Configuring Alerts settings

This section contains the following topics:

About role management in Alerts

The roles management feature in Alerts lets you assign roles to users and control whether they will be able to do the following operations:

- View and/or configure Alerts general settings
- View and/or configure Alerts metric settings for specific instances
- View and/or configure Alerts metrics default settings for all instances
- View and/or configure Alerts Precise status metrics settings
- Add and remove Alerts customized metrics

Permission is a combination of permission types (such as:ADMINISTRATE, MONITOR) and operation (such as: VIEW, EXECUTE).

In order to enable the Alerts settings screen, the user needs to have a role with ADMINISTRATE.VIEW permissions. For metric settings, VIEW or EXECUTE permissions should be granted for: Environments, AppTiers, Instances, or Technologies.

To add and remove customized metrics, the user should have specific technology EXECUTE permissions. For Precise status metrics settings control, the user should have Precise technology EXECUTE permissions. Configure permissions and roles in AdminPoint. See Defining roles and users in Precise.

About configuring Alerts general settings

The Alerts, issued by Alerts, are based on information collected by Insight agents, agents of Precise products, or Report Manager agents. See Configuring Alerts general settings.

About configuring Alerts metric settings

The Alerts, issued by Alerts, are based on information collected by Insight agents, agents of Precise products, or Report Manager agents. For most of the metrics, Alerts enables you to launch the relevant Precise product without returning to the StartPoint screen. See Configuring Alerts metric settings.

The Alerts metric settings dialog box includes the following tabs:

- Settings
- Activities
- Copy Metrics Settings

About editing metric properties

You can edit the properties of each metric that is available in your Precise environment, including Cross-AppTiers metrics, such as: FocalPoints, Agents, Processes, and Licenses. See Editing metric properties.

The Metric Properties dialog box includes the following tabs:

- Scheduling
- Thresholds
- Actions

See About metric properties for Action settings.

About metric properties for Action settings

An environment that is monitored by Alerts may generate alerts at any time. Sitting in front of the screen waiting for a metric to go critical may be strenuous and time consuming. Instead, you can set Alerts to inform you about any alert, or to run your repair utility to fix certain problems.

Alerts provides the following action types when an alert is raised:

- **Email**. Alerts sends a user-defined message to a user-defined recipient. You can set an email action per instance level or per metric.
- **Message Box**. Alerts displays a user-defined message in a pop-up box on each running Alerts user interface.
- **Program**. Alerts runs a user-defined program that is stored in the directory: `<i3_root>\products\pulse\userprograms`.
- **SNMP**. Informs your management framework of any changes in the metric status.

- **MOM**. Alerts sends an alert to the MOM server console.

In the Actions tab you define the rules that stipulate when an alert is triggered.

You can test each action that you define in the Action Types by clicking the Test button (for the Email, Message, and Program action types you must first enter parameters to enable the Test button). This test triggers the action including its dynamic parameters. The dynamic parameters retrieve the values of the last sample (if a metric was not sampled before the test, some of the parameters may hold invalid data). See Using dynamic parameters in actions.

On the lower half of the Actions tab window a metric summary table is displayed indicating:

- Total number of declared email actions
- Total number of declared Message actions
- Total number of declared Program actions and whether or not an SNMP or MOM action is applied to the metric.

The two tables below indicate how rules are applied when issuing alerts (both Near-Critical and Critical) for different alert transitions (applies to the following actions: email, message, and program):

Table 16-1 Near-Critical

Old / New	Critical	Near-Critical	Normal	Not sampled
Critical	Do not issue	Issue	Do not issue	Do not issue
Near-Critical	Issue	Do not issue	Do not issue	Do not issue
Normal	Issue	Issue	Do not issue	Do not issue
Not Sampled	Issue	Issue	Do not issue	Do not issue

Table 16-2 Critical

Old / New	Critical	Near-Critical	Normal	Not sampled
Critical	Do not issue	Do not issue	Do not issue	Do not issue
Near-Critical	Issue	Do not issue	Do not issue	Do not issue
Normal	Issue	Do not issue	Do not issue	Do not issue
Not Sampled	Issue	Do not issue	Do not issue	Do not issue

The table below indicates how rules are applied when issuing alerts for different alert transitions (applies to the following actions: SNMP or MOM):

Table 16-3 SNMP or MOM

Old / New	Critical	Near-Critical	Normal	Not sampled
Critical	Do not issue	Issue	Issue	Issue
Near-Critical	Issue	Do not issue	Issue	Issue
Normal	Issue	Issue	Do not issue	Issue
Not Sampled	Issue	Issue	Do not issue	Do not issue

Email action properties

Use the Email Action Type to set the email address of the recipient and the message text that will be sent when an alert is raised by the specified metric.

The email is by default in HTML format. The user can change the default HTML format by creating a .css file and placing it under the following path:

```
\products\pulse\pulsefocal\etc\html_email_action.css
```

Alerts sets the subject of the email automatically as:

```
The $METRIC_NAME metric exceeded its $METRIC_ALERT threshold. The metric's value was: $METRIC_VALUE. The
metric's thresholds are:$THRESHOLDS.
```

The message text can also include dynamic parameters. See Using dynamic parameters in actions.

To set Alerts to send an email when an alert is raised

1. From the Actions tab on the Metric Properties dialog, select the email option on the Action Type list box.
2. Click **Add**.
3. From the When list box, select the minimum alert severity level that will cause this action to run.
4. If you want to alert higher management only, after several sequential alert triggers, enter a value into the more than <...> time text box.
5. Set a valid email address. You may skip this step if you have entered a default email address for the metric instance in the **Settings>Email** dialog box. See Editing instance settings on the Instances tab.
6. Make sure that the email definitions of Alerts are configured correctly so that the email will reach its destination.
7. Type the content of the text to be sent. The text content can contain any character including action dynamic parameters.

8. To test your definitions, click **Test**.
9. To save your definitions, choose whether to save them either for the selected instance, or for all the environments' instances. Then click **Save** and **Close**.

