Precise 10.3

Administration Guide

Exported on 03/16/2021

IDERA

Table of Contents

The following topics are available in this guide:

- AdminPoint basics
- AdminPoint tab
- Configuring AdminPoint settings
- Administering the PMDB
- Utilizing the PMDB
- Configuring PMDB process parameters
- Configuring Precise for J2EE
- Configuring Precise for Web
- Configuring Precise for Oracle
- Configuring Precise for BW
- Configuring Precise for SAP
- Configuring Precise for Microsoft .NET
- Configuring Precise for Tuxedo
- Administrating Precise communication
- Configuring Alerts settings
- Advanced settings
- Using custom J2EE instrumentation

AdminPoint basics

This section includes the following topics:

- Launching Precise AdminPoint
- About the tab selection bar
- About the Precise bar in AdminPoint
- About the filtering mechanism

Launching Precise AdminPoint

AdminPoint is a central application that lets you manage your Precise product suite from one location. Through AdminPoint, you can view, control, and configure all major Precise components, such as agents or FocalPoints. You can also define other settings across products, such as environments, SLAs, or Groups. If you ever need to contact Precise Technical Support, AdminPoint provides an easy way of assembling the required support information.

In addition, AdminPoint lets you manage all relevant PMDB process settings. The PMDB is an integrated database that Precise uses to store historical performance and statistics data for all products and instances.

AdminPoint can be launched a number of ways, including via StartPoint. You can also launch to AdminPoint from any of the Precise product screens.

> ⓘ To launch directly into AdminPoint, use your browser to jump to http://url:port/admin.

To launch AdminPoint

- On the StartPoint screen, click **AdminPoint** on the Precise bar. The Precise AdminPoint Dashboard tab opens. See About the Dashboard tab.

About the tab selection bar

The tab selection bar, at the top of the AdminPoint screen, lets you view any of the following tabs:

- Dashboard
- Agents
- Licenses
- Setup
- Warehouse

In each tab, specific command buttons let you perform actions on the kind of information that you are viewing. The AdminPoint user interface includes the following elements:

- Precise bar menu for AdminPoint
- Tab selection bar
- Main area and details area for a selected tab

See the appropriate section for specific details regarding a tab.

About the Precise bar in AdminPoint

The Precise bar lets you configure Favorites, support, Precise settings, or launch to StartPoint. The following table gives a description of the screens available on the Precise bar.

Table 2-1 Precise bar in AdminPoint

Icon	Name	Description
	Back	During a work session, Precise products keep track of where you have navigated to. The Back, Forward, and History buttons enable you to navigate between previously visited views. The Back control displays your previous view.
	Forward	Enables you to navigate to the next view. This button is only enabled if you clicked Back, or if you chose a history option.
	Stop	Stops a request for information from the server.
	Refresh	Updates the data currently displayed.
	Home	Navigates to the highest level entity, usually the instance or AppTier (all instances). The time frame settings remain the same.
	Send	Opens new email message in your email program with the link to the current environment in context.
	Help	Opens the online help in context.
	Favorites	Enables you to add or remove favorites in your Precise Favorites list. See Adding, viewing, and deleting Favorites.
	Settings	Lets you define Precise settings in StartPoint. See Launching StartPoint.

About the Print function

The new Print function includes the following features:

- Capability to export a page from Precise to an .htm format (Save option). The saved file will include all the information contained the original file but all mouse functions will be disabled except for the Print option.
- Capability to save the print page to the local disk.
- User-friendly and resizable print functionality.

> ⓘ The Save option is only possible within the same network of the installation. When used outside the network it may include broken links to images (instead of the image a frame with an X may be displayed).

Recommended print settings

For best fit and presentation we highly recommend you make the following changes to your print settings:

- Configure print options to landscape page layout.

- With the Internet Explorer open, go to **Tools>Internet Options>Advanced tab**, and in the Printing section verify that the Print background colors and images check box is marked.

To print wide tables with many columns, resize table columns to fit the landscape layout. Columns that overflow the landscape width will not be printed.

Additional information

- Note that when saving a file with a graph, the graph image is saved as a separate .bmp file in the same folder. If you want to move the print file to a different folder you must also move the .bmp file to that folder.
- New files are saved with a default name and path. To prevent inadvertently deleting a previously saved file, enter a different name and/or path for new files.

To print, save, or preview a currently displayed screen

1. Click **Print** on the Precise bar.
2. In the Print Option dialog, select the area(s) of the screen you want to print or save.
3. Click **OK**.
4. On the resizable preview screen, the dashed line on the right indicates the right margin. Change table column widths to ensure the information you want to print or save is within this margin.
5. Click the Print or Save icon at the top right-hand side of the screen, follow the displayed dialog box instructions and click **Print** or **OK**.

Adding, viewing, and deleting Favorites

The Favorites function allows you to save a specific location in Precise and to retrieve the same location later without having to navigate to it. For this purpose, a friendly user interface similar to the familiar Favorites option in the Internet Explorer has been implemented.

About the Favorites function

The new Favorites function includes the following features:

- **Relative Time Frame**. Saving relative time frame instead of static date. For example, saving the last 7 days will always display the last 7 days, depending on the day entered.
- **One click to specific location**. Once you open Precise by launching a saved Favorite item, you will not have to enter a login credential nor click the login button.
- **IE Favorites support**. Adding a new Favorite item in Precise will also add it to the IE Favorites menu.
- **Auto Complete**. The Favorites dialog includes a new combo box which supports Auto Complete.
- **Auto Naming**. The Favorites dialog generates item names based on the current location.

To add a new Favorite location

1. On the Add/Delete Favorites dialog box, enter the name of the new Favorites entry.
2. Click **Add**. The dialog box is closed and the new Favorite is added to the list.

To view a Favorites location

1. On the Precise bar, click **Favorites**.
2. Select the Favorites location you want to view.

To delete an existing Favorite location

1. On the Add/Delete Favorites dialog box, select the Favorite location to be deleted.
2. Click **Delete**. The dialog box closes and the selected Favorite is deleted from the list.

> ⓘ The favorite address is displayed in the **Address** field and cannot be edited.

Generating a support file

When you contact Precise Technical Support you need to include in your help request a set of log files that assists the Support staff to determine which components cause the current problem. The Gather Support Information dialog box provides a quick and convenient way of generating all required files and collecting them in one zipped support file. You can create an express support file or a full support file.

To generate a support file

1. On the Support menu, in the AdminPoint toolbar, click **Support**.
2. In the Gather Support Information dialog box, select the servers that experience the problem and click the right arrow to move them to the Selected Servers list.
3. Select one of the following options:
   - **Express**. Creates a support file that holds detailed information about the selected server so that Support staff can determine the faulty component on the server.
   - **Full**. The Full support file works exactly the same way as Express, but provides more files and information. The servers are still selected by the user.
4. Click **Create Support File**. A dialog box indicating the progress of the support file creation opens. Creating the file may take several minutes.
5. Open the support file, or save it locally on your computer.
6. Send the file to Precise Technical Support.

> ⓘ Unless instructed otherwise by Precise Technical support, select express support.
> Precise automatically adds support information from related FocalPoint servers even though you have not explicitly selected these servers.
> Support files require disk space on the main Precise FocalPoint machine. For an Express support file, 5M is needed per server. For a Full support file, 100M is needed per server.

Sending an email message

You can send an email message to one or more recipients from the Precise toolbar. The default subject for the message will be "Link to a Precise environment".

The email will include a link to the Precise product in the current context (time frame and selected entries).

To send an email message

1. Click the email icon on the Precise toolbar. The default email program opens.
2. Fill in the required fields and click **Send**.

Determining which table columns to display

Tables are used to display information about a set of related entities in the Main and Association areas. It is possible to determine which columns to display in the Association area tables.

To determine which columns to display in the Association area

1. Click the Table icon on the upper right-hand side of a table and select **Column Chooser**.
2. In the Table columns dialog box, click the arrows to move the names of the columns that you want to display to the Visible box and the ones that you do not want to display to the Invisible box.
3. Click **OK**.

About Clear All Filters

In StartPoint the filtered table columns can be cleared.

To clear the filters

- Click the Table icon on the upper right-hand side of a table and select **Clear All Filters**.

The table will be refreshed.

Copying data to the clipboard

At times you may want to save data displayed in the table area in a Microsoft Excel spreadsheet for further analysis or save an image of a graph to the clipboard.

To copy data displayed in the Association area to the clipboard

- Click the Table icon on the upper right-hand side of a table and select **Copy to clipboard**.

The table can be pasted into Microsoft Excel or as an HTML file.

To copy a graph to the clipboard

- Right-click a graph and choose **Copy to clipboard**.

You can now paste the image into any application that works with the clipboard.

Exporting to ASD

The Export to ASD (Application Service Dashboard) Portlet feature enables you to export the view of the chosen table or graph and generate a portlet with that view in ASD, so that it will provide you with another way of monitoring your environment.

Prerequisites

To be able to use this feature, you need to have the following rights in Precise:

- View permissions to all AppTiers in the environment

If you do not have sufficient rights, you will get an error message when trying to execute this feature.

Exporting the information

You can either export a table view or a graph view.

> ⓘ The name field has the following restrictions: maximum 100 characters.

To export a table view

1. Click the Column Chooser icon.
2. Select **Export to ASD Portlet**.
3. Insert a name in the name field that clearly describes the table view.
4. Click **OK**.

To export a graph view

1. Right-click the graph.
2. Select **Export to ASD Portlet**.
3. Insert a name in the name field that clearly describes the graph view.
4. Click **OK**.

Launching StartPoint

StartPoint can be launched via AdminPoint.

To launch StartPoint

- On the AdminPoint screen, click **StartPoint** on the Precise bar. The Precise StartPoint opens.

About the filtering mechanism

This section includes the following topics:

- Top *n* filtering
- Retention of totals and partial filtering
- Two-phase filtering
- Ranking formula
- Technology-specific implementations
- Additional filtering during summary

Top *n* filtering

The filtering mechanism is based on top *n*. We rank all activities and then apply the top *n* filtering. Subsequently, only the top *n* activities will be loaded. The rest will be filtered out.

This approach has two major benefits:

- **One threshold fits all**. The same n threshold can fit both production and test systems. Loading top activities ensures that most important activities will not be filtered out.
- **Predictability**. The *n* threshold limits the number of rows loaded into the PMDB. Thus, it should be a lot easier to predict the PMDB size.

Some technologies collect data and prepare multiple performance tables to be loaded. When necessary, multiple *n* thresholds are defined to allow a more granular control on the filtering.

All filtering parameters are defined in the PMDB (per technology) Load process. Filtering thresholds (as well as any other PMDB process parameters) can specify a default value per technology plus specific values for selected instances.

Retention of totals and partial filtering

> ⓘ Retention of totals and partial filtering is not new. It has been in use since version 7.1.

If the filtering mechanism just filters out rows, the sum of all remaining rows will be less than the actual totals. Thus, all filtered rows are aggregated to one row. This row will be loaded into the PMDB and will retain the totals. The value of all identifiers in the filtered row is "_I3OTHER_".

Most string identifiers are not kept in the statistics (fact) tables. String values are kept in dimension tables. Integer keys are kept in the statistics tables instead. In all dimension tables a zero key is reserved for the "_I3OTHER_" value.

Some technologies (mainly databases) use partial filtering, meaning that few identifiers of the filtered rows are kept as is (protected identifiers). Other identifiers are changed to "_I3OTHER_". The filtered rows are aggregated and loaded into the PMDB. Instead of loading only one row to retain totals, multiple rows of partially filtered rows are loaded.

The amount of partially filtered rows depends on the number of different combinations of values of the protected identifiers. To control the amount of partially filtered rows, version 8.5 introduced Two-phase filtering.

Two-phase filtering

This filtering mechanism works as follows:

1. Aggregate all raw data groups by all identifiers of the table.
2. Sort aggregated rows, ordered by the primary counter, descending.
3. Calculate rows count as the number of aggregated rows.

Calculate instance totals as the instance level totals for the primary counter. Calculate top *n-1* totals as the primary counter totals for the top *n-1* rows.

1. If rows count is less than the top *n* parameter, then exit.
2. If top *n-1* totals divided by instance totals is greater than 0.9, then skip to step 9.
3. For each aggregated row which is not part of the top *n-1* rows, change less important identifiers to _I3OTHER_ (partial filtering).
4. Aggregate again to combine _I3OTHER_ rows.

Because we did partial filtering, _I3OTHER_ rows will be aggregated to more than one row. These rows will compete with the top *n-1* rows in step 9.

1. Sort aggregated rows, ordered by the primary counter, descending once again.
2. Aggregate all rows that are not part of the top *n-1* rows into one row (the *n* row).

Example of Precise for Oracle using the filter mechanism

Table `PW_ORSO_STMT_OBJ_STATS_T` is the most detailed statistics table. It contains I/O and locks counters for combinations of Program, User, Machine, Statement, File, and Object (Table or Index).

Prior to version 8.5, we had to choose between two bad alternatives:

- If we specify the Object identifier as non-filtered, hundreds of partially filtered rows are loaded, no matter how high the filtering threshold is. At least one row per accessed object will be loaded regardless of its access time.
- If we specify the Object identifier as filtered, we may miss a top accessed Object just because it is accessed by many different Statements and Users.

Since version 8.5:

- We will specify that an Object is a non-filtered identifier.
- Objects that are accessed by many Statements and Users will be aggregated per Object. These partially filtered rows will compete with the non-filtered rows. Only the most accessed Objects will make it into the top *n*.

This way we can protect our most important identifier and at the same time avoid flooding the PMDB with low values of that identifier.

Ranking formula

Products that consider Service Time (Avg) as more important than Service Time (Summed) and also use Service Time (Avg) as the default sort field have additional considerations.

Defining Service Time (Avg) as the primary counter is problematic. Sometimes excessive execution of a highly responsive transaction is the main problem. We do not want to lose that transaction for one-off slow transactions.

Defining Service Time (Summed) as the primary counter is problematic as well. The activities with worst Service Time (Avg) might be filtered out.

Thus, in version 8.5, Precise for Web and Precise for J2EE define the primary counter as a formula. This formula is used to rank all activities by the top *n* mechanism.

The ranking formula is: Service Time (Avg) * Logy Executions.

The user can influence the weight of Executions as High, Medium, or Low. The filtering mechanism will chose the y base of the log function as follows:

- **High**. No log function. Ranking based on Service time (Summed). If an activity is executed 1,000,000 times, it will be weighted 1,000,000 times more.
- **Med**. Base is 2. If an activity is executed 1,000,000 times, it will be weighted 20 times more.
- **Low**. Base is 10. If an activity is executed 1,000,000 times, it will be weighted 6 times more.

