mail route.

If you do not change the routes in the ***PAM*** config entries, an entry that is still set to the route to be deleted will be reset to ‘DEF’

A ***PAM*** e-mail route can be deleted using the following ***PAM*** API:

```
exec PIPER_RX_PAM_API.PAM_EMAIL_ROUTING_DELETE ( 'WF' );
```

Parameter 1: The e-mail route to be deleted

10 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.