To edit Alerts email action properties

1. From the Actions tab on the Metric Properties dialog, select the email option on the Action Type list box.
2. Select the email action you want to edit.
3. Click **Edit**.
4. In the Edit email dialog, make the necessary changes.
5. To test your new definitions, click **Test**.
6. To save your new definitions, choose whether to save them either for the selected instance, or for all the environments' instances. Then click **Save** and **Close**.

To delete Alerts email action properties

1. From the Actions tab on the Metric Properties dialog, select the email option on the Action Type list box.
2. Select the email action you want to edit.
3. Click **Delete**.

Message action properties

Use the Message Action Type to set the text message that will be displayed on every screen with an open Alerts user interface, when an alert is raised by the specified metric. The message text can also include dynamic parameters.

> There may be a delay of up to 15 minutes before the raised alarm is displayed.

See Using dynamic parameters in actions.

To set Alerts to display a message when an alert is raised

1. From the Actions tab on the Metric Properties dialog, select the message option on the Action Type list box.
2. Click **Add**.
3. From the When list box, select the minimum alert severity level that will cause this action to run.
4. If you only want to alert higher management, after several sequential alert triggers, enter a value into the more than <...> times text box.
5. In the Message text box, type the message you want to display on the user interface.
6. To test your definitions, click **Test**.
7. To save your definitions, click **OK**.

To edit Alerts message action properties

1. From the Actions tab on the Metric Properties dialog, select the message option on the Action Type list box.
2. Select the message action you want to edit.
3. Click **Edit**.
4. In the Edit message dialog, make the necessary changes.
5. To test your new definitions, click **Test**.
6. To save your new definitions, click **OK**.

To delete Alerts message action properties

1. From the Actions tab on the Metric Properties dialog, select the message option on the Action Type list box.
2. Select the message action you want to edit.
3. Click **Delete**.

Program action properties

Use the Program Action Type to set your program to be ran as an action when an alert is raised by the specified metric.

To set Alerts to run your program as an action when an alert is raised

1. From the Actions tab, select the Program option on the Action Type list box.
2. Click **Add**.
3. From the When list box, select the minimum alert severity level that will cause this action to run.
4. If you want to alert higher management only, after several sequential alert triggers, enter a value into the more than <...> times text box.
5. In the Program text box, specify your program name without a path. Consider the following:
    - Your program must be located in the directory:
      `<i3_root>\products\pulse\userprograms`
      on the server where you want to activate the program action, if you choose to run it in the Alerts agent server.

      > Verify that the script is deployed by you in any required server where the Alerts agent is located and should perform an action.

    - You can use dynamic parameters at the command line. See Using dynamic parameters in actions.
    - Verify that Windows scripts start with the line `@echo off` to avoid that commands be printed.

- Verify that UNIX scripts start with the shell type, for example:
  `#!/bin/ksh`

6. From the On list box, select whether to run your program on the Alerts FocalPoint server, or on the server where your instance is running.
7. To run programs on the instance side, copy the `pulseprogram` executable from the directory: Utilities\alerts\AlertsProgs\Win\Pulseprogram.exe on the Precise DVD to the directory:
   `<i3_root>\products\pulse\bin` on the instance server.
   If you call another program, or write into a log file, the active directory is the Precise root directory (not the directory: `<i3_root>\products\pulse\userprograms`). Do not set a path for running a program in another directory.
8. To test your definitions, click **Test**.
9. To save your definitions, click **OK**.

To edit Alerts program action properties

1. From the Actions tab on the Metric Properties dialog, select the program option on the Action Type list box.
2. Select the program action you want to edit.
3. Click **Edit**.
4. In the Edit program dialog, make the necessary changes.
5. To test your new definitions, click **Test**.
6. To save your new definitions, click **OK**.

To delete Alerts program action properties

1. From the Actions tab on the Metric Properties dialog, select the program option on the Action Type list box.
2. Select the program action you want to edit.
3. Click **Delete**.

SNMP action properties

Use the SNMP Action Type to set alerts to be reported to an SNMP based management tool that you may have, such as CA Unicenter® and HP OpenView.

> To enable the SNMP functionality, verify that the Alerts SNMP definitions are configured properly.

See Setting an SNMP server for actions on the SNMP tab.

To set the SNMP functionality for the specified metric

1. From the Actions tab, select the SNMP option on the Action Type list box.
2. To apply SNMP traps, check **Send an SNMP trap whenever the metric alert level changes**.
3. To test your definitions, click **Test**.
4. To save your definitions, choose whether to save them either for the selected instance, or for all the environments' instances. Then click **Save** and **Close**.

MOM action properties

Use the MOM Action Type to set alerts to be reported to the MOM server whenever the metric alert level changes.

> To enable the MOM functionality, verify that the Alerts MOM definitions are configured properly and that integration with the MOM server succeeded.

See Setting a MOM server for actions on the MOM tab.

To set the MOM functionality for the specified metric

1. From the Actions tab, select the MOM option on the Action Type list box.
2. To send alerts to the MOM server, check **Send alert to MOM server whenever the metric alert level changes**.
3. To test your definitions, click **Test**.
4. To save your definitions, at the bottom of the screen, choose to save either the selected instance, or all the environment's instances. Then click **Save** and **Close**.

Using dynamic parameters in actions

When you set rules in actions, you can use the dynamic parameters listed in the following table. Set the dynamic parameters in the Message text area of the email tab, Message tab, or in the Program text box of the Program tab.

> The dynamic parameters in actions are not the same as used in customized metrics.

See Using dynamic parameters in customized metrics. The table below describes the dynamic parameters that you can set.