Technology-specific implementations

Oracle

Precise for Oracle loads three sets of data:

- Programs (plus users, machines, modules, actions, and work types)
- Statements (plus PL/SQLs, execution plans, and all the above)
- Objects (plus files, Storage devices, and all the above)

Each combination of program, user, etc. may execute many SQL statements. Each combination of program, user, and statement may access several objects. Thus, the objects table may grow very fast.

By default, Precise for Oracle will load only top 300 programs, top 500 statements, and top 700 objects per 15 minutes. Top items are ranked according to In Oracle (Summed).

In Oracle (Avg) is not taken into consideration at all. The focus here is on top resource consumers.

To change the defaults

- Go to **AdminPoint>Warehouse>Processes** and select **Load Data process of Oracle technology>Parameters**.

Precise for Oracle uses the Two-phase filtering technique as follows:

- **Program**. After the first aggregation phase, all records that are not part of the top 300 are partially filtered by protecting all programs, modules, and users. All other identifiers are converted to _I3OTHER_ and then the second aggregation phase takes place. The partially filtered records compete with the original top 300 records. When an ERP extension is installed, the user is overridden by the ERP user. In this case it will not protect the user any more. If many SAP users executed the same SAP transaction, after second aggregation phase all of them will be aggregated into one record. So, this protected SAP transaction is promoted and may make it into the top 300 programs.
- **Statement**. After the first aggregation phase, all records that are not part of the top 500 are partially filtered by protecting all statements and execution plans. All other identifiers are converted to _I3OTHER_ and then the second aggregation phase takes place. The partially filtered records compete with the original top 500 records.

If many users and/or programs executed the same statement, after the second aggregation phase all of them will be aggregated into one record. So, this protected statement is promoted and may make it into the top 500 statements.

- **Object**. After first aggregation phase, all records that are not part of the top 700 are partially filtered by all objects' name, schema, and type. All other identifiers are converted to _I3OTHER_ and then the second aggregation phase takes place. The partially filtered records compete with the original top 700 records.

If many statements and/or programs access the same object, after the second aggregation phase all of them will be aggregated into one record. So, this protected object is promoted and may make it into the top 700 objects.

DB2

The Precise for DB2 load mechanism works the same as Precise for Oracle. The only difference is: Precise for DB2 collects and loads only the first two sets of data: Programs and Statements.

The "Calculate Objects Usage" PMDB process calculates the Objects data daily based on the Statements performance and execution plan.

SQL Server and Sybase

Precise for SQL Server and Precise for Sybase implement a similar top n filtering mechanism. Both technologies generate and filter only one set of records: each record contains a combination of database, user, program, login, machine, statement, and batch.

The Two-phase filtering technique is used as follows:

- After the first aggregation phase, all records that are not part of the top 500 are partially filtered by protecting all databases, users, and programs. All other identifiers are converted to _I3OTHER_ and then the second aggregation phase takes place. The partially filtered records compete with the original top 500 records.
- If a user and/or program executed many statements, after the second aggregation phase all of them will be aggregated into one record. So, this protected combination of database, user, and program is promoted and probably makes it into the top 500 statements.

As specified above, by default only top 500 statements will be loaded. To change the default: go to **AdminPoint>Warehouse>Processes** and select **Load Data process of the SQL Server or Sybase technology>Parameters**.

Web

Precise for Web has both pre-aggregation and post-aggregation filtering.

- Pre-aggregation filtering
- Post-aggregation filtering

Pre-aggregation filtering

By default, a client IP address is not loaded into the PMDB. The original client IP address is converted to Location (if defined), Country, State, and City (if enabled and if found in the Precise IP addresses database). Then, the original IP address is replaced with the "N/A" value.

When the raw data will be aggregated, the thousands of distinct IP addresses will be aggregated into a few dozen combinations of: location, country, state, and city.

If the monitored application is a local intranet application serving less than 100 distinct client IP addresses, you may consider enabling Collect Client IPs in **AdminPoint>Warehouse>Processes** and select **Load Data process of the Web technology>Parameters**.

Post-aggregation filtering

Precise for Web loads two different sets of data:

- **Web server**. Major entity is URL.
- **Web client**. Major entity is Web Page.

For each of the two sets you can configure the maximum number of rows to be loaded. Top records will be loaded based on a formula that weights both average service time and number of executions as described in Ranking formula.

By default, top 1000 URL records and top 100 Page records will be loaded each 15 minutes. In both sets, the weight of executions is MEDIUM. If you want to focus on the worst average response time, change the weight of executions to LOW. If you want to focus on the top resource consumers, change the weight of the executions to HIGH.

All the above can be configured for all Web instances or per instance using **AdminPoint>Warehouse>Processes** and select **Load Data process of the Web technology>Parameters**.

OS, MQ, Tuxedo, OA, and Other

Insight technologies have post-aggregation filtering.

Insight loads several different sets of data including OS, MQ, Tuxedo, OA, and Other. For each of these sets you can configure the maximum number of rows to be loaded. Top records for MQ, Tuxedo, OA, and Other will be loaded based on a formula that weighs both the average service time and the number of executions, as described in the Ranking formula. Top records for OS will be loaded based on a formula that weights both average memory consumption and CPU consumption, similar to the method described above for the other technologies.

By default, the top 200 records will be loaded every 15 minutes. In all sets, the weight of executions is **MEDIUM**. If you want to focus on the worst average response time, change the weight of executions to **LOW**. If you want to focus on the top resource consumers, change the weight of the executions to **HIGH**.

All the above can be configured for all instances or on a per instance basis using **AdminPoint>Warehouse>Processes** and selecting **Load Data process of the OS | MQ | Tuxedo | OA | Other technology>Parameters**.

Microsoft .NET, J2EE and TPM

Precise for J2EE has two data filtering mechanisms, one that takes place inside the client JVM (where it can be defined as activated or deactivated), and one that takes place within the PMDB. For information about data filtering inside the client JVM, see "About configuring data filtering" on page 105.

Version 9.5 introduces an additional data aggregation interval for Precise for J2EE and Precise SmartLink - the Short Time Slice (STS), that can be defined as either 30 or 300 seconds. To configure this aggregation interval, see "Modifying the Precise Load Data slice size" on page 77.

Additional filtering during summary

All the above mentioned filtering refers to filtering the performance data prior to the load into the PMDB. This section deals with additional filtering applied after loading data into the PMDB, when summarizing data into daily, weekly, and monthly tables.

The following example illustrates the need for additional filtering.

For example, we load the top *500* SQL statements per instance, every 15 minutes. The daily summary process aggregates the entire day (96 time slices of 15 minutes each) into one row per SQL statement that contains its daily activity.

The best case scenario is when the application is static and the same top *500* statements are loaded every 15 minutes. In this case, the daily summary will only load 500 statements to the daily tables.

The worst case scenario is when a completely different set of top *500* statements are loaded. In this case nearly 50,000 (500*96) rows will be loaded into the daily tables per instance per day.

In real life, 500 rows times 96 slices will be aggregated to something between 5,000 to 20,000 rows (depending on the application).

With additional filtering, while aggregating the rows from the time slice tables, only the top *n* rows will be loaded into daily tables (plus one row per hour group that retains totals).

The same mechanism is applied when summarizing daily tables into weekly tables and monthly tables. By default, the *n* threshold is defined per summary level as follows:

- **Daily**. Top 2,000 rows will be loaded into daily tables.
- **Weekly**. Top 3,000 rows will be loaded into weekly tables.
- **Monthly**. Top 3,000 rows will be loaded into monthly tables.

The Precise administrator can modify these three thresholds per technology or per instance, using the **Parameters** button when the PMDB Summarize Data process is selected.

In version 9.0, the new STS (short time slice) table is introduced for Precise for J2EE and Precise for SmartLink. The Precise administrator cannot modify the top n threshold value when summarizing a J2EE or SmartLink tree-like table from the STS level to the regular, cube-like, T (15 minute) level. This threshold is defined as the technology's default daily value, and this summary filters the top *n* trees (each tree is an activity with multiple rows), into a daily table of rows.

AdminPoint tab

This section includes the following topics:

- About the Dashboard tab
- About the Agents tab
- About the Installation tab
- About the Management tab
- About the Warehouse tab

About the Dashboard tab

The Dashboard tab provides an overview of Precise components. It displays information that allows the Precise administrator to determine (at a single glance) the health of his installations. This tab analyzes and reports on your system's current Precise status.

The Dashboard tab indicates the following situations:

- Data is not loaded to the PMDB
- Agents are failing
- FocalPoints are failing
- License Limitation Metrics which have been exceeded
- Licenses have expired or are near expiration
- Licenses that are installed
- Actions Items which are critical
- Agents are stopped
- Instances configured to experience downtime during the selected time frame

How the Dashboard tab is structured

The Dashboard tab is divided into two areas: the Category Details table (left pane) and the Main area (right pane). Each area can include different control elements, such as tabs and view controls, and displays information in various formats, such as tables.

The Category Details table area lists all the components, such as: Agents, FocalPoints, and Licenses, that are monitored by the Precise main FocalPoint, grouped by category.

The Main area provides information regarding the category selected in the Category Details area. Once a certain component is selected in the Category Details table, the upper portion of the Main area displays details (a list of the entities) for the selected category. Once a particular entity is selected in the upper portion of the Main area, the lower portion of the Main area displays additional relevant details for the entity. This information will help the user pinpoint a problem.

About the view controls

In the Dashboard tab, above the Category Details area, is the view controls bar. This bar includes:

- **Last Refreshed display**. This displays the date and time that the information was last refreshed.
- **Auto Refresh is off / Auto Refresh is on button**. This button enables you to control whether the information is refreshed automatically.
- **Node view control**. By default information is displayed for all Nodes.
- To display the overview status of all Precise components in a specific node, select the node from the drop-down menu.
- **Application view control**. By default information is displayed for all Applications.

- To display the overview status of all Precise components in a specific application, select the application from the drop-down menu.

About the Category Details table

The Category Details table displays all the components monitored by the Precise main FocalPoint, grouped into categories. The categories are displayed in the table by their error severity level, which is defined by the component in each category with the highest error severity level.

You can control the way data is displayed in the table. See "About sorting data" on page 184.

The Main area (right pane) will display tables based on the selection made previously on the Category Details table. The following table shows the information displayed in the Category Details table.

Table 3-1 Category Details table

Header	Description
Status	Indicates the highest severity error (if any) encountered by one of the category's components, as follows: 🟥 **Red**. Indicates that at least one of the category's components encountered a high severity error. 🟧 **Orange**. Indicates that at least one of the category's components encountered a medium severity error. 🟨 **Yellow**. Indicates that at least one of the category's components encountered a low severity error. ✅ **Green**. Indicates that none of the category's components encountered errors. ⬜ Indicates components configured to experience downtime in the defined time frame. Errors for this component are ignored. ⬜ Indicates problematic components that have been acknowledged and are currently being handled. These components, and the errors resulting from them, are excluded from their category's total and are grouped together in this category. For more information, see Defining problematic components as acknowledged.
Category	Indicates a group per technology, such as: • Agents • FocalPoints • Warehouse processes • License Limitation Metrics • Licenses • Action items
Total components	Indicates the total number of components in the selected category.

Header	Description
Errors	The color-coded stacked bar indicates the number of underlying components that have an error. Only red (high) and orange (medium) severity errors are shown.
	Click on the icon on the header to switch between the following display methods:
	1. Show relative-sized colored toolbar according to the number of components that have problems. 2. Show full-sized colored toolbar. 3. Show the number of components that have problems.

About the Main area in the Dashboard tab

The Main area displays different information depending upon which category is selected in the Category Details table. Clicking on the hyperlink for specific data items launches, in-context, the tab from which the data was derived.

About the Detailed Description

The fish-eye displays text for the purpose of helping the user to understand and focus onto problems.

After selecting a category (Agents, FocalPoints, Warehouse processes, Licenses, Action items, or Acknowledged Items) in the Category Details table, the Detailed Description fish-eye displays text relevant to the selected category. For example: if Agents or Focalpoints are selected in the Category Details table, the Detailed Description fish-eye displays a detailed description of the current status for the selected Agents or FocalPoints. Recommendations to resolve the problem are also given.

Defining problematic components as acknowledged

Handling a problematic component and solving its underlying issues might take time, and continue over into additional time frames. By default, problematic components continue to produce error reports until the underlying issue is resolved. To avoid excess error reports stemming from the same problematic component, you can acknowledge the component as problematic and currently being handled. Components can be defined as acknowledged for a defined period of time or until specified differently.

While a component is defined as acknowledged, it doesn't appear as part of its original category, but rather as part of a general "Acknowledged Items" category. Errors related to this item are therefore not included as part of their category's total errors.

To define a component as acknowledged

1. In the fish-eye text describing the component's error, click **acknowledged**.
2. In the dialog box, select and define the desired acknowledgement time frame.
3. Click **OK**.

To cancel a component's definition as acknowledged

- In the fish-eye text describing the component's error, click **Cancel**.

About the FocalPoints category and the Agents category related table

Either the FocalPoints category or the Agents category may be displayed. The FocalPoints category lists all of the Precise FocalPoints and the Agents category lists all of the Agents and the status for each, in the selected category. Agents whose status is stopped will be reported as problematic. (This option can be controlled from the Admin Dashboard settings).

You can control the way data is displayed in the table. See Managing agents.

The following table shows the information displayed in the FocalPoints or Agents table.

Table 3-2 FocalPoints or Agents table headings

Header	Description
Status	Indicates the highest severity error (if any) encountered by the component, as follows: ■ **Red**. Indicates that the component encountered a high severity error. □ **Orange**. Indicates that the component encountered a medium severity error. □ **Yellow**. Indicates that the component encountered a low severity error. ☑ **Green**. Indicates that no component encountered errors. ▢ Indicates components configured to experience downtime in the defined time frame. Errors for this component are ignored.
Node	Name of the node.
Server	Name of the server on which the FocalPoint or Agent is running.
Agent	Name of the FocalPoint or Agent.
Description	A short description of the current status for the selected FocalPoint or Agent.
Criticality	Displays the related instance(s) criticality setting.

About the Events table

The Events table lists all the Precise events for the selected FocalPoint or Agent. Each row corresponds to an event. You can control the way data is displayed in the table. See "About sorting data" on page 184.

The following table shows the information displayed in the Events table.

Table 3-3 Events table headings

Header	Description
Level	Indicates severity: • Red stop sign indicates that a critical alarm has been raised. • Yellow warning sign indicates a near-critical alarm has been raised.
Date	Displays the date and time the event occurred.
Message	A short description of the current status.
ID	The ID (identification number) for the specific event.