Table 16-4 Dynamic parameters

Dynamic parameter	Definition
$METRIC_NAME	Name of the metric.
$METRIC_SET	Name of the metric set.
$TECHNOLOGY_NAME	Name of the instance technology.
$INSTANCE_NAME	Name of the instance sampled by the metric.
$APPTIER_NAME	Name of one of the AppTiers to which the instance applies.
$APPTIERS_NAME	Correlated to $ENVIRONMENT_NAME.
$ENVIRONMENT_NAME	Name of one of the environments to which the instance applies.
$ENVIRONMENTS_NAME	Name of one of the environments to which the instance applies.
$METRIC_ALERT	The alert severity level of the metric that was issued when the action was activated.
$SAMPLE_REASON	Reason of the sample that caused the action. The reasons can be one of the following: • **Schedule**. A regular sample, which is initiated according to the sampling schedule. • **Resample**. A sample initiated by clicking the **Resample** button. • **Restart**. A sample of pre-configured metrics after InformPoint restarts. • **Resample As Startup**. A sample of pre-configured metrics after Alerts FocalPoint restarts. • **Resample By Demand**. A sample due to a change in the instance's availability (for example, if the monitored Oracle database was shutdown, the Oracle Availability metric will be resampled by demand).
$ITEMS	Has been deprecated (will not be used in future versions). Use $METRIC_VALUE instead.
$METRIC_VALUE	• For Single value metrics: Value returned by the metric that was issued when the action was activated. • For list metrics: List of items returned by the metric sampling. The format of the returned string is as follows: Tab-delimited in Email and Message actions. Underscore-delimited in Program actions.
$METRIC_TIME	The time of the last actual sampling.
$SAMPLE_RANGE_START_TIME	The sampling period start time of the last sample.
$SAMPLE_RANGE_END_TIME	The sampling period end time of the last sample.
$METRIC_PROGRESS	Progress status of the metric.
$PROGRESS_UPDATING_USER	Role name of last user that modified the progress status of the metric.
$PROGRESS_UPDATE_TIME	Time of the last update of the progress status of the metric.
$THRESHOLDS	Warning (near critical) and Critical threshold values defined for the metric. Relevant only for a metric with sub-metrics.
$NEAR_CRITICAL_THRESHOLD	Warning (near critical) threshold value defined for the metric. Relevant only for a metric with no sub-metrics.
$CRITICAL_THRESHOLD	Critical threshold value defined for the metric. Relevant only for a metric with no sub-metrics.
$SERVER_MACHINE_NAME	Name of the server machine on which the instance is running.
$SAMPLING_RATE	Sampling rate of the metric.
$SAMPLING_PERIOD	Sampling period of the metric.
$MIN_VALUE	Minimum value that is acceptable for the metric.

An example for using dynamic parameters in Email or Message actions can be found in the Message text box in the Email tab (default message).

Examples for using dynamic parameters in Program actions can be found in the directory:

`<i3_root>\products\pulse\userprograms`.

The following examples are provided:

- `action_example.bat` (for Windows). To activate this file, set the following command line:
  - action_example.bat $METRIC_ALERT $METRIC_TIME $METRIC_NAME $METRIC_VALUE
- `action_example.sh` (for Linux or UNIX). To activate this file, set the following command line:
  - action_example.sh $METRIC_ALERT $METRIC_TIME $METRIC_NAME $METRIC_VALUE

For both examples, when Alerts activates the file, it writes to a log file information about the alert, which can be helpful for troubleshooting alerts. The log file name is `action_example.log` and it contains the following text:

==========

ALERT: <$METRIC_ALERT result>

TIME: <$METRIC_TIME result>

Metric <$METRIC_NAME result> showed value

<$METRIC_VALUE result>.

See Email action properties, Message action properties, Program action properties, SNMP action properties, and MOM action properties.

About setting Alerts SNMP connectivity

SNMP (Simple Network Management Protocol) is the Internet standard protocol for network management software. Alerts supports connectivity to an SNMP Framework using the following Protocol Data Unit (PDU) operations:

- **SNMP get Operation**. Alerts FocalPoint can receive SNMP Get requests from an SNMP manager to read the Precise Alerts database accordingly.
- **SNMP trap operation**. When Alerts SNMP server is enabled for actions, Alerts sends SNMP traps to your SNMP manager in case an alert state is raised or resolved.

SNMP Get Operation

Alerts has an extended user interface, allowing you to set metric definitions and to view the samples results. However, you may want to use your SNMP manager to receive metric specific data from Alerts.

To enable this option, first create a Management Information Bases (MIB) file. The MIB file maps the Alerts database entities, such as the various environments, various instances, and the metrics relevant for each of the instances. Because different sites have different environments, instances, and so on, you must adjust the Alerts MIB file to each site.

Creating an MIB file

Alerts FocalPoint creates the MIB file and displays its status (success or failure) on the standard output. The newly created MIB file name is InformForAlertsMib.mib and is stored in the following directory:

`<i3_root>\products\pulse\userprograms`

Information about the MIB creation process can be found in the log file:

`<i3_root>\logs\alerts.mibbuilder.log`

To create an MIB file

- On the server that Alerts FocalPoint is installed, run the following command, according to the server operating system, from the Precise root directory:

  On a Windows NT server. `<i3_root>\products\pulse\pulsefocal\bin\ createalertsmib.bat`

  On a Linux or UNIX server. `<i3_root>/products/pulse/pulsefocal/bin/ createalertsmib.sh`

Enabling the Get operation

To use your MIB browser to open the MIB file that you have created and browse the different metrics, you must set the Get parameters on both your SNMP manager and Alerts. The examples provided along with the following instructions relate to the CA Unicenter MIB browser, which is an SNMP management application.

To set the get parameters

1. Copy the newly created MIB file (InformForAlertsMib.mib) to the SNMP server's MIBs directory.
   For example in CA-Unicenter: `<TND_root>\SERVICES\CONFIG\MIBS`.
2. Upload the MIB file to your SNMP MIB browser.
   For example in CA-Unicenter, run the following command from the TND directory
   <TND_root>\SERVICES\CONFIG\MIBS:
   ldmib -n oidprecise -m InformForAlertsMib.mib
3. In the SNMP tab of Alerts Settings dialog box, set an available listening port in the SNMP port box. This port will receive the Get requests from your SNMP MIB browser. Also, set the SNMP version according to your MIB browser.

Browsing the Alerts MIB

Before you start browsing the Alerts MIB, it is recommended to be familiar with the following issues:

- Technology representation
- Identifying environments and instances in the MIB
- Identifying metrics in the MIB
- Identifying property fields in the MIB
- MIB structure

Technology representation

The MIB tree shows the technologies of the Precise environment as the numbers 1 - 15. The table below shows how each number in a MIB tree is mapped to a technology.

Table 16-5 MIB tree mapping

MIB number	Mapped to ...
1	Oracle
2	Sybase
3	MS-SQL
4	Tuxedo
5	Web
6	J2EE
7	SAP
8	Oracle Applications
9	Microsoft .NET
10	RESERVED
11	EMC Storage
12	Other
13	OS
14	Precise status
15	WebSphere MQ
16	Sybase Replication Server
17	DB2

Identifying environments and instances in the MIB

Alerts displays environments and instances by their names (`environment_name, instance_name`), while the MIB presents them by their identifiers (`environment_id, instance_id`).

To identify environments and instances in the MIB

1. Retrieve a mapping table that maps the environment and instance names to their identifiers by running the following SQL statement in the Alerts schema:

```
select
    INCE_ID INSTANCE_ID, INCE_NAME
    INSTANCE_NAME, INEN_ID
    ENVIRONMENT_ID, INEN_NAME
    ENVIRONMENT_NAME
from
```

```
PS_INCE_INSTANCE_INSTANCE, PS_INII_INSTANCE_APPTIER
INSTANCE_APPTIER, PS_INAP_APP_TIER APPTIER,
PS_INEN_ENVIRONMENT ENVIRONMENT
where
    INCE_ID = INII_INCE_ID AND
    INAP_ID = INII_INAP_ID AND
    INAP_INEN_ID = INEN_ID AND
    INCE_DELETED = 'F' AND
    INII_DELETED = 'F' AND
    INAP_DELETED = 'F' AND
    INEN_DELETED = 'F' AND
```

Identifying metrics in the MIB

Alerts displays metrics by their names (`metric_name`), while the MIB presents them by their identifiers (`metric_id`).

The mapping table is sorted by the metric name.

To identify metrics in the MIB

- Retrieve a mapping table that maps between the metric names and their identifiers by running the following SQL statement in the Alerts schema: select indicator_id,indicator_name, from pulse_indicators order by indicator_name;

Available property fields in the MIB

The table below describes the available property fields for a metric.

Table 16-6 Available property fields in the MIB

No.	Field	Description	Applies to
0	Value	Specifies the metric's value. Same as the dynamic parameter: `$METRIC_VALUE` Displayed in the Value metrics tab only. In case of a parent metric, only the sub-metrics show this field.	Get
1	Status	Specifies the metric's status. Possible values: Critical, Near-critical, Normal, and Unsampled. In the MIB browser, Downtime and Disabled statuses appear also as Unsampled status.	Get
2	SampleRate	Specifies the sampling rate in minutes. Same as the dynamic parameter: `$SAMPLING_RATE` In case of a parent metric, only the parent shows this field.	Get
3	NearCrThr	Specifies the metric's Near-critical threshold. Same as the dynamic parameter: `$NEAR_CRITICAL_THRESHOLD` In case of a parent metric, only the sub-metrics show this field.	Get
4	CriticalThr	Specifies the metric's Critical threshold. Same as the dynamic parameter: `$CRITICAL_THRESHOLD` In case of a parent metric, only the sub-metrics show this field.	Get
5	Enabled	Specifies whether or not the metric is enabled (values: yes or no).	Get
6	SampleTime	Specifies the metric's last sampling time. Same as the dynamic parameter: `$ACTUAL_SAMPLING_TIME`	Get
7	MetricName	Specifies the metric's name. Same as the dynamic parameter: `$METRIC_NAME`	Get
8	InstanceName	Specifies the instance's name. Same as the dynamic parameter: `$INSTANCE_NAME`	Get
9	ItemsTable	Specifies the list of items returned by the metric sampling. Same as the dynamic parameter: `$ITEMS` Displayed in the list of items of the metrics list. For parent metrics, the list contains also the sub-metrics.	Get
10	Technology	Specifies the Technology name. Same as the dynamic parameter: `$TECHNOLOGY_NAME`	Get

11	Machine	Specifies the Machine name. Same as the dynamic parameter: `$SERVER_MACHINE_NAME`	Get
12	AppTier	Specifies the AppTier name. Same as the dynamic parameter: `$APPTIER_NAME`	Get
13	Environment	Specifies the Environment name. Same as the dynamic parameter: `$ENVIRONMENT_NAME`	Get
14	MetricID	Specifies the Metric ID name.	Get
15	Returned value	Specifies the Returned value name. Same as the dynamic parameter: `$METRIC_VALUE`	Get
16	Thresholds	Specifies the Thresholds name. Same as the dynamic parameter :`$THRESHOLDS`	Get

MIB structure

The Alerts MIB structure complies with the Alerts SNMP Object Identifier (OID) structure. The OID of the Alerts Get requests for a specified metric is:

1.3.6.1.4.1.2608.1000.8.envId.techId.instId.metricId.field

Where `field` is the field number as specified in the metric fields table (previous table). You can identify this OID from the SNMP trap messages, using the `$METRIC_TOKEN` dynamic parameter.

SNMP trap operation

Using SNMP trap operations, you can automatically receive alerts in your SNMP server. The trap message contains critical information about the trap alert.

After receiving the SNMP trap message, you can use your SNMP manager to resolve the problem, notify the management level.

Alerts supports both SNMP message versions, SNMPv1 and SNMPv2. Alerts sends traps when a change occurs in a metric severity level, that is, when the metric state is changed between the following states: Critical, Near-Critical, Normal, and Unsampled.