About the Warehouse processes category related table

The Warehouse processes table lists all warehouse processes in the selected category and the status for each. Warehouse Load processes that have stopped loading data will be reported as problematic (This option can be controlled from the Admin Dashboard settings).

You can control the way data is displayed in the table. See "About sorting data" on page 184.

The following table shows the information displayed in the Warehouse processes table.

Table 3-4 Warehouse processes table headings

Header	Description
Status	Indicates the highest severity error (if any) encountered by the component, as follows: 🟥 **Red**. Indicates that the component encountered a high severity error. 🟧 **Orange**. Indicates that the component encountered a medium severity error. 🟨 **Yellow**. Indicates that the component encountered a low severity error. ✅ **Green**. Indicates that no component encountered errors. ⬜ Indicates components configured to experience downtime in the defined time frame. Errors for this component are ignored.
Node	Name of the node.
Instance	Name of the instance.
Server	Name of the server on which the instance is installed.
Product	Name of the product.
Process	Name of the process.

Header	Description
Last Run	The date of the last run for this process.
Criticality	Displays the related instance(s) criticality setting.

About the Licenses category related table

The Licenses table lists all licenses installed and the status for each. You can control the way data is displayed in the table. See "About sorting data" on page 184.

The following table shows the information displayed in the Licenses table.

Table 3-5 Licenses table headings

Header	Description
Status	Indicates the licenses installed and the status for each.
Server	Name of the server on which the FocalPoint or Agent is running.
Agent	Name of the FocalPoint or Agent.
Feature	A description of the license feature.
Licensing Status	One of the following statuses: • **Not Licensed**. The feature is installed, but the current license file does not include a product key for the feature. • **Licensed**. The feature has a license. • **Expired**. The feature was licensed, but its license has expired. This status usually appears when a feature is included in a trial license with an expiration date. • **License Expires in <x> Days**. The feature has a license, but the license will expire soon. This status appears when the expiration date occurs within the next 14 days.
Expiration Date	Expiration date for the currently installed license for a feature.

To use the Precise product suite, you need the appropriate license file. All product keys are saved in a single license file managed by Precise FocalPoint. The Precise license file contains information about the product features that you are licensed to use. It also contains data about your information system environment, including CPU, model, server name, or IP addresses of the servers where the products run.

A license can be either temporary or permanent. Temporary licenses expire after a specific date, while permanent licenses enable you to use the products indefinitely.

You can gather server information for each server and features in your environment by taking an agent snapshot. This information is needed to pass to your customer support representative when you want to obtain a permanent license file. The information is saved as an XML file on your local drive. The path and the filename appear in a message box.

To create an XML file of all servers and features in your application

1. Go to **AdminPoint > Dashboard**.
2. In the All Categories table, select **Licenses**.
3. In any of the License descriptions in the Main area, click **Apply another license**.
4. In the Apply License dialog box, click **Agent Snapshot**.

To create a file of all installed licenses in your application

1. Go to **AdminPoint > Dashboard**.
2. In the All Categories table, select **Licenses**.
3. In any of the License descriptions in the Main area, click **Apply another license**.
4. In the Apply License dialog box, click **Retrieve License**.

To apply a license

1. Go to **AdminPoint > Dashboard**.
2. In the All Categories table, select **Licenses**.
3. In any of the License descriptions in the Main area, click **Apply another license**.
4. In the Apply License dialog box, specify the location of the license file, or click browse to find and select the location.
5. Click **OK**.

About the License Limitation Metrics category related table

The License Limitation Metrics category related table list all License Limitation Metrics installed and the status for each. You can control the way data is displayed in the table. See "About sorting data" on page 184.

The following table shows the information displayed in the Licenses Limitation Metrics table.

Table 3-6 License Limitation Metrics

Label	Description
status	Indicates the License Limitation Metrics status for the installed technology/CPUs.
Technology	Technology to which the License Limitation Metrics applies.
Amount Installed	The number of instances of the relevant technology/CPUs that are installed.
Amount of Applied Licenses	The limitation of the number of instances of the relevant technology/CPUs.

About the Action Items category related table

The Action Items table lists all action items in the selected category and the status for each. You can control the way data is displayed in the table. See "About sorting data" on page 184. See About the Action items view.

The following table shows the information displayed in the Action Items table.

Table 3-7 Action Items table headings

Header	Description
Icon	The icons display the criticality of the Action Item related instance, whether high, medium, or low.
Status	Indicates the status of an Action Item.
ID	The Action Item ID number.
Node	Name of the node.
Instance	Name of the instance.
Server	Name of the server on which the FocalPoint or Agent is running.
Origin	the origin of an Action Item.
Status	Indicates the status of an action item, whether done, failed, pending, blocking, or executable.
Update time	The date of the last run for this process.
Criticality	Displays the criticality of the Action Item related instance, whether high, medium, or low.

About the Agents tab

You can view information on and manage Precise agents by using the Agents tab. In this tab, the table includes the following columns:

- **Agent Server**. Name of the server where the Precise agent is running.
- **Technology**. The technology type of the AppTier monitored by this agent.
- **Instance**. Name of the instance that is monitored by this agent (if relevant).
- **Application**. Name of the instance application. This name is displayed only for agents that have an instance specified in the instance column. Some agents do not have this name, for example FocalPoints.

This column displays all the environments that an instance belongs to.

- **Agent**. The name of this agent.
- **Product**. Name of the Precise product this agent belongs to.
- **Most Recent Error**. If problems occurred during the Installation Verification Procedure, the time the last Installation Verification Procedure occurred.
- **Status**. Current state of the specific Precise agent. See the explanation below.
- **Last Update Time**. The time when the agent status was last updated.
- **OS**. The operating system where this agent is installed.
- **Node**. Name of the node.

The current status of an agent can be any of the following statuses:

Icon	Description
	The agent is starting up.
	The agent is running.
	The agent is stopped.
	The agent is running but has encountered a problem. See the agent log files for details, verify the agent (using the **Verify** button), or contact Precise Technical Support.
	The agent is in failed status. This occurs in case the agent has problems. For example, in case it has crashed at least three times in an hour. See the agent log files for details, verify the agent (using the **Verify** button), or contact Precise Technical Support. ⓘ Once the agent is in failed status it keeps returning this status even if the original problem is fixed. To recover the agent from this status, fix the original problem, then start the agent using **AdminPoint>Agents tab**.
(Empty)	Precise FocalPoint cannot retrieve the current agent status. See the Precise FocalPoint logs and the agent log files for details. See Viewing a log file from the Agents tab.

You can control the way data is displayed in the table. See "About sorting data" on page 184.

You can select several agents to start/stop or refresh status. The following table describes the different procedures for selecting multiple agents.

Table 3-8 Selecting agents

Select	Action
An individual agent	Click a specific row in the table.
An arbitrary set of agents	Click the first agent row, hold down the <Ctrl> key, and then click all other required agent rows.
A consecutive group of agents	Click the first agent row, hold down the <Shift> key, and then click the last agent row.

Managing agents

Agent command buttons appear at the bottom of the Agents tab.

Using the Agent command buttons, you can perform the following actions on one or more agents in the table:

- Start one or more agents at a time.
- Stop one or more agents at a time.

- Refresh the status of one or more selected agents.
- View the log files that are available for a selected agent.
- Verify the installation by manually running the Installation Verification Procedure for the selected agent. By default, this procedure runs once a day. It validates that the agent is running properly. If it detects problems, a message appears on the screen, reporting the found errors. The time of the last error that was found during the Installation Verification Procedure is indicated in the Recent Errors column in the table. The findings of the Installation Verification Procedure process are also reflected in the Dashboard tab.

Installation Verification Procedure process findings are also reflected in the AdminPoint Dashboard tab.

About log files

Log files record information, warning, and error messages. They may be helpful to understand the scope of a problem and provide directions to solve it.

Viewing a log file from the Agents tab

In AdminPoint you can view a log file from the Agents tab.

To view a log file from the Agents tab

1. In the Display Agents table, select an agent and click **Log**.
2. In the Select Log File dialog box, select a log file and click **View**. The log file opens in a separate dialog box.

Saving a log file

In AdminPoint you can save the log file using the log file dialog box.

To save a log file

1. In the log file dialog box, click **Save**. The log file opens as HTML text in a new browser window.
2. Use your browser menu to save the file on your computer.
3. Close the browser window; then click **Cancel** to close the log file dialog box.

About the Installation tab

Through the Setup tab, you can view and manage various setup functions:

- Adding a new application
- Editing an existing application
- Customizing custom columns in the application
- About the Nodes view
- About the Applications view

Adding a new application

On the Installation tab, you can add a new application to the Precise system.

To add a new application

1. On the Installation view, click **Add**.
2. In the Add Application dialog box, enter the application name and select a node from the list.
3. Click **OK** to apply your settings, or click **Cancel** to close without saving the settings.

Editing an existing application

You can edit existing application via the Edit dialog box. In the Edit dialog box, you can perform any of the following editing options:

- To edit an existing application, in the Applications list, select the application you want to edit, and then click **Edit**.
- To rename an existing application, in the Applications Installer dialog box, click **Rename the Application Name**, and then click **Finish** to save the new name or click **Cancel** to close without saving.

Customizing custom columns in the application

You can customize the application's column headers and their contents as displayed on the application's table in both AdminPoint and StartPoint.

To customize column headers on a table

1. Click **AdminPoint > Intallation** tab.
2. Select the application, and then click **Customize Columns**.
3. Click **OK** to save the customized columns settings, or click **Cancel** to close the dialog box without saving.

About the Management tab

Through the Management tab, you can view and manage various functions:

- About the Updates View
- About the Action Items View
- About the Nodes View

About the Updates view

Known in previous versions as the Patch view, the Updates view in version 9.0 displays available updates and manages their application process. Using the Updates view, you can check which updates were applied to your servers, view the update properties, and launch the update application process. For more information about applying updates, see Applying an update.

To view and manage updates

1. In AdminPoint, select **Management > Updates**.
2. In the screen that appears, select one of the following display modes from the drop-down menu(s) above the table.
   - **Server Status**. This mode displays all servers that the update selected in the "For" drop-down menu should be applied to, and the current application status. The Server Status table displays the following information:

Header	Description
Update ID	Name of the update.
Server	Name of the server.
Status	An icon that indicates whether the update was applied successfully or with errors.

Header	Description
Application Type	Indicates how the update was applied to the server. The application types include: • **Applied (Directly)**. The update was applied directly on the selected server. • **Applied (Upon Installation)**. The update was automatically applied on the server when the server was installed. • **Applied (Indirectly)**. The changes and functionalities provided by the update were applied on the selected server when a newer update containing these functionalities was applied to it. • **Update Required**. The update should be applied on the selected server. • **Update Required (Priority)**. The updated must be applied on the selected server before it is applied on other servers in the same framework. A FocalPoint is installed on the selected server. Within each node, the servers with FocalPoints installed on them must be updated before the other servers. Therefore, the update must be applied on the selected (FocalPoint) server before it is applied on other servers in the same node. ⓘ Because there can be multiple nodes in a Precise installation, there can be multiple servers with this status. This status only refers to servers in the same node. You can update a server (without a FocalPoint) in a different node before updating the selected server, if the servers with FocalPoints installed on them in the other node have already been updated. • **Pending Prerequisites**. The update cannot be applied until all prerequisites are fulfilled. A list of prerequisites can be found per update on https://precise.secure.force.com. • **Pending Server-Update Version Upgrade**. The server version is not supported by the update. The server version must be upgraded before the update can be applied.
Applied On	The date and time that the update was applied on the server.
Node	Name of Framework node that the selected server is part of.
Installed Agents	List of all agents that are currently installed on the server.

• **Update Status**. This mode displays all updates currently loaded and their status for all relevant servers. The table displays the following information:

Header	Description
Update ID	Name of the update.

Header	Description
Server	Name of the server.
Status	An icon that indicates whether the update was applied successfully (insert icon) or with errors (insert icon).
Application Types	Indicates how the update was applied to the server. The application types include: • **Applied (Directly)**. The update was applied directly on the selected server. • **Applied (Upon Installation)**. The update was automatically applied on the server when the server was installed. • **Applied (Indirectly)**. The changes and functionalities provided by the update were applied on the selected server when a newer update containing these functionalities was applied to it. • **Update Required**. The update should be applied on the selected server. • **Update Required (Priority)**. The updated must be applied on the selected server before it is applied on other servers in the same framework. • **A FocalPoint is installed on the selected server**. Within each node, the servers with FocalPoints installed on them must be updated before the other servers. Therefore, the update must be applied on the selected (FocalPoint) server before it is applied on other servers in the same node. ⓘ Because there can be multiple nodes in a Precise installation, there can be multiple servers with this status. This status only refers to servers in the same node. You can update a server (without a FocalPoint) in a different node before updating the selected server, if the servers with FocalPoints installed on them in the other node have already been updated. • **Pending Prerequisites**. The update cannot be applied until all prerequisites are fulfilled. A list of prerequisites can be found per update at https://precise.secure.force.com. • **Pending Server-Update Version Upgrade**. The server version is not supported by the update. The server version must be upgraded before the update can be applied.
Applied On	The date and time that the update was applied on the server.
Node	Name of framework node that the selected server is part of.
Product	Name of the Precise components that are affected by the update.

You can control the way data is displayed in the table. See "About sorting data" and "About filtering data" on page 184.

Applying an update

To apply an update

1. Go to https://precise.secure.force.com and locate the desired update.
2. Download the update to a local directory on the computer from which you run the Precise GUI.
3. Go to **AdminPoint > Management > Updates**.
4. In the Updates view, click **Apply updates**. The Update Installation wizard appears.
5. In the "Select Updates" screen, locate the desired update in the Available Updates table. If the update does not appear, click **Add Update** and enter the file path from the folder you downloaded the update to. Alternatively, click the browse button, select the path, and click **OK**. The update now appears in the Available Updates table.

   > ⓘ For more information regarding a specific update, see the **Description** area at the bottom of the screen. To view all information for the update, scroll down or click the expand icon at the top right of the area to open a new window with the information.

   Selecting an update to appear in the Available Updates table does not apply the update. The update application process begins only after the servers have been selected.
6. Select the desired update in the Available Updates table and move it to the Updates to Install table using the arrows between the two tables.

   > ⓘ If there is a newer update that contains the update you selected, the newer update will be selected and appear in the Available Updates table.