Alerts trap message

The following is an example of an Alerts'SNMP trap message:

Trap(v1) received from host test.precise.com(10.42.136.103) at Nov 18, 2008 12:27 PM. Enterprise Oid:

.1.3.6.1.4.1.2608.1000.8.1079.1.1080.1199 (.iso.org.dod.internet.private.enterprises.2608.1000.8.1079.1.1080.1199) , Specific Type : 1, Trap Varbinds :

Object ID: .1.3.6.1.4.1.2608.1000.8.1079.1.1080.1199.7
(.iso.org.dod.internet.private.enterprises.2608.1000.8.1079.1.1080.1199.7) STRING: Availability

Object ID: .1.3.6.1.4.1.2608.1000.8.1079.1.1080.1199.14
(.iso.org.dod.internet.private.enterprises.2608.1000.8.1079.1.1080.1199.14) STRING: 1199

Object ID: .1.3.6.1.4.1.2608.1000.8.1079.1.1080.1199.8
(.iso.org.dod.internet.private.enterprises.2608.1000.8.1079.1.1080.1199.8)

STRING: ORCL

Object ID: .1.3.6.1.4.1.2608.1000.8.1079.1.1080.1199.10
(.iso.org.dod.internet.private.enterprises.2608.1000.8.1079.1.1080.1199.10) STRING: Oracle Object ID:
.1.3.6.1.4.1.2608.1000.8.1079.1.1080.1199.11

(.iso.org.dod.internet.private.enterprises.2608.1000.8.1079.1.1080.1199.11) STRING: server-name1

Object ID: .1.3.6.1.4.1.2608.1000.8.1079.1.1080.1199.12
(.iso.org.dod.internet.private.enterprises.2608.1000.8.1079.1.1080.1199.12) STRING: Oracle Object ID:
.1.3.6.1.4.1.2608.1000.8.1079.1.1080.1199.13 (.iso.org.dod.internet.private.enterprises.2608.1000.8.1079.1.1080.1199.13)
STRING: Default

Object ID: .1.3.6.1.4.1.2608.1000.8.1079.1.1080.1199.6
(.iso.org.dod.internet.private.enterprises.2608.1000.8.1079.1.1080.1199.6) STRING: test Precise trap

Object ID: .1.3.6.1.4.1.2608.1000.8.1079.1.1080.1199.1
(.iso.org.dod.internet.private.enterprises.2608.1000.8.1079.1.1080.1199.1)

STRING: test Precise trap

Object ID: .1.3.6.1.4.1.2608.1000.8.1079.1.1080.1199.15
(.iso.org.dod.internet.private.enterprises.2608.1000.8.1079.1.1080.1199.15) STRING: test Precise trap Object ID:
.1.3.6.1.4.1.2608.1000.8.1079.1.1080.1199.16 (.iso.org.dod.internet.private.enterprises.2608.1000.8.1079.1.1080.1199.16)
STRING: test Precise trap

Object ID: .1.3.6.1.4.1.2608.1000.8.1079.1.1080.1199.0
(.iso.org.dod.internet.private.enterprises.2608.1000.8.1079.1.1080.1199.0)

STRING: Test trap-message from Alerts

From the above table you can see that the Varbind Object ID for all the metrics of an Alerts trap starts with: 1.3.6.1.4.1.2608

The table below describes the SNMP trap structure. The abbreviated OID (Object ID) is the number after the last period on the varbinds table (as stated before, the rest of the number is the same for all varbinds).

Table 16-7 SNMP trap structure

Description	Abbreviated object identifier (OID)
Metric	7
Instance	8
Technology	10
Machine	11
AppTier	12
Environment	13
Sampled on	6
Alert	1
Value	15
Thresholds	16
Message until v. 8.0	0

The trap can be parsed with a commercial trap catcher according to the position of the varbinds or their OID. The following apply only to the last varbind on the table (OID=0):

- All the fields with the dollar sign ($) are dynamic parameters. Alerts translates these parameters before sending the SNMP trap.
- Spaces inside items are converted to the underscore character (_) to allow saving the position.

To enable identification of the alerted metric's OID, required for the Get and Set requests, the SNMP action supports the following dynamic parameter: `$METRIC_TOKEN`. This dynamic parameter is the MIB OID of the metric. In addition, consider the following:

- All items in the SNMP trap message keep their position, so you can access particular message items using built-in SNMP functions.
- You can set all metrics of an instance to trigger SNMP actions through the Instances tab of the Settings dialog box.
- In case of parent metrics, Alerts sends an SNMP trap message only to the parent metric.

To cause the SNMP trap to act as in version 7.5, add the following parameter to the Alerts FocalPoint registry, and then restart the Alerts FocalPoint:

```
....\products\i3fp\registry\products\alerts\pulsefocal.xml

Pulsefocal\snmp\trap

<oldTrapStyle>YES</oldTrapStyle>
```

About Alerts MOM connectivity

This section describes how to go about setting up Alerts for MOM connectivity.

Activating MOM integration in Alerts

To activate and deactivate the Alerts integration with the MOM server, go to Setting a MOM server for actions on the MOM tab.

Setting anonymous access to a MOM server

Before integrating Alerts with the MOM server in AdminPoint, anonymous access to the MOM Web console must be set.

To set anonymous access to the MOM server

1. Open the IIS Manager on the MOM server.
2. Select to open the “Microsoft Operations Manager Web console” properties screen.
3. Click the Directory Security tab.
4. On the “Anonymous access and authentication control” panel, click **Edit**.
5. Mark “Enable Anonymous access” and type in the username and password (to be used for MOM integration setup on the AdminPoint Alerts General Settings screen).
6. Save the changes.

Issuing rules for MOM actions

If MOM action is defined for a metric, MOM integration is activated and a MOM action is issued every time a metric's alert changes.

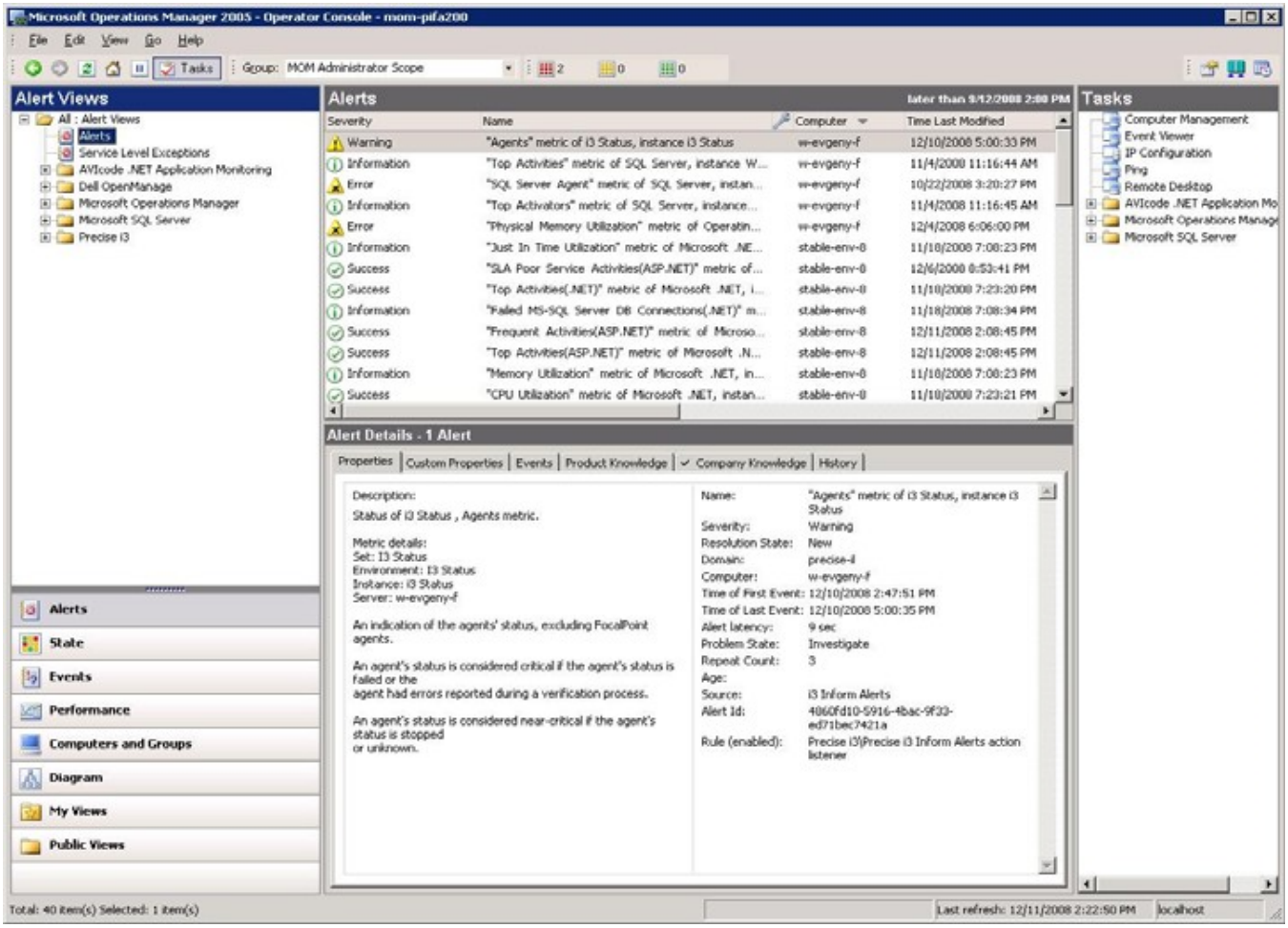
> For disabled metrics, the metric's initial status is not reported to MOM. The metric is only reported when the disabled status changes to another status. Disabled metrics do not appear in the MOM console.

Display of MOM alerts in the MOM server

The MOM alert can be displayed only when integration between Alerts and the MOM server is activated and the registration process is completed successfully. A MOM alert can be the result of either clicking the Test button or of a real action performed by the Alerts FocalPoint.

When you open the MOM Operator Console, select Alert Views in the Alert Views pane. The figure below displays the MOM Operator console user interface

Figure 16-1 MOM Operator console



If you have successfully installed the Precise Management Pack (see *Precise Installation Guide*), you will see a Precise folder in the Alert Views tree. This folder displays on the Alerts View Results pane, all actions generated as MOM alerts.

The Precise folder is divided into sub-folders, one for each of the supported technologies. To see only the alerts of a single technology, select that technology's folder.

Each alert in the Alerts View Results pane represents a metric that monitors a single instance integrated with MOM. When a metric triggers a MOM action for the first time, a line will be added to the pane. If a Customized metric that is integrated with MOM is deleted, the entry's Resolution State will change to Resolved, and the entry will be removed according to your Database Grooming properties.

On the Alert Details pane you can see details for the selected MOM alert. The details only include basic metric information. That is, the current value of the metric will not be displayed. To investigate the metric's value further, you will need to open the Precise user interface.

The table below shows the different MOM severities displayed in the MOM operator console according to the Alerts alert levels.

Table 16-8 Alert level names in the MOM operator console

Alerts alert	MOM severity
Normal	Success
Near_Critical	Warning
Critical	Error
Critical (Key Metric)	Critical Error
Not Sampled	Unknown (alert will not be seen in the MOM console)

Default SQL Server and MS .NET metrics MOM actions definitions

Alerts is delivered with a set of predefined MOM alert actions for metrics monitored by Precise Microsoft technologies.

About creating customized metrics

You can add customized metrics to an AppTier (excluding the Cross-AppTiers), or delete customized metrics.

Alerts allows you to monitor any performance aspects using pre-defined metrics for each AppTier. For data that is not collected by any of the pre-defined metrics, you can create new customized metrics. (Only users with Administrator privilege are allowed to define customized metrics.)