7. Click **Next**. The Select Servers screen appears, displaying all servers and their update status.
   By default, all the servers will be selected to be updated. You can choose to exclude servers from the application process at this time by removing the selection mark next to them.

   > ⓘ In some cases, applying the update to one server will only be allowed if another server is updated at the same time. In these cases, you will be prompted to include the relevant servers in the selection to be updated.

8. Click **Next**. The update application "preparing" phase begins. If there are pre-action items to be executed, they will appear on the screen and must be completed to continue to the update application "applying update" process. Perform these action items.
9. Once all pre-action items are completed, the update application process will resume. If there were no pre-action items, the update application process will automatically continue.
10. The "Applying Updates" screen appears, displaying an update application process progress bar for each server.
11. After the update application is complete, post-action items will appear on screen. Perform these action items. You can decide to perform these action items at a later stage.

    > ⓘ If there are no pre- or post-action items, the wizard will automatically move from through the "Preparing", "Pre Action Items", "Apply update", and "Post Action Items" screens.

About the Action items view

The action items management mechanism tracks and automates manual action items required for installations and updates. You can manage action items by displaying which action items have been performed and which have not.

Through the Action Items view, you can check if instance-related action items are pending, failed, or completed. You need to perform all action items because some of the action items block the related agent startup. This means you cannot start the related agent until performing the action item.

You can manually perform any action item. Once you have performed it, notify Precise of this by selecting the action item and clicking the Mark as done button.

Some action items support automatic execution. For such action items the "execute" button is enabled. You can instruct Precise to automatically perform the action item for you by selecting the action item and clicking "Execute". You can select several action items and click execute or mark as done.

About the action items table

Using the Status combo at the top of the view you can select to view only pending action items, failed action items, or completed action items:

Header	Description
ID	The action items resource identification number; this ID number may be used more than once. For example, there may be several messages with the same Action ID for different instances.
Server	The name of the server where the action item was produced.
Technology	The technology type of the monitored instance. Exists only for action items which have a value in the instance column.
Instance	Name of the specific instance that is being monitored (if relevant).
Action Item	The title of the action item. After clicking an action item, the action item's full text is displayed lower on the screen. You can also view the action item's full text in the tooltip of this column.
Order	The order of Precise installation action items should be performed as specified - in ascending order. The specified order is necessary because of dependencies between action items.
Status	This is an icon which indicates the status of Precise installation action items. The action item status can be one of the following: completed, failed, or pending. In the case of a failed status, the tooltip describes the reason for failure.
Blocking	The icon indicates whether the start of the related agent is being blocked, until the action item has completed.
Origin	This the source of the action item. The source can be any of the following: install, uninstall, update, or verification.
Executable	A check mark indicates the action item is executable. An executable action item can be handled automatically by selecting the action items and clicking **Execute**. To execute the action item, additional user input (such as: user and password) may be required.

Header	Description
Node	Name of Framework node.
Update Time	Date and time when the action items were produced or last updated due to failure, execution or marked as done.
Required Fields	This column displays the name of fields requiring user input before automatic execution of the action item.

About icons for action items

The legend for the action items table is:

Icon	Description
☑ Done	The action item was completed successfully.
✖ Failed	Invocation of the action item failed. See this icon's tooltip for the failure reason.
⚠ Pending	The action item was not yet performed.
! Blocking	The action item's related agent cannot be started until the action item is performed.
✔ Executable	The action item can be automatically executed by Precise. Click execute to instruct Precise to execute it for you.

Displaying Dependencies on/off

On the Action Items view, the Dependencies On/Dependencies Off button will filter selected instances by displaying or not displaying their dependencies. By default, all instances and their action items are displayed as "Dependencies On".

"Dependencies On" signifies that dependent action items will be displayed in the Action Items table regardless of any filters set on the table.

To not display dependencies

- Select the instance from the Action Items table and click **Dependencies On** near the top of the screen.

Advanced Settings

On the Action Items view, the Advanced button lets you set the way you want to execute the action items.

If you have many action items, click **Advanced** and select **Expert mode**. Once **Expert mode** is selected, the executing action items prompt for required inputs only once (at the beginning of the operation), and only update the status of the executed action items. The status of the executed action items does not display any popup messages for each problem encountered while executing action items.

To execute action items by Expert mode

1. If you have many action items, click **Advanced**.
2. On the Settings dialog box, mark the Expert mode status check box.
3. Click **OK** to save the setting, or **Cancel** to close the dialog box without saving.

About the Nodes view

Through the Nodes view, you can add new Framework nodes, or edit properties, or remove an existing Framework node.

To view and manage Framework nodes

- On the AdminPoint screen, click **Management > Nodes**.

The Display Nodes table displays a list of installed nodes. You can add a new Framework node to the list, rename, or remove a Framework node from the list.

The Display Nodes table displays the following columns for Framework nodes:

- **Node Name**. Framework node name.
- **Proxy Server**. Server name or IP address of the node's Precise FocalPoint server.
- **Proxy Port**. Communication listener port number of the node's Precise FocalPoint server.
- **Primary Node**. Indicates the primary node.
- **URL**. URL used to access the node's GUI.
- **Proxy Relay to FocalPoint**. Indicates communication from the node's Precise FocalPoint to the main Precise FocalPoint is blocked, and that only the main Precise FocalPoint will access the node's Precise FocalPoint.
- **Proxy Relay to Servers**. Indicates that communication from the main Precise FocalPoint to the node's servers is blocked, and the main Precise FocalPoint will send requests through the node's Precise FocalPoint to the node's servers.

Managing Framework nodes

Using the Nodes command buttons, you can perform the following actions on one or more nodes in the Display Nodes table:

- Click **Add** to select between installing a new Framework node, or connecting an existing one. For more details, see the *Precise Installation Guide*.
- Click **Edit** to change properties, such as: enabling the proxy relay to the FocalPoint or to servers.
- Click **Remove** to uninstall or disconnect the selected node. For more details, see the *Precise Installation Guide*.

To edit a node (on the Nodes-Edit dialog box), enter the node name, proxy server name, and URL address. The following optional proxy settings can also be selected.

- **Enable the proxy relay to the FocalPoint**. Unchecked by default. Use only when, for security reasons, the node's Precise FocalPoint cannot directly access the Precise Main FocalPoint. Note: Its use slows down system performance.
- **Enable the proxy relay to the servers**. Unchecked by default. Use only when, for security reasons, the Precise Main FocalPoint cannot directly access the node's production servers. Marking this check box causes the Precise Main FocalPoint to access the node's production servers through the node's Precise FocalPoint.

> ⓘ Its use slows down system performance.

Generating an inventory report

The Inventory Report creates an excel file that lists details regarding installed servers, instances, License Limitation Metrics, and the PMDB. A generated report provides information for all nodes.

To generate an inventory report

- Click **Inventory Report**. The excel file will be downloaded to your computer, and can be saved locally.

About the Applications view

A Precise application is a collection of instances that are organized to work together. An application includes one or more application tiers (AppTiers), and each AppTier can consist of one or more instances of the same technology.

The Precise agents installed on an AppTier monitor, process, and communicate performance-related information for that tier.

By default, an AppTier carries the name of its technology (such as 'Oracle'). You can set the AppTier name to another significant name to be able to distinguish between different AppTiers. In AdminPoint, you can add a suffix to the technology name (such as 'Oracle_PeopleSoft'), rename or edit an existing AppTier, and add or delete AppTiers.

> ⓘ   Verify that you have renamed an AppTier before you add another AppTier of the same technology.

Every AppTier has instances assigned to it. Instances can be assigned to more than one AppTier. An instance can be copied or moved from one AppTier to another. Applications, AppTiers, and instances belong to only one node; therefore you can manage all these entities only within the same node. For example, you can copy an instance only to applications that are part of the instance's node.

AppTiers are combined in applications. By defining multiple applications, you can make your work more manageable, for example by distinguishing between a test application and a production application. You can also define applications to implement a security policy and authorize specific users to access performance information for only their application.

During the Precise installation, a default application based on the installed components is created. Using the Edit Application dialog box, you can change the default application or create a new application. You can also add, rename, or correlate applications.

The AppTier table, in the Edit application dialog box, lists all AppTiers in the application and their respective information source.The current correlation of an application is shown in the data flow diagram to the right of the AppTier table. Within Precise, you can view the data flow diagram on the StartPoint page and in the Overview tab of Insight.

To view and manage applications

- In AdminPoint, click **Setup>Applications**.

The Applications table displays information in different columns for each instance. The user can customize each column header and its value.

Table 3-11 Display Applications table displayed columns

Column name	Description
Application	Name of the application. In the Agents table, AdminPoint displays all of the applications that an instance belongs to, according to the AppTiers it is part of.

Column name	Description
Node	Name of the node.
AppTiers	Name of the AppTiers.
Technologies	Name of the technologies within the application.
SmartLink Application	If available, the name of the SmartLink application.
Product	See Customizing custom columns in the environment.
Type	See Customizing custom columns in the environment.
Location	See Customizing custom columns in the environment.
Tag	See Customizing custom columns in the environment.
Comment	See Customizing custom columns in the environment.

Managing applications

Using the applications command buttons, you can perform the following actions on one or more applications in the Display applications table:

- Click **Add** to add a new application.
- Click **Edit** to change the members of an existing application.
- Click **Remove** to delete the selected application from the Display applications table.
- Click **Customize Columns** to customize the display of columns on the table.

Adding a new application

On the applications view, you can add a new application to the Precise system.

To add a new application

1. On the applications view, click **Add**.
2. In the Add application dialog box, enter the application name and select a node from the list.
3. Click **OK** to apply your settings, or **Cancel** to close and not save the settings.

Editing an existing application

Existing Environments can be edited via the Edit Environment dialog box. In the Edit Environment dialog box, you can perform any of the following editing options:

- Rename a selected Environment.
- Change the order of all AppTiers that are part of the selected environment.
- Change the data flow between AppTiers.
- Mark the front-end AppTier
- Configure the SmartLink correlation for the AppTier.

> ⓘ   In the Edit Environment dialog box, the node for the selected environment is displayed.

To edit an existing Environment

- To edit an existing AppTier in the table, on the Environments view select the Environment you want to edit and click **Edit**.

To rename an existing Environment

1. To rename an AppTier, in the Edit Environment dialog box, click **Rename**.
2. In the Rename Environment dialog box, enter the new name for the Environment.
3. Click **OK** to save the new name, or **Cancel** to close and not save.

To change the AppTier order in the AppTiers table

1. In the Edit Environment dialog box, select the AppTier to be moved in the AppTiers table.
2. Click the appropriate up/down arrow to change where the selected AppTier appears in the table.
3. Click **OK** to save the changes, or **Cancel** to close and not save the change(s).

> ⓘ   Any AppTier change performed on an instance which is part of a cluster, is automatically applied to all instances of that cluster.

Changing the data flow between AppTiers

The Environment map displays the data flow between AppTiers and displays the front-end AppTier.

To delete a relation (relation arrow), click an AppTier with an existing relation and drag the plus onto another related AppTier. To mark or clear an AppTier as front end, right-click on an AppTier to mark or clear the front-end mark.

To change the data flow between AppTiers

1. On the Environment map, click an AppTier to select it.
2. To define a new data flow, drag the plus sign (from the previously selected AppTier) to another AppTier.

Marking a Front-end AppTier for an environment

It is important that you indicate which AppTier is to the front-end. On the Environment map, a Front-end AppTier is marked with an arrow. You can change the Front-end AppTier whenever needed by unmarking a current Front-end AppTier and marking another AppTier as Front-end.

To change the Front-end AppTier for an environment

1. In the Edit Environment dialog box, right-click an AppTier on the Environment map.
2. To remove a Front-end indication, right-click on the AppTier and click **UnMark as Front-end AppTier**.
3. To assign an AppTier as Front-end, right-click on the AppTier and click **Mark as Front-end AppTier**.

Configuring an AppTier's application

An AppTier's application can be configured via the Environments view. This enables related application associations in the related Precise product tab.

To configure an AppTier's application

1. On the Setup tab menu, click **Environments**.
2. In the relevant environment, click the AppTier.
3. Click **Edit** and set the application to required application

About configuring Insight SmartLink correlation

Insight SmartLink correlates end-to-end activities across multiple tiers, and provides information on the environment's performance, from the user's perspective by tracking individual transactions. To use Insight SmartLink, you need to specify the SmartLink mode (application) to be applied to the selected environment.

There are two SmartLink modes for two types of applications:

- PeopleSoft application
- Web, MS .NET and J2EE applications. (This includes any application that has some or all of these AppTiers in any combination, including any database AppTier that is invoked by the .NET or J2EE AppTiers.)

For more information on Insight SmartLink, see the *Precise Insight User's Guide*.

Configuring Insight SmartLink correlation

You can configure the Insight SmartLink correlation via the Edit Environment dialog box.

To configure Insight SmartLink correlation

1. On the Setup tab menu, click **Environments**.
2. In the Edit Environment dialog box, click **SmartLink**.
3. Select a mode: **PeopleSoft** or **Web, MS .NET and J2EE application**.
   a. PeopleSoft configuration
      i. The PeopleSoft configuration dialog contains a table with all AppTiers to correlate data from. The Web, J2EE and Tuxedo AppTiers are mandatory; the Oracle AppTier is optional.
      ii. Once configuration is finished, you must restart: Insight, Web, J2EE, Oracle FocalPoints and agents.

   > ⓘ Only one SmartLink for PeopleSoft environment can be configured per specific PeopleSoft application.

   b. Web, MS .NET and J2EE configuration
      i. In Web, MS .NET and J2EE mode, there is no need to select specific AppTiers. The SmartLink correlation engine correlates data from all Web, MS .NET and J2EE AppTiers in the environment.
      ii. Once configuration is finished, and if configuration was changed from PeopleSoft to Web, MS .NET and J2EE mode, you must restart the FocalPoints and agents.
4. Click **OK** to save, or **Cancel** to not save this configuration.

Customizing custom columns in the environment

You can customize the environment's column headers and their contents as displayed on the environment's table in both AdminPoint and StartPoint.

To customize column headers on a table

1. Click **AdminPoint>Environments** view.
2. Select one or more environments and click **Customize Columns**.
3. Click **OK** to save the customized columns setting, or **Cancel** to close the dialog box without saving.

About the Warehouse tab

Through the Warehouse tab, you can view and manage PMDB statuses and processes. See "About the PMDB" on page 68.

Configuring AdminPoint settings

This section includes the following topics:

- Changing passwords in AdminPoint
- Defining roles and users in Precise
- About configuring planned downtimes
- About configuring SLAs
- About configuring locations
- About configuring grouping settings
- Configuring hour groups
- Configuring Alerts general settings
- Configuring Alerts metric settings
- Configuring Admin Dashboard settings

Changing passwords in AdminPoint

You can change your password from the Settings menu in StartPoint or AdminPoint.