For example, you can create a customized metric that uses a UNIX shell that collects data about the memory size allocated for specific processes (in this case, the metric type is Table because it collects multiple values). The metric will return the data by using a host script. Alerts will display the collected data as a list in the Current tab. The History tab will display the processes behavior over time. The Events tab will trace the alert levels produced by this metric including its failures (Not Sampled status).

When creating a customized metric, it is associated with all instances of an existing AppTier. The customized metrics are part of the Customized set. You can also edit the Thresholds, Sampling, and Actions for each customized metric individually.

To sample customized metrics, InformPoint must be installed on the sampled instance server. In the case of a remote SQL server instance, the InformPoint that samples the instance is the one installed on the remote SQL server collector.

Creating customized executable files

A customized metric can be associated with a host script that you create, so that Alerts operatively monitors a parameter specific to your environment.

Save the host script in the server machine where the sampled instance is running, in the directory:

`<i3_root>\products\pulse\userprograms`.

Although the root directory of the running script is `<i3_root>`, the program runs from the `userprograms` directory.

To simulate a run of customized metric, use a command line shell and type from the Precise root directory:

For Windows operating systems `.\products\pulse\userprograms\MYPROG.bat`.

For UNIX or Linux operating systems `./products/pulse/userprograms/MYPROG.sh`.

You can create a customized metric that returns a single value or a list of values. A customized metric that returns a single value is created by apply the following guidelines:

- The script must return a single value to the Standard Output (`echo` command). The type of the value must be numeric or float. Alerts compares this value to the customized metric thresholds to determine the metric alert status.
- The script must output only the value. To avoid that commands be printed, Windows scripts must start with `@echo off`. UNIX scripts must start with the shell type, for example `#!/bin/ksh`.
- You can use dynamic parameters for customized metrics in the command line.

See Using dynamic parameters in customized metrics.

A customized metric that returns a table with multiple values is created by apply the following guidelines:

- The script must return a list of values to the Standard Output (`echo` command). Each line in the output must contain a name and a value separated by a tab (\t only, not a space). The value must be numeric or float. Alerts compares these values to the customized metric thresholds to determine the metric alert status.
- The script must output only the value. To avoid that commands be printed, Windows scripts must start with `@echo off`. UNIX scripts must start with the shell type, for example `#!/bin/ksh`.
- In some operating systems, the tab character may be ignored when printing it using the echo command. In those cases, use the ' character around the text, as follows: `echo 'Alerts 80'`.

You can use dynamic parameters for customized metrics in the command line. See Using dynamic parameters in customized metrics.

> It is recommended to test your scripts in your environment before running them through Alerts. For UNIX scripts, verify that the script has rx security permissions.

Windows script examples

The examples for Windows scripts, whose file names end with _example, are listed in the following tables. These files can be found also in any server where InformPoint is installed, in the directory:

`<i3_root>\products\pulse\userprograms`

The table below shows an example of customized metrics with a single value.

Table 16-9 Example of customized metrics with a single value - Windows

Script file	Script lines	Description
simple_example.bat	@ECHO OFF ECHO <value>	This script returns a constant value that creates a straight line graph.
simple2_example.bat	@ECHO OFF <program_run_command> ECHO %ERRORLEVEL%	This script indicates whether or not a certain program is running. The script returns 0 if the program runs with no errors, and n>0 if errors occurred. You can set Near-Critical threshold to 1 to alert each time the program fails.

The table below shows an example of customized metrics with multiple values.

Table 16-10 Example of customized metrics with multiple values (table type) — Windows

Script file	Script lines	Description
table_example.bat	@ECHO OFF ECHO Demo 0 ECHO Demo2 1 ECHO Demo3 2	This script returns a list of items including their values for each sample. A tab character (not space) separates between the item name and the item value.

UNIX/Linux shell script examples

The examples for UNIX/Linux shell scripts, which their file names end with _example, are listed in the following tables. These files can be found also in any server where InformPoint is installed, in the directory:

`<i3_root>/products/pulse/userprograms`.

The table below shows an example of customized metrics with a single value for UNIX or Linux scripts.

Table 16-11 Examples of customized metrics with a single value (UNIX/Linux)

Script file	Script lines	Description
simple_example.sh	#!/bin/ksh echo <value>	This script returns a constant value that creates a straight line graph.
simple2_example.sh	#!/bin/ksh if test -e test.	This script counts the number of characters of a specified text file. For example, you can set the metric thresholds to alert when the file size exceeds a certain size.

	txt then cat test.txt | wc -c else echo 0 fi	

The table below shows an example of customized metrics with multiple values (table type) for UNIX or Linux scripts.

Table 16-12 Example of customized metrics with multiple values (table type) - UNIX/Linux

Script file	Script lines	Description
table_example.bat	#!/bin/ksh echo 'Demo 0' echo 'Demo2 1' echo 'Demo3 2'	This script returns a list of items including their values for each sample. A tab character (not space) separates between the item name and the item value.

Creating customized stored procedures

To monitor a parameter specific to Oracle or SQL Server AppTiers, you can create a stored procedure and connect it to a customized metric of Alerts. The type of your stored procedure, Oracle or MS-SQL, must be according to your PMDB type.

In addition, the stored procedure can be defined only in Oracle or SQL Server AppTier instances. To create Oracle stored procedure metrics, you must install the Precise for Oracle FocalPoint with at least one instance. To create MS-SQL stored procedure metrics, you must install the Precise for SQL Server FocalPoint with at least one instance.

Creating MS-SQL stored procedures

When creating a stored procedure in MS-SQL, apply the following guidelines:

To create MS-SQL stored procedures

1. Connect to the database as Precise for SQL Server user.
2. Ensure that the name of the stored procedure includes the database name. For Oracle users, the procedure name must not end with a semicolon (;).
3. The naming convention for the program in Alerts must be: `databaseName..Procedure()`.
4. Verify that the return value of the execution is a single numeric value (integer).