To change a password.

1. On the Settings menu, click **Change Password**.
2. In the **Current password** field, enter your current password.
3. In the **New password** field, enter your new password.
4. Confirm the new password by typing the new password again.
5. Click **OK** to save the new password, or **Cancel** to close the dialog box without saving.

Defining roles and users in Precise

The following section describes how to define the roles and users in Precise.

About roles and users in Precise

Precise provides a centralized role management capability across the entire Precise deployment, including granting specific users the ability to access a specified technology/environment/AppTier/instance and perform restricted actions on it. Furthermore, Precise administrators are able to delegate sub-sets of their permissions to other users, thus allowing them to manage their own users and permissions.

User permissions are role-based. This means that each user that is defined in Precise should have at least one role that is assigned to him. A role defines the allowed permissions on a sub-set of resources.

Permissions can be defined for the following resources:

- Technology
- Environment
- AppTier
- Instance

Each role can have resources assigned only from one of the above-named scopes. The following table describes the predefined roles for Precise.

Table 4-1 Pre-defined roles for Precise

Role	Description
Precise manager	Allowed to install all technologies, administrate all technologies, monitor and tune all technologies. Included permissions: • ADMINISTRATE.FULL_CONTROL on ALL technologies. • MONITOR.FULL_CONTROL on ALL technologies. • TUNE.FULL_CONTROL on ALL technologies. ⓘ This role cannot be modified or deleted.
Precise guest	Allowed to monitor the default environment.
<Techology> manager	Allowed to install, monitor and administer the specified technology. Included permissions: • ADMINISTRATE.FULL_CONTROL on <Technology> and operating system technologies. • MONITOR.FULL_CONTROL on ALL <Technology> and operating system technologies. • TUNE.FULL_CONTROL on <DB Technology> (if a database).
Precise Monitor	Allowed to monitor and tune all technologies. Included permissions: • MONITOR.FULL_CONTROL on ALL technologies. • TUNE.FULL_CONTROL on ALL technologies.

Delegating permissions

Any user assigned with the ADMINISTRATE.EXECUTE permission on the Precise technology can customize roles and permissions. This user can customize roles and permissions only in the context of the permissions assigned to him, for example: a user with MONITOR.VIEW permission on the environment "Production Environment" can grant the monitor view permission on the "Production Environment" environment to other users.

If a user has permissions on all resources of a specific scope, he can delegate all resources belonging to all other scopes too. For example: a user with the following roles:

• Role that includes user assigned with the ADMINISTRATE.EXECUTE permission on the Precise technology
• Role that includes all environments (The "All Environments" check box is checked for the role) can delegate this permission on all technologies, AppTiers, and instances.

To delegate a user's permission(s) to other users

1. Create a new role with the appropriate permissions
2. Assign the role to another existing user, or to a new user

Performing activities on roles

You can define, edit, clone, and delete roles. To be able to define, edit, clone, and delete roles, you must have ADMINISTRATE.EXECUTE permission on the Precise technology. See Delegating permissions.

To define, edit, clone, or delete roles

1. From AdminPoint, on the Settings menu, click **Roles**.
2. Specify the permissions for the role.

About nodes as they apply to roles and users

Roles apply on all nodes in an installation. Resource-permissions are node related; each node will have only the relevant resource-permissions assigned to it, based by its scope.

Table 4-2 Resource permissions for nodes based by scope

Scope	Description
Technology	You need to specify in the role dialog the nodes it is associated to. The role can be associated to a specific node, or to the "all" node option. If the "all" (default) node option is selected, new attached nodes are added automatically to the existing list of roles for that role.
Environment	The node association is based on the environment's node. ⓘ monitor.view permission can only be set on the environment scope.
AppTier	The node association is based on the AppTier's node.
Instance	The node association is based on the instance's node.

Defining a role

You can define a role via the Role Settings dialog box.

> ⓘ The case-sensitive role name can have a maximum length of 50 characters. The valid characters are A-Z, a-z, -, _, , space, @, and 0-9.

To define a role

1. In the Role Settings dialog box, click **Add**.
2. In the Role Settings - Add dialog box, insert a role name for the new role.
3. Select a role scope: **technology**, **instance**, **environment**, or **AppTier**.
4. Select the resources to which this role will apply.

When you select "ALL parameters for the technology (or other selected scope)," all of the resources for the selected scope are included.

1. Select the permissions for the selected resources.
2. For the technology scope, click **Change node** to select the nodes this role applies. By default, "all" is selected.
3. Click **OK** to save your settings.

Editing a role

You can edit a role via the Role Settings dialog box.

To edit a role

1. In the Role Settings dialog box, select the role you want to edit and click **Edit**.
2. Make the required changes (where needed) for role name, scope, resources, permissions and nodes.
3. Click **OK** to save your settings. The changes are immediately applied to all logged users.

Cloning a role

You can clone a role via the Role Settings dialog box.

To clone a role

1. In the Role Settings dialog box, click **Clone**.
2. Make the required changes (where needed) for role name, scope, resources, permissions and nodes.
3. Click **OK** to save your settings.

Deleting a role

You can delete a role via the Role Settings dialog box.

> ⓘ Apart from the Precise manager entity, you are able to delete all available roles.

To delete a role

1. In the Role Settings dialog box, select the role you want to edit and click **Delete**.
2. Click **Yes** to confirm, if you want to delete this role.

Performing activities on users

You can define, edit, and delete users. A user can only give or revoke permissions he is permitted to delegate. Whether you define, edit, or delete a user, you need to start in the Users dialog box, and must have ADMINISTRATE.EXECUTE permission on Precise technology.

> ⓘ If using LDAP integration, a Precise user can only manage resources and assign permissions. All other user and role-related operations will be blocked. For more information on LDAP, see the "Configuring a secured Precise system" section in the *Precise Installation Guide*.

To define, edit, or delete users

- From AdminPoint, on the Settings menu, click **Users**.

Defining a user

You can define a user via the Users dialog box.

> ⓘ The user name (not case-sensitive) can have a maximum length of 50 characters. The valid characters are A-Z, a-z, -, _, space, and 0-9.

To define a user

1. In the User Settings dialog box, click **Add**.
2. Insert a user name for the new user.
3. Insert a password for this user and confirm this password.

4. Click an arrow to add one or more roles, by moving the appropriate role(s) from the left pane to the right pane.
5. Click **OK** to save your settings.

Editing a user

You can edit a user via the User Settings dialog box.

To edit a user

1. In the User Settings dialog box, select a user from the list, and click **Edit**.
2. Insert a password and confirm it. The user can change this password after login.
3. Click an arrow to connect this user to one or more roles, by moving the appropriate role(s) from the left pane to the right pane.
4. Click an arrow to disconnect this user to one or more roles, by moving the appropriate role(s) from the right pane to the left pane.

> ⓘ "Admin" user roles cannot be changed.

5. Click **OK** to save your settings.

Deleting a user

You can delete a user via the User Settings dialog box.

> ⓘ "Admin" user roles cannot be changed.

To delete a user

1. In the User Settings dialog box, select a user from the list.
2. Click **Delete**.
3. Click **Yes** to confirm if you want to delete this user.

About role-based access

The following table describes the permissions needed to perform an operation.

The user can view the environment in StartPoint; the data which the user can view, and the operations he can perform are based on the roles given to him. See "About role management in Alerts" on page 169.

Table 4-3 Permissions needed to perform an operation

Operation	Required permission	Required resource
Install instance	ADMINISTRATE.INSTALL	The instance's technology. ⓘ To install an instance, the ADMINISTRATE.INSTALL permission is also required on each environment where the user can install the instance.

Operation	Required permission	Required resource
Edit/uninstall instance	ADMINISTRATE.INSTALL	The instance.
Set instance AppTiers	ADMINISTRATE.EXECUTE	The instance.
Set instance customized columns	ADMINISTRATE.EXECUTE	The instance.
Apply update	ADMINISTRATE.INSTALL	Precise technology on all nodes.
Change instances customized column headers	ADMINISTRATE.EXECUTE	Precise technology on all nodes.
Apply license	ADMINISTRATE.EXECUTE	Precise technology on all nodes.
Add node	ADMINISTRATE.INSTALL	Precise technology on all nodes.
Edit/remove node	ADMINISTRATE.INSTALL	Precise technology on the node.
Add environment	ADMINISTRATE.EXECUTE	Precise technology on the node the environment belongs.
Edit/remove/customize columns environment	ADMINISTRATE.EXECUTE	The environment.
Change environments customized column header	ADMINISTRATE.EXECUTE	Precise technology on all nodes.
Add AppTier	ADMINISTRATE.EXECUTE	The AppTier's environment.
Edit/remove AppTier	ADMINISTRATE.EXECUTE	The AppTier.
Manage roles and users	ADMINISTRATE.EXECUTE	Precise technology on all nodes.
Manage clusters	ADMINISTRATE.EXECUTE	The cluster's technology.
Manage downtime	ADMINISTRATE.EXECUTE	The instance.
Manage SLA availability	ADMINISTRATE.EXECUTE	The instance's technology.
Manage SLA response time or service time	ADMINISTRATE.EXECUTE	The instance's and Precise technology on the nodes the user wants to manage.
Manage grouping	ADMINISTRATE.EXECUTE	Precise technology on the node the group is defined and on the technology the group is defined.
Manage locations	ADMINISTRATE.EXECUTE	Precise technology on the node the location is defined.
View Environment in StartPoint	MONITOR.VIEW	The environment.

Operation	Required permission	Required resource
Create Support file	ADMINISTRATE.EXECUTE	Precise technology

About configuring planned downtimes

A planned downtime is a period during which an instance is considered not available, for example for maintenance. During a planned downtime, no data is collected for the specified time period so that availability calculations remain accurate. For example using Precise for SAP, no data is displayed for a planned downtime period in the Availability tab. In addition, Alerts does not report instance-related problems during this planned downtime period.

A planned downtime can be defined as a regular downtime, which occurs only once, or as recurring downtime, which returns periodically. You can define, update, and delete planned downtime periods for all instances by using the Planned Downtime dialog box. In this dialog box, scheduled planned downtimes appear in both the Instance table and in the calendar. You can use the arrow buttons above the calendar to scroll to a wider time frame for planned downtimes.

Adding a planned downtime

You can add a planned downtime via the Planned Downtime Settings dialog box.

To add a planned downtime

1. On the Settings menu, click **Downtime**.
2. From the technologies drop-down list, select the technology you want to define the downtime for.
3. In the Planned Downtime Settings dialog box, below the table, click **Add**.
4. In the Planned Downtime Settings — Add dialog box, from the Instance list, select the instance for which you want to add a planned downtime.
5. Set the time range during which the planned downtime must occur. Use the From/To dates and time settings.
6. If the downtime is recurrent, click **Recurrence**.
7. In the Planned Downtime Settings — Add Recurring Downtime dialog box, define how often you want the downtime to recur.
   - **Daily**. The interval in days at which the downtime is to recur.
   - **Weekly**. The weekday and interval in weeks at which the downtime is to recur.
   - **Monthly**. One of the following definitions:
     - On which day (numeral) of the month and at which interval in months the downtime is to recur
     - On which weekday of the month and at which interval in months the downtime is to recur
8. If the downtime period is limited, select the **End by** option and set the date.
9. Click **OK**.

Editing a planned downtime

You can edit a planned downtime via the Planned Downtime Settings dialog box.

To edit a planned downtime

1. On the Settings menu, click **Downtime**.
2. From the technologies drop-down list, select the technology you want to edit the downtime for.
3. In the Planned Downtime Settings dialog box, below the table, click **Edit**.
4. In the Planned Downtime Settings — Edit dialog box, from the Instance list, select the instance for which you want to edit a planned downtime.

5. If the downtime is recurrent, click **Recurrence**.
6. In the Planned Downtime Settings — Edit Recurring Downtime dialog box, define how often you want the downtime to recur.
   - **Daily**. The interval in days at which the downtime is to recur.
   - **Weekly**. The weekday and interval in weeks at which the downtime is to recur.
   - **Monthly**. One of the following definitions:
     - On which day (numeral) of the month and at which interval in months the downtime is to recur
     - On which weekday of the month and at which interval in months the downtime is to recur

7. If the downtime period is limited, select the **End by** option and set the date.
8. Click **OK**.

Deleting a planned downtime

You can delete a planned downtime via the Planned Downtime Settings dialog box.

To delete a planned downtime

1. On the Settings menu, click **Downtime**.
2. From the technologies drop-down list, select the technology and instance you want to delete.
3. In the Planned Downtime Settings dialog box, below the table, click **Delete**.

About configuring SLAs

Service Level Agreements (SLAs) define the performance goals of an information system. They help you identify when the service level that is associated with your information system as a whole, or with some parts of it (such as Web pages), fall below a threshold. When you discover a drop in performance, further investigation and drill down operations are required to isolate the location of the drop and to pinpoint its cause.

You can define three types of SLAs:

- **Service Time**. Defined for activities of a specific technology. Service Time SLAs refer to the net duration needed to process a request, from the time the request arrived until the time the service that is associated with the request was completed. For this SLA type, you define threshold values for activity durations. These values represent units of time and distinguish between the different levels of service.
  Service Time SLAs are available for the following technologies: .NET, J2EE, Oracle, SQL Server, Web, Tuxedo, Oracle Applications, WebSphere MQ, and other technologies that are defined as part of the 'Other' AppTier.
- **Response Time**. Defined for activities of a specific application. Response Time SLAs refer to the total duration needed to process an application request, from the time the user initiated the request until that time when the response to the request is returned to the user. For this SLA type, you define threshold values for activity durations. These values represent units of time and distinguish between the different levels of service.
  Response Time SLAs are available for the following technologies: Web and SAP.
- **Availability**. Indicate the lowest acceptable level of an instance's availability. For Availability SLAs, you define a threshold value for each instance. The value represents the minimum percentage of time an instance should be up for its availability to still be considered acceptable.
  Availability SLAs are available for the following technologies: .NET, J2EE, Oracle, SQL Server, Sybase, Web, Tuxedo, Oracle Applications, WebSphere MQ, and Oracle Applications.

Further, to help you manage your information system proactively, you can define a near-breach threshold for each Response Time and Service Time SLA. The near-breach threshold identifies situations where the SLA has not yet been breached but the service level approaches an unacceptable level. You can then take the necessary steps and precautions to assure this situation does not deteriorate further and does not become a performance problem.

The SLA Settings dialog box lets you manage SLAs. You may also add, edit, or delete Response Time and Service Time SLA definitions. Availability SLA definitions are created per instance during the installation of the instance, and they are removed when the instance is uninstalled. Therefore, you cannot add or delete Availability SLA definitions, but you can edit them any time.

The SLA Settings dialog box opens, displaying a list of all SLA definitions currently defined under the selected SLA type, the activity or activities this SLA definition refers to, the near-breach and breach threshold values, and the last time the SLA definitions was updated.

> ⓘ An SLA definition for a specific activity is assigned by best fit. For example, if a specific Human Resource transaction - such as "HR12" - should have a different SLA threshold than other transactions, you could define an additional SLA definition for it. In this case, all transactions starting with "HR" would be assigned to the first SLA definition, while "HR12" would be assigned by best fit to its specific SLA definition.

Viewing SLA settings

You can view SLA settings via the SLA Settings dialog box.

To view SLA settings

1. On the Settings menu, click **SLA**.
2. In the SLA Settings dialog box, to view the Service Time SLA table, the Response Time SLA table, or the Availability SLA table, click the relevant tab.
3. From the Technology list, select the technology for which you want to view SLAs.

Managing response time

Response time SLAs refer to the total duration needed to process an application request, from the time the user initiated the request until that time when a response to the request is returned to the user.

Adding a response time SLA definitions

You can add a response time SLA definition via the SLA Settings dialog box.

To add a response time SLA definition

1. On the Settings menu, click **SLA**.
2. In the SLA Settings dialog box, on the relevant tab, from the Technology list, select the technology for which you want to add an SLA definition and click **Add**.
3. In the SLA Settings - Add SLA Definition dialog box, perform one of the following steps:
   - **For all technologies except 'Other'**. In the SLA name text box, type the name of the new SLA definition.
   - **For a technology of type 'Other'**. From the AppTier list, select an AppTier.
   - **For SAP**. From the Activity type list, select the relevant activity type.
4. Select a value in the node list; the SLA can be defined on a single node, or on all nodes.
5. The filter button will retrieve the data from the selected node. If all nodes was selected, the operation will be performed on all nodes and the result will be a combination of all results.
   Click **Filter** to populate the Select list with items. To display a smaller, more focused set of items, type a string expression that contains the character string common to all the items that you want to add to the SLA definition, combined with wildcard characters, into the text box. For example, suppose you want to define an SLA definition for the SAP AppTier that contains only the SAP transactions that are executed by the Human Resources department. If all such SAP transactions start with the letters "HR," you can type the string HR% in the Filter text box to display only Human Resources department transactions in the Select list. If the name of the item that you want to add to the SLA definition does not appear in the Select list, type its

name into the Free text box. You can also use this text box to add a pattern to the SLA definition instead of selecting distinct transactions. To add a pattern to the SLA definition

6. Use the arrow buttons to move items to the list on the right. To delete an item from the list, select it and click **Remove**.
7. Specify time values, in seconds, for Near-breach and Breach thresholds and click **OK**. You can fine-tune the threshold up to milliseconds, for example 1.02 seconds.

Editing a response time SLA definition

You can edit a response time SLA definition via the SLA Settings dialog box.

To edit a response time SLA definition

1. On the Settings menu, click **SLA**.
2. In the SLA Settings dialog box, on the relevant tab, from the Technology list, select the technology for which you want to add an SLA definition and click **Edit**.
3. In the SLA Settings - Edit SLA Definition dialog box, perform one of the following steps:
   • **For all technologies except 'Other'**. In the SLA name text box, type the name of the new SLA definition.
   • **For a technology of type 'Other'**. From the AppTier list, select an AppTier.
   • **For SAP**. From the Activity type list, select the relevant activity type.

4. Select a value in the node list; the SLA can be defined on a single node, or on all nodes. The filter button will retrieve the data from the selected node. If all nodes was selected, the operation will be performed on all nodes and the result will be a combination of all results.
5. Click **Filter** to populate the Select list with items.
   To display a smaller, more focused set of items, type a string expression that contains the character string common to all the items that you want to add to the SLA definition, combined with wildcard characters, into the text box. For example, suppose you want to define an SLA definition for the SAP AppTier that contains only the SAP transactions that are executed by the Human Resources department. If all such SAP transactions start with the letters "HR," you can type the string HR% in the Filter text box to display only Human Resources department transactions in the Select list.
   If the name of the item that you want to add to the SLA definition does not appear in the Select list, type its name into the Free text box. You can also use this text box to add a pattern to the SLA definition instead of selecting distinct transactions.

To add a pattern to the SLA definition

1. Use the arrow buttons to move items to the list on the right. To delete an item from the list, select it and click **Remove**.
2. Specify time values, in seconds, for Near-breach and Breach thresholds and click **OK**. You can fine-tune the threshold up to milliseconds, for example 1.02 seconds.

Deleting a response time SLA definition

You can delete a response time SLA definition via the SLA Settings dialog box.

To delete a response time SLA definition

1. On the Settings menu, click **SLA**.
2. In the SLA Settings dialog box, on the relevant tab, from the Technology list, select the technology for which you want to add an SLA definition and click **Delete**.

Managing service time

Service Time SLAs refer to the net duration needed to process a request, from the time the request arrived until the time the service that is associated with the request was completed.

Adding a service time SLA definition

You can add a service time SLA definition via the SLA Settings dialog box.

To add a service time SLA definition

1. On the Settings menu, click **SLA**.
2. In the SLA Settings dialog box, on the relevant tab, from the Technology list, select the technology for which you want to add an SLA definition and click **Add**.
3. In the SLA Settings - Add SLA Definition dialog box, perform one of the following steps:
    - **For all technologies except 'Other'**. In the SLA name text box, type the name of the new SLA definition.
    - **For a technology of type 'Other'**. From the AppTier list, select an AppTier.
4. Select a value in the node list; the SLA can be defined on a single node, or on all nodes.
5. The filter button will retrieve the data from the selected node. If all nodes was selected, the operation will be performed on all nodes and the result will be a combination of all results.
    Click **Filter** to populate the Select list with items.
    To display a smaller, more focused set of items, type a string expression that contains the character string common to all the items that you want to add to the SLA definition, combined with wildcard characters, into the text box. For example, suppose you want to define an SLA definition for the SAP AppTier that contains only the SAP transactions that are executed by the Human Resources department. If all such SAP transactions start with the letters "HR," you can type the string HR% in the Filter text box to display only Human Resources department transactions in the Select list. If the name of the item that you want to add to the SLA definition does not appear in the Select list, type its name into the Free text box. You can also use this text box to add a pattern to the SLA definition instead of selecting distinct transactions.

To add a pattern to the SLA definition

1. Use the arrow buttons to move items to the list on the right. To delete an item from the list, select it and click **Remove**.
2. Specify time values, in seconds, for Near-breach and Breach thresholds and click **OK**.

You can fine-tune the threshold up to milliseconds, for example 1.02 seconds.

Editing a service time SLA definition

You can edit a service time SLA definition via the SLA Settings dialog box.

To edit a service time SLA definition

1. On the Settings menu, click **SLA**.
2. In the SLA Settings dialog box, on the relevant tab, from the Technology list, select the technology for which you want to add an SLA definition and click **Edit**.
3. In the SLA Settings - Edit SLA Definition dialog box, perform one of the following steps:
    - **For all technologies except 'Other'**. In the SLA name text box, type the name of the new SLA definition.
    - **For a technology of type 'Other'**. From the AppTier list, select an AppTier.
4. Select a value in the node list; the SLA can be defined on a single node, or on all nodes.
5. The filter button will retrieve the data from the selected node. If all nodes was selected, the operation will be performed on all nodes and the result will be a combination of all results.
    Click **Filter** to populate the Select list with items.
    To display a smaller, more focused set of items, type a string expression that contains the character string common to all the items that you want to add to the SLA definition, combined with wildcard characters, into the text box. For example, suppose you want to define an SLA definition for the SAP AppTier that contains only the SAP transactions that are executed by the Human Resources department. If all such SAP transactions start with the letters "HR," you can type the string HR% in the Filter text box to display only Human Resources department transactions in the Select list.
    If the name of the item that you want to add to the SLA definition does not appear in the Select list, type its

name into the Free text box. You can also use this text box to add a pattern to the SLA definition instead of selecting distinct transactions.

To add a pattern to the SLA definition

1. Use the arrow buttons to move items to the list on the right. To delete an item from the list, select it and click **Remove**.
2. Specify time values, in seconds, for Near-breach and Breach thresholds and click **OK**.

You can fine-tune the threshold up to milliseconds, for example 1.02 seconds.

Deleting a service time SLA definition

You can delete a service time SLA definition via the SLA Settings dialog box.

To delete a service time SLA definition

1. On the Settings menu, click **SLA**.
2. In the SLA Settings dialog box, on the relevant tab, from the Technology list, select the technology for which you want to add an SLA definition and click **Delete**.

Managing availability

Availability SLAs indicate the lowest acceptable level of an instance's availability.

Editing an availability SLA definition

You can edit an availability SLA definition via the SLA Settings dialog box.

> ⓘ   You cannot add or delete Availability SLA definitions, but you can edit them at any time.

To edit availability SLA definitions

1. From the Settings menu, click **SLAs**.
2. In the SLA Settings dialog box, click the Availability tab.
3. From the Technology list, select the technology for which you want to edit an SLA definition.
4. In the table, select an instance. Then click **Edit**.
5. In the SLA Settings - Edit SLA Definition dialog box, type the new SLA value (in percent) and click **OK**.

About configuring locations

By defining Precise locations, you can logically divide your organization into geographical or organizational groups. Defining locations this way can help you to isolate local performance problems and also gain a better understanding of system behavior patterns.

Location definitions are based on the IP addresses of client machines. You can define a location that is based on one IP address or based on a range of IP addresses. Location definitions affect Insight and Precise for Web.

The table in the Location Settings dialog box displays a list of all locations currently defined, the specific IP addresses or IP address ranges that are part of the location, and the time and date of the last update. You can only define locations for newly loaded data, not for previously loaded data.

An activity with a specific IP address is assigned to the best-fit location. For example: Location A has an IP address range from 10.1.0.0 to 10.1.0.255 while location B is defined for IP addresses 10.1.0.77 and 10.1.0.79. In this case, an activity with the IP address 10.1.0.79 is assigned to location B, not A.

Adding a location

You can add a location via the Location Settings dialog box.

> ⓘ When you insert IP ranges, verify that you do not create an invalid overlap. For example:
> 211.2.2.2-223.0.0.0 and 222.2.2.2-224.0.0.0 overlap and create an invalid overlap. 211.2.2.2-223.0.0.0 and
> 223.0.0.1-224.0.0.0 or 211.2.3.3-222.1.2.9 are valid overlaps, if overlapping is checked as the selected node.
> If all nodes was selected, the overlapping check will be performed on all nodes.

To add a location

1. On the Settings menu, click **Location**.
2. In the Location Settings dialog box, click **Add**.
3. Select a value from the node list; the location can be defined on a single node or on all nodes.
4. The filter button will retrieve the data from the selected node. If all nodes was selected, the operation will be performed on all nodes and the result will be a combination of all results.
5. In the Location Settings - Add dialog box, click **Filter** to populate the Select list with items.
   To display a smaller, more focused set of items in the Select list, type a string expression that contains the character string common to all the IP addresses you want to add to the SLA definition, combined with wildcard characters, into the text box.
   For example: If you define a Chicago location and all IP addresses in this location start with "10.6," you could type the string 10.6* in the Filter box to display only IP addresses that belong to this location.
6. To specify a range of IP addresses for the location, type the first and last address into the From...To text boxes.
7. Use the arrow buttons to move items to the list on the right.
8. To delete an item from the right list, select it and click **Remove**.
9. Click **OK**.

Editing a location

You can edit a location via the Location Settings dialog box.

> ⓘ When you insert IP ranges, verify that you do not create an invalid overlap. For example:
> 211.2.2.2-223.0.0.0 and 222.2.2.2-224.0.0.0 overlap and create an invalid overlap. 211.2.2.2-223.0.0.0 and
> 223.0.0.1-224.0.0.0 or 211.2.3.3-222.1.2.9 are valid overlaps, if overlapping is checked as the selected node.
> If all nodes was selected, the overlapping check will be performed on all nodes.

To edit a location

1. On the Settings menu, click **Location**.
2. In the Location Settings dialog box, click **Edit**.
3. Select a value from the node list; the location can be defined on a single node or on all nodes.
4. The filter button will retrieve the data from the selected node. If all nodes was selected, the operation will be performed on all nodes and the result will be a combination of all results.
5. In the Location Settings - Edit dialog box, click **Filter** to populate the Select list with items.
   To display a smaller, more focused set of items in the Select list, type a string expression that contains the character string common to all the IP addresses you want to edit to the SLA definition, combined with wildcard characters, into the text box. For example: If you define a Chicago location and all IP addresses in this location start with "10.6," you could type the string 10.6* in the Filter box to display only IP addresses that belong to this location.
6. To specify a range of IP addresses for the location, type the first and last address into the From...To text boxes.

7. Use the arrow buttons to move items to the list on the right.
8. To delete an item from the right list, select it and click **Remove**.
9. Click **OK**.

Deleting a location

You can delete a location via the Location Setting dialog box.

To delete a location

1. On the Settings menu, click **Location**.
2. In the Location Settings dialog box, select the location you want to remove and click **Delete**.

About configuring grouping settings

Grouping settings are relevant for displaying information in Insight. In Insight, the graphs in the AppTier area compare performance attributes of various entities for each AppTier. This can be useful in pinpointing problem areas. In cases where the information in the graphs is too granular, you can create a higher-level view of the graph by grouping entities together.

For example, examining a graph that shows SQL Server Login Names reveals that many users use the system. It is difficult to understand from the graph who actually uses the database for normal application purposes and who uses it for administrative or other purposes. It is even more difficult to understand the geographical or organizational distribution of system users.

An effective way to obtain such information is to group entities (Login Names in this example). By grouping login names into designated groups, you can obtain a higher-level understanding of system usage and performance.

Viewing existing groups

You can view existing groups via the Grouping Settings dialog box.

To view existing groups

1. On the Settings menu, click **Grouping**.
2. In the Grouping Settings dialog box, from the Technology list, select the technology for which you want to view existing groups. The values in the Group list change according to the selected technology.
3. From the Group list, select the group whose definition you want to view. The group table populates with information on the selected group.

Adding a group

You can add a group via the Grouping Settings dialog box.