> You can use any procedure and an unlimited number of parameters, including no parameters.

To create an MS-SQL stored procedure

1. Apply the following format:
   CREATE PROCEDURE user_def @@var1 type1, ..., @@var-n type-n
   AS
   - Custom query
   GO
2. Use the following format within the call from the customized metric (the code should be entered in the Program box when adding a new Customized Metric):
   user_database..user_def val1, ..., val-n

Example of an MS-SQL stored procedure

The following stored procedure enables Alerts to send an email when the number of rows in a table called SALES.ITEMS reaches an amount that you set in the Thresholds tab.

To apply the MS-SQL stored procedure

1. Connect to the database schema of the Precise for SQL Server.
2. Create the function `user_def` in the user_database as follows:
   CREATE PROCEDURE user_def @@sampling_rateint,@@sampling_period int
   AS
   SELECT count(*) FROM SALES..ITEMS
   GO
3. When you create the customized metric under the MS-SQL instance, set the program name to:
   SALES..user_def @SAMPLING_RATE, @SAMPLING_PERIOD

Use dynamic parameters. See Using dynamic parameters in customized metrics.

To test the MS-SQL stored procedure

1. Connect to the database schema of the Precise for SQL Server.
2. Run the function `user_def` in the user_database as follows:
   user_database..user_def sampling_rate , sampling_period

Creating Oracle stored procedures

When creating a stored procedure in Oracle, apply the following guidelines:

To create Oracle stored procedures

1. Connect to the database as Precise for Oracle user.
2. Ensure that the name of the stored procedure includes the package name and ends without a semicolon (;).
3. The naming convention for the program in Alerts must be: `PackageName.ProcedureName()`.
4. Verify that the return value of the execution is a single numeric value (integer).

To create the Oracle stored procedure package

- Apply the format as follows:
  Create or replace package user_defined_pack as
  function user_def(var1 type1, ... , var-n type-n)
  return return-type;
  pragma restrict_references(user_def,WNDS,WNPS);
  end user_defined_pack;
  /

To create the Oracle stored procedure package body and insert your custom code into the user defined functions

1. Apply the following format
   Create or replace package body user_defined_pack as
   function user_def(var1 type1, ..., var-n type-n) return
   return- type is
   begin
   -- Custom
   code end
   user_def;
   end user_defined_pack;
2. Use the following format within the call the call from the customized metric (the code should be entered in the Program box when adding a new Customized Metric):
   user_database..user_def val1, ..., val-n

You can use any package or function name, and an unlimited number of parameters (including NONE).

Example of an Oracle stored procedure

The following stored procedure enables Alerts to send an email when the number of rows in a table called ITEMS, which belongs to a user called SALES, reaches an amount that you set in the Thresholds tab.

To apply the Oracle stored procedure

1. Connect to the database schema of the Precise for Oracle.
2. Create the package `user_defined_pack` as follows:
   Create or replace package user_defined_pack as
   function user_def(sampling_rate number, sampling_period
   number, warning_threshold number, critical_threshold number)
   return number; pragma
   restrict_references(user_def,WNDS,WNPS);
   end user_defined_pack;
   /
3. Run the following script in the Precise for Oracle database schema to create the package body of package user_defined_pack:
   Create or replace package body user_defined_pack as
   function user_def(sampling_rate number, sampling_period
   number, warning_threshold number, critical_threshold number)
   return number is
   tbl_num_rows number; BEGIN
   select num_rows into tbl_num_rows from sys.dba_tables where
   table_name='ITEMS' and owner='SALES';
   return (tbl_num_rows);
   end user_def;
   end user_defined_pack;
   /
4. When you create the customized metric under the Oracle instance, set the program name to:
   user_defined_pack.user_def (@SAMPLING_RATE , @SAMPLING_PERIOD ,
   @NEAR_CRITICAL_THRESHOLD , @CRITICAL_THRESHOLD)

Use dynamic parameters. See Using dynamic parameters in customized metrics.

To test the Oracle stored procedure

1. Connect to the database schema of the Precise for Oracle.
2. Run the function user_def in the user_database with your values as follows:
   select user_defined_pack.user_def(sampling_rate,
   sampling_period, warning_threshold,
   critical_threshold) as value from dual;

Using dynamic parameters in customized metrics

When you set a customized metric, you can use dynamic parameters in the Program command line.

> The dynamic parameters in customized metrics are not the same as used in actions.

The table below describes the dynamic parameters that you can use in customized metrics.

Table 16-13 Dynamic parameters

Dynamic parameter	Definition
@METRIC_ID	ID of the customized metric.
@METRIC_NAME	Name of the customized metric.
@INSTANCE_ID	ID of the instance that is sampled by the customized metric.
@INSTANCE_NAME	Name of the instance that is sampled by the customized metric.
@SERVER_MACHINE_NAME	Name of the server machine on which the InformPoint agent is installed.
@INSTANCE_SERVER_MACHINE_NAME	Name of the server machine on which the instance is running. (May be differ from @SERVER_MACHINE_NAME only on MS-SQL instances.)
@TECHNOLOGY_CODE	The code of the metric's technology.
@SAMPLING_RATE	Sampling rate (in minutes).
@ANSI_CURRENT_TIME	Timestamp of the sampling.
@WARNING_THRESHOLD	Warning (near critical) threshold value.
@CRITICAL_THRESHOLD	Critical threshold value.
@INCLUDE_LIST	Include list of the items to consider when sampling data (the format is: value1, value2, value3, ...). The values of this parameter equal to the values that you set in the Consider only the following items text box in the threshold tab. See Defining thresholds on the Thresholds tab.
@EXCLUDE_LIST	Exclude list of the items to ignore when sampling data (the format is: value1, value2, value3, ...). The values of this parameter equal to the values that you set in the Ignore the following items text box in the threshold tab. See Defining thresholds on the Thresholds tab.