To add a group

1. On the Settings menu, click **Grouping**.
2. In the Grouping Settings dialog box, select a technology and group.
3. Click **Add**.
4. In the Grouping Settings — Add dialog box, in the Group text box, type the name for the group you want to create.
5. Select a value from the node list; the grouping can be defined on a single node or on all nodes.
6. The filter button will retrieve the data from the selected node. If all nodes was selected, the operation will be performed on all nodes and the result will be a combination of all results.
7. Click **Filter** to populate the Select list with items.
   To display a smaller, more focused set of items in the Select list, type a string expression that contains the

character string common to all the values you want to add to the group, combined with wildcard characters, into the text box.

For example: If you define a group of login names and you want to view a list of all the login names that start with "U," you can type the string U% (or %) in the Filter box to display only login names that start with "U." If a specific value does not appear in the list, use the Free text box to enter the value.

8. Use the arrow buttons to move items to the list on the right.
9. To delete an item from the right list, select it and click **Remove**.
10. Click **OK**. A new group is added to the group list.

Editing a group

You can edit a group via the Group Settings dialog box.

To edit a group

1. On the Settings menu, click **Grouping**.
2. In the Grouping Settings dialog box, select a technology and group.
3. Click **Edit**.
4. In the Grouping Settings — Edit dialog box, in the Group text box, edit those fields for the group that need to be changed.
5. The filter button will retrieve the data from the selected node. If all nodes was selected, the operation will be performed on all nodes and the result will be a combination of all results.
6. Click **Filter** to populate the Select list with items.
   To display a smaller, more focused set of items in the Select list, type a string expression that contains the character string common to all the values you want to add to the group, combined with wildcard characters, into the text box.
   For example: If you define a group of login names and you want to view a list of all the login names that start with "U," you can type the string U% (or %) in the Filter box to display only login names that start with "U." If a specific value does not appear in the list, use the Free text box to enter the value.
7. Use the arrow buttons to move items to the list on the right.
8. To delete an item from the right list, select it and click **Remove**.
9. Click **OK**. An existing group is edited for the group list.

Deleting a group

You can delete a group via the Grouping Settings dialog box.

To delete a group

1. On the Settings menu, click **Grouping**.
2. In the Grouping Settings dialog box, select a technology and group to remove.
3. Click **Delete**.

Configuring hour groups

PMDB data is summarized into hourly-based time units. In large environments with a high volume of transactions, the PMDB can use a lot of disk space. To reduce disk space consumption, Precise automatically aggregates hourly data into daily, weekly, and monthly data. Aggregation saves space, but it eliminates the raw details of hourly performance data. To specify the hours for which you want data maintained, use the Hour Group option.

The Hour Group option divides the week into hour groups. The default groups are day, morning, night, and weekend. During the installation process, you can create your own hour groups or change the defaults to whatever is appropriate for your environment. For example, you could define a peak hour every day between 10 AM and 11 AM.

When you have declared the hour groups you want, you can further define which performance data is collected within each hour group.

> ⓘ Changing the hour group settings does not apply to information already stored in the PMDB. The new information will only be used for calculations that are related to future information.

To configure hour groups

1. On the Configuration bar, click **Hour Groups**.
2. In the Hour Group Settings dialog box, do one of the following tasks:
    a. Disable the use of hour groups by clearing the **Use hour group definition for calculations** check box. Disabling hour groups affects the baseline calculation. Old hour groups already stored in the PMDB remain enabled.
    b. Mark an hour or a sequential set of hours with the mouse. The Hour Group Settings - Edit dialog box opens with the day and time duration set. Select the preferred hour group from your organization's hour groups as listed in the table. Each group appears in a different color.
    c. Click **Edit**. The Hour Group Settings - Edit dialog box opens with no day or time duration set. Specify the day of the week and the time duration you want and select the preferred hour group from your organization's hour groups, as listed in the table. Each group appears in a different color.
3. To change the name of a group, click **Rename** and type the new name in the relevant table cell. For example, to change "Customized 4" to "Peak," select the **Customized 4** row, click **Rename**, and type Peak in the name text box on the table.
4. Click **OK**.

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. For most of the metrics, Alerts enables you to launch the relevant Precise product without returning to the StartPoint screen.

> ⓘ In order to enable the Alerts settings screen, the user must have ADMIN.VIEW role permissions.

To configure Alerts general settings

1. On the Settings menu, click **Alerts General Settings**.
2. In the Alerts General Settings dialog box, select the appropriate tab.

Setting alerts defaults on the General tab

The alert defaults affect the view of the various tables and graphs in Alerts. You can set the alert defaults to fine-tune the view according to your preferences.

The General tab includes the following parameters:

- **Display alert history over the last [hours]**. This parameter affects all history table columns (in the Metric tab, Instances tab, and AppTiers table), maximum, minimum, and average counters (in the Current tab). You can set the history period up to 168 hours (week).
  After modifying this parameter you must restart the Alerts FocalPoint to apply the new setting.
- **Display only top *n* metrics or events**. Sets the maximum number of Metrics or Events to display in the Metric viewer, Event viewer, and the Main Area of the Metric tab. Lowering the value improves the User Interface response time.

To set alerts defaults on the General tab

1. Specify the display parameters.
2. Click **OK** to save the parameters.

Setting an Email server for actions on the Email tab

The Email tab allows you to set an external email server, through which Alerts can send email actions to a specified address. The email actions are automatically triggered by alerts, after configuring email actions for a metric.

The Email tab includes the following parameters:

- **Integrate Alerts with an email server**. Select this option box to enable Alerts to send email actions.
- **Email server name or IP address**. Set the mail server name or IP address. The mail server must be recognized by the network of the Precise installation.
- **Alerts email messages should be sent by**. Set the address you want to appear in the sent by field in email actions.

To set an Email server for actions on the Email tab

1. From the Email tab, check mark **Integrate Alerts with an Email server**.
2. Specify the details regarding the Email server.

Setting an SNMP server for actions on the SNMP tab

The SNMP tab lets you set the details of an external SNMP manager that receives the metric status from Alerts through SNMP traps, and to configure a port through which you can perform SNMP Get commands. See "About setting Alerts SNMP connectivity" on page 176.

The SNMP tab includes the following parameters:

- **Integrate Alerts with SNMP console**. Select this option box to enable SNMP functionality.
- **Server name or IP address**. Set the server name or IP address of the SNMP manager.
- **SNMP Trap Port**. Set the SNMP Trap port of the SNMP manager.
- **SNMP Trap version**. Set the SNMP Trap version of the SNMP manager.
- **SNMP Port**. Set the SNMP port of the Alerts FocalPoint server that listens to the SNMP requests.
- **SNMP Version**. Set the SNMP version of the Alerts FocalPoint server that listens to the SNMP requests.

To set an SNMP server for actions on the SNMP tab

1. From the SNMP tab, check mark **Integrate Alerts with SNMP console**.
2. Specify the details of the SNMP console for receiving Alerts SNMP traps.

Setting a MOM server for actions on the MOM tab

The MOM tab lets you set the details of an external MOM server that receives the metric status from Alerts. Specify the details of the MOM application server that will receive the alerts from Precise Alerts. See "About Alerts MOM connectivity" on page 182.

The MOM tab includes the following parameters:

- **Integrate Alerts with Microsoft Operations Manager**. Select this option box to enable MOM functionality.
- **Server name or IP address**. Insert the server name or IP address of the MOM server.
- **User name**. Insert the user name of the administrator on the MOM server.
- **Password**. Insert the password.
- **Domain**. Insert the domain.

To set a MOM server for actions on the MOM tab

1. From the MOM tab, check mark **Integrate Alerts with Microsoft Operation Manager**.

2. Specify the details of the MOM application server that will receive the alerts from Precise Alerts.

Editing instance settings on the Instances tab

The Instances tab allows you to modify the association properties of all instances within a selected AppTier. You can associate or disassociate instances with alerts, enable or disable SNMP actions for each instance, and add a default email address for each server.

You can display instances within a specific AppTier and their general settings, by selecting it from the AppTier list box. The following table describes the properties displayed for each instance.

Table 4-4 Instance association table

Properties	Description
Instance	Instance name.
Server	Indicates the server name where the instance is running.
Alerts Reported	Indicates whether alerts are reported for the instance or not.
SNMP	Indicates whether SNMP actions are enabled for the instance or not. When enabled, all the metrics that sample the instance are automatically set with SNMP actions. When disabled, all metrics that sample the instance are set with no SNMP actions.
Default email	Displays the default email address of the recipient. This email address is used as a default in the email definition in the Actions tab. See About metric properties for Action settings.

To edit instance settings

1. From the Instances tab, select the instance that you want to edit and then click **Edit**.
2. In the Instance Settings dialog box, set the Instance properties and click **OK**.

In case of a busy system, to improve the system performance, you can check the Do not report alerts for this instance option box of instances that are not important to sample.

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.

> ⓘ In order to enable the Alerts Metrics settings screen, the user must have ADMIN.VIEW or ADMIN.UPDATE role permissions.

To configure Alerts metric settings

1. On the Settings menu, click **Alerts Metric Settings**.
2. In the Alerts Metric settings dialog box, select the appropriate tab.

Setting alerts metrics on the Settings tab

The Settings dialog box allows you to edit the properties of each metric that is available in your Precise environment, and the Cross-AppTiers metrics (FocalPoints, Agents, Processes, and Licenses) available in the Precise-generated 'i3 Status' environment. You can also add customized metrics to an AppTier (excluding the Cross-AppTiers), or delete customized metrics.

To edit a metric setting for one or all instances, from an AppTier (or Cross-AppTier) of the same technology

1. From the AppTier list box, select the required AppTier (or Cross-AppTiers). The table in the Settings tab displays all the metrics and their metric set.
2. From the Metrics table, select the required metric.
3. Click **Edit**.

If you click **Edit** according to the instructions above, you next need to edit the metric properties. See Editing metric properties.

To add a customized metric to an AppTier of the same technology

1. From the AppTier list box, select the required AppTier. The table in the Settings tab displays all the metrics and their metric set of the selected AppTier.
2. Click **Add**.

If you click Add according to the instructions above, you next need to create a customized metric. See Creating customized metrics.

Creating 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 support customized metrics that use scripts in Precise servers, Alerts InformPoint must be installed. If InformPoint is not automatically installed on the specific server where you want to run customized metrics, you can install InformPoint on that particular server manually. Use the Agent Installer's Servers screen to install the InformPoint. For more information, see the *Precise Installation Guide*.

See "About creating customized metrics" on page 184.

To create a customized metric

1. On the Precise bar, click **Settings>Alerts Metric Settings**.
2. Select the Settings tab. From the AppTier list box, select the AppTier for which you want to create the customized metric (Cross-AppTiers cannot include customized metrics).
3. Click **Add**.
4. On the Metric Properties dialog box, in the Metric name text box, type a name for the new metric.
5. From Sample this metric by running options, select either stored procedure or executable file.

6. In the Executable/Stored procedure textbooks, type the command line that runs the required stored procedure or executable file.
7. In the Metric's response type option, select the metric type, either single value or item list (multiple values).
8. If necessary, select **Alert** when value drops below the designated thresholds and click **Save**.
9. Select the Description tab. In the Description box, type a description of the new metric and click **Save**.

> (i) To move from one tab to the next you must first save the changes made in the previous tab.

10. Edit the settings of the customized metric in the remaining tabs (Scheduling, Thresholds, Actions, and Customize) as required. Click **Save** after editing each tab's settings.
11. Click **Close**. The new customized metric is created.

To delete a customized metric from an AppTier

1. On the Precise bar, click **Settings>Alerts Metric Settings**. The Settings dialog box opens.
2. Select the Settings tab.
3. On the Settings tab, from the AppTier list box, select the AppTier for which you want to delete the customized metric.
4. From the Metrics list, select the metric you want to delete (only metrics of the Customized set can be deleted).
5. Click **Delete**.

Enabling and disabling metrics on the Activities tab

In the Activities tab, you have quick customization options to enable or disable metrics. These operations can also be performed in the settings tab of this dialog, except the focus in the settings tab is on only enabling or disabling metrics.

To disable all metrics for all instances of all environments

1. From the AppTier list box, select the required AppTier (or Cross-AppTiers). The table in the Activities tab displays all the enabled metrics and their metric set.
2. Click **Disable all metrics**.
3. Click **Save** to save the settings.

To enable an Availability metric for all instances of all environments

1. From the AppTier list box, select the required AppTier (or Cross-AppTiers). The table in the Activities tab displays all the enabled Availability metrics and their metric set.
2. Click **Enable Availability metrics**.
3. Click **Save** to save the settings.

To enable/disable metrics from the various sets for a specific instance

1. Select the Environment, AppTier, and instance.
2. Check any metric to enable it, or uncheck any metric to disable it.
3. Click **Save** to save the settings.

Copying metric properties on the Copy Metric Settings tab

Metric settings you set on the Activities tab, or in the settings tab, can be copied to other instances using the Copy Metric Settings tab. To copy the settings to other instances, follow the steps as given on the Copy Metric Settings tab.

To copy metric properties on the Copy Metric Settings tab:

1. Under Source Definitions, select the Environment, AppTier, and Instance you want to copy.
2. On the table, check mark the metrics you want to copy.

3. Check mark the settings group you want to copy.
4. Under Destination Instances, click **Populate destination Instances**.
5. On the table, check mark the destination instance(s).
6. Click **Execute** to copy the selected metric settings.

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.

Alerts issues alerts according to the metric properties, which must be adjusted to your individual environment and organization preferences. The metric definitions must be accurate and adequate. Sampling frequencies and periods require careful considerations. Thresholds need to be set in accordance with the performance level you want to meet.

In addition, alerts must reach the relevant personnel, or in severe cases, management representatives immediately and regardless of their location. Subsequently, the threshold-exceeding values must be examined.

Alerts enables you to adjust the metric definitions through the Metric Properties dialog box.

If you had previously selected Edit or Add for alerts metric settings on the Settings tab, you next need to edit the metric properties.

The Metric Properties dialog box includes the following tabs:

- Description
- Scheduling
- Threshold
- Actions

To edit metric properties for one or all instances, from an AppTier (or Cross-AppTier) of the same technology

1. In the Settings tab, click **Edit** or **Add**. See Setting alerts metrics on the Settings tab.
2. Click the appropriate settings tab (Scheduling, Threshold, or Action)

About setting metric sampling properties on the Scheduling tab

Alerts copies the scheduling settings for the source instance.

To set the metric sample properties

1. From the Metric Properties dialog box, select the Scheduling tab.
2. To set the sampling rate, in the Sample metric every <...> boxes, set the time (day, hours, and minutes) to the frequency, in which you want Alerts to sample the metric. If you do not want to sample this metric, select the Disable metric sampling radio button.

   ⓘ Time slice metrics' sampling parameters are usually disabled for editing.

3. To modify the sampling base, in Start sampling at <...> boxes, set the time (day, hours, and minutes) in which Alerts starts to sample the metric (the day parameter is available in case the sampling rate is a week and above). The default is: Sunday, 00:00 AM.

   ⓘ These sampling parameters are useful for metrics whose sampling rate is once a day or more. For example, if sampling every 24 hours (1 day), the metric will be sampled every midnight. For sampling it at 3:00 am, change the time parameters to 3:00.

4. In the Analyze metric over the last <...> boxes, set the sampling period for which you want to analyze the metric. The Sampling period is the time frame for retrieving statistical data from the monitored product. The sampling period is used only in metrics that return statistics for a period of time.
5. To sample the metric during downtime period, check **Sample this metric even during downtime**.
6. If the metric monitors an important performance aspect, check **This is a key metric**. Critical alerts related to a key metric are indicated by an exclamation point.

Defining thresholds on the Thresholds tab

Alerts defines "copy threshold settings" from the source instance.

The Thresholds tab allows you to define your performance requirements, and to ignore or consider specific items or conditions. Alerts will then be issued logically and according to your specific configuration.

Because some alerts are calculated based upon total instance running time, a false alert may sometimes be issued. For example, assume the Top Programs sub-metric of the Top Activities metric for SQL Server AppTier is assigned a 10% Near-Critical threshold and a 20% Critical threshold. If a program runs only 10 seconds during the time slice and there are no other programs running on the instance, it will issue a false alert because it exceeds the defined threshold (10 seconds out of a total running time of 10 seconds is 100% of instance running time). This occurs though the program ran for only 1.1% of the time slice total time [10 seconds/(60 seconds x 15 min)]. This false alerts issue is resolved by using a Minimum value setting defined on the Thresholds tab of the Metric Properties dialog box.

The Minimum value setting is a minimum value in seconds of MS-SQL time below which an alert will not be issued. Suppose you do not want to issue an alert for the Top Programs sub-metric if it does not reach 15% of the MS-SQL time. You would then set a Minimum value of 135 seconds (15 min time slice x 60 seconds/min x 15%). No alarm will be issued until the defined minimum value is exceeded.

> ⓘ   Minimum value is only relevant for list metrics whose unit value is given as a percentage.

Alerts comes with a default set of thresholds defined to suit general needs. You should tune these thresholds, as required, in relation to the setup and definitions of your environment.

There are two types of metrics:

- **Single value metrics**. These are metrics that collect only a single value. An alert is issued when its value exceeds the predefined threshold (for example the Availability metric). Parent Single Value metrics include sub-metrics (child metrics). Each child metric has its own thresholds and may be enabled or disabled individually (for example the General Behavior metric).
- **List metrics**. These are metrics that collect a dynamic list of items identifying each item by its description and value (for example the Locked sessions metric). Parent List metrics include sub-metrics (child metrics). Each child metric has its own thresholds and may be enabled or disabled individually (for example the Top Activities metric).

To edit a Sub-Metrics threshold

1. From the Metric Properties dialog box, select the Thresholds tab.
2. Select the sub-metric you want to edit.
3. Click **Edit**.
4. In the Metric Properties - Edit thresholds dialog, enter the required value in the Critical Threshold and Near-critical Threshold text boxes.
5. To consider only specific items when sampling data, include them in the Include list text box.
6. To ignore specific items when sampling data, include them in the Exclude list text box.
   When entering list items, verify that they are separated by a semicolon. Use the percent sign (%) as a wildcard. For example, Alerts%; %Alerts; %Alerts%.

7. To disable a sub-metric, clear the check mark in the Enable Sub-Metric checkbox. If a List metric type, a disabled sub-metric will not be sampled. If a Single value metric type, a disabled sub-metric will be sampled (retrieve a value), however, this will not generate an alert (the sub-metric will always be green).
8. Click **OK**. Repeat steps 2 through 7 for each sub-metric to be defined.
9. In the Minimum value <…> text box at the bottom of the Threshold tab dialog, enter the minimum value necessary for Alerts to consider the item.
10. 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**.

About metric properties for Action settings

Alerts copies the action settings from the source instance. Alerts provides the following action types when an alert is raised:

- Email
- Message Box
- Program
- SNMP
- MOM

See About metric properties for Action settings, "About setting Alerts SNMP connectivity" on page 176, and "About Alerts MOM connectivity" on page 182.

Configuring Admin Dashboard settings

The Admin Dashboard settings enable you to set additional report triggers and the Refresh Rate of the data on the screen.

The following options can be selected:

- Report also when one or more instances are not loading data into the PMDB.
- Report also when the status of one or more agent is Stopped.

The default Refresh Rate setting is 15 minutes and this amount can be changed.

Administering the PMDB

This section includes the following topics:

- About the PMDB
- About viewing and administering performance processes
- About viewing the status of the PMDB

About the PMDB

The PMDB helps you identify resource consumption patterns and predict future resource consumption and response times. The following methods perform the identification and predictions:

- Track historical resource consumption trends so you can understand and predict long-term performance behavior.
- Perform period-to-period comparisons so you can analyze performance improvements or performance degradation over time.
- Track load patterns, entity changes, entity statistics, and component parameter changes so you can understand their effects on performance.
- Track data growth and data distribution changes so you can optimize data storage management.
- Proactively detect performance bottlenecks before they turn into problems and issue alerts because performance degrades from established baselines.

By using the PMDB to store this information, you can better manage your applications, make knowledgeable decisions about application changes and hardware upgrades, and improve plans for the future.

The PMDB uses the following methods to store and manage the PMDB data:

- Schedule and run batch processes that load historical data from each instance in each AppTier.
- Manage the summary procedure.
- Provide a common interface for requests coming from the Precise user interfaces.
- Manage and maintain the PMDB by filtering unnecessary data, saving baseline data, summarizing data, and purging old data.

You can control all PMDB settings from AdminPoint, in the following way:

- Edit your Warehouse password from the Warehouse tab. See Editing your Warehouse password.
- Run, schedule, and change the parameter default values of the PMDB processes from the Processes view, under the Warehouse menu. See Viewing the PMDB processes.
- Examine the status of instances and the PMDB processes and check in which PMDB tables instance data is stored from the Status view, under the Warehouse menu. See About viewing the status of the PMDB.
- Manage hour groups from the Hour Group Settings dialog box. See Configuring hour groups, About the Dashboard tab, and About the Applications view.

Editing your Warehouse password

> ⓘ If your PMDB is SQL-based, see the instructions on how to change the database password manually in the Changing the PMDB SQL Server Collector Password technical note that can be found on the Precise Customer Portal.

To change your Warehouse password

1. In the Warehouse Processes tab in AdminPoint, click **Edit Warehouse Password**. The Edit Warehouse Password dialog box appears.

2. Select whether to update the Precise registry only (set as the default option), or to update both the Precise registry and the warehouse database.
3. As indicated, enter your current password (when enabled), your new password, and confirm your new password.
4. Click **OK**. A confirmation window appears.

> ⓘ After editing your Warehouse password, there is an automatic restart to FocalPoints.

About viewing and administering performance processes

The Processes view shows information about batch processes performance on the Warehouse work station. The Processes table in the Main area includes the following columns:

Column	Description
Node	The specific Precise framework installation. > ⓘ A Precise system can have one or more framework deployments.
Technology	The name of the technology that the process gathers data for. Processes that maintain the PMDB itself are listed, but the technology column remains empty.
Process	The process name is a logical name that describes the process function. For example, "Calculate Baselines." Two types of processes gather PMDB data. One process type includes processes that are part of the PMDB FocalPoint and run on the PMDB server, such as the Summarize Data process. The other process type includes processes that are part of another product, such as the Schema Changes process in a Microsoft SQL Server instance that Precise for SQL Server monitors. These processes run on the product's server and on the PMDB FocalPoint, which is in charge of scheduling them. See Batch processes in Precise.
Product	The name of the product the process runs for.
Last Run	The last time the process that is named in this row ran.
Next Scheduled Run	Time when the process that is named in this row is scheduled to run again.
Status	An icon indicating the current status of the process that is named in this row.

The current status of a process can be any of the following statuses:

Icon	Description
	The process is running.
	The process completed successfully.
	The process failed during the run.

Under the Processes view (on the Warehouse workstation), you can run a process, view the execution history of a process, and configure the scheduling and the parameters of a process.

Manually running a process

All processes are scheduled to automatically run at the set interval, but you may also manually run a process whenever required. See Process parameters.

To run a process manually

- In the Process table, select a process and click **Run**.

The process starts and the page refreshes to show the updated status icon.

Viewing the run history of a process

You can examine additional information about the past executions of a process, for example to find out which tables were affected during a specific execution or which component has failed during a run.

To view the run history

1. In the Process table, select a process and click **Run History**.
2. In the Run History dialog box, in the top table, select the execution that you want to focus on, such as the last execution that failed (indicated by the status icon). The bottom table populates with detailed information on the performance processes tables that the execution affected.
3. Investigate all relevant data and click **Close**. For example, if an error occurred, focus on the details in the Error Message column.

Process parameters

Specific parameters define the execution behavior of performance processes. These parameters function as a filter in that they determine exactly which information is collected and saved in the Warehouse workstation.

You can manually configure these parameters according to the requirements of your environment. You can edit the default values or override them by defining instance-specific values. If a process does not include any configurable parameters, the Parameters button is disabled. See About configurable process parameters.

The following table lists all the processes that you can configure per monitored instance.

Table 5-1 Processes configurable per instance

Technology	Process
DB2	Explain Statement

Technology	Process
DB2	Explain New Statements
DB2	Load Data
DB2	Purge Internal Data
J2EE	Load Data
Oracle	Load Data
Oracle	Purge Data
Oracle	Explain Statements
Oracle (SmarTune)	Changes
SQL Server	Load Data You can restrict the number of sessions that performance data is collected for by increasing the In MS-SQL Time value. However, this value applies to the collapsed statement, including all of its children. For example, if you set the In MS-SQL Time value to 2 and the collapsed statement includes five children that spent 1 second each in MS-SQL, this amounts to 5 seconds overall and causes Precise for SQL Server to collect the performance data for the collapsed statement, even though each child spent less than the defined 2 seconds in MS-SQL.
SQL Server	Explain Statements
SQL Server	Explain New Statements
SQL Server	Collect Objects Performance
SQL Server	Collect Operational Statistics
SQL Server	Collect Space Usage
SQL Server	Collect Schema Changes
SQL Server	Purge Internal Data
SQL Server	Perform SmarTune Analysis
Sybase	Explain Statements
Sybase	Collect Space Utilization
Sybase	Explain New Statements

Technology	Process
Sybase	Load Data
SAP	Load Data
Microsoft .NET	Load Data
Microsoft .NET (SmarTune)	Perform SmarTune Analysis
Operating System	Load Data
Other	Load Data
Tuxedo	Load Data
WebSphere MQ	Load Data
PMDB	Purge Data
PMDB	Summarize Data
SmartLink	Load Data
Web	Load Data

The following table lists the processes that you cannot configure.

Table 5-2 Processes that cannot be configured

Technology	Process
DB2	Calculate Objects Usage
Oracle	Collect Instance Definitions
Oracle	Collect Instance Statistics
Oracle	Collect Schema Changes
Oracle Storage	Load Data
Oracle	Perform SmarTune Analysis on DB Behavior
SQL Server	Collect Instance Statistics
SQL Server	Collect DB File Statistics
SQL Server	Collect Job Statistics
SQL Server (Storage)	Load Data

Technology	Process
Sybase	Collect Instance Statistics
Sybase	Purge Internal Data
Sybase Replication Server	Load Data
SAP	Organizational Mapping
PMDB	Data Maintenance (Daily)
.NET	Load Data
	Summarize Current Data

To configure process parameters

1. In the Main area table, select the process whose parameters you want to configure.
2. Click **Parameters**.
3. In the Parameters dialog box, perform one of the following actions:

To edit the default values for the selected technology	Click **Edit Default Values**. In the Default Values dialog box, specify the values and click **OK**. Unless instance-specific values override these values, they apply to all monitored instances.
To edit the values for a specific instance	Select an instance in the table. Click **Edit**. In the Edit Instance dialog box, clear the Use Defaults check box. Specify the values to be used for this instance. Click **OK**.

4. In the Parameters dialog box, click **OK**.

> (i) After you change the default of the Minimum In Oracle time to load parameter in the Load Data process, you must stop and then restart the Precise for Oracle FocalPoint. The next step is to restart the Collector agents.

> (i) Changing the "Load Data" value from "300" to "30" for Precise for J2EE can take up to ten minutes to take effect.

> (i) You can also configure the Precise for Web URL mapping rule for a domain in a system: "Treat pages with different titles as different pages". Select to view the URL along with the various titles that belong to it. The collected data includes page URLs and page titles. When cleared, you can only differentiate between different URLs, not between page titles. In this case, the collected data includes the page URL and the first page title for any URL in the system.

Scheduling processes

When the default scheduling settings do not match the requirements of your environment, you can manually schedule processes. Processes on the Warehouse workstation let you enable or disable each process. You can also schedule each process individually and run it at any frequency and interval - every 8 hours, hourly, daily, weekly, or by slice.

In environments with several instances, each instance creates a different amount of data and has its own frequency of activity. Therefore, it is sometimes necessary to schedule each instance according to the specific instance's behavior. Processes on the Warehouse workstation let you schedule some processes per instance.

The following table provides information on the default execution intervals of all Precise batch processes.

Table 5-3 Default intervals of Precise processes

Default Interval	Technology	Process
Weekly	PMDB	Purge Data
Weekly	PMDB	Warehouse Maintenance (Weekly)
Weekly	PMDB	Calculate Baselines
Weekly	Oracle	Collect Instance Definitions
Weekly	Oracle	Collect Schema Changes
Weekly	Oracle	Purge Data
Weekly	Oracle	Identity Changes (SmarTune)
Weekly	SQL Server	Collect Schema Changes
Weekly	SQL Server	Purge Internal Data
Weekly	Sybase	Purge Internal Data
Daily	Oracle	Explain Statements
Daily	Oracle	Collect Objects Statistics Changes
Daily	SQL Server	Explain Statements
Daily	SQL Server	Collect Space Utilization
Daily	Sybase	Explain Statements
Daily	Sybase	Collect Space Utilization
Daily	SAP	Organizational Mapping
Daily	PMDB	Warehouse Maintenance (Daily)

Default Interval	Technology	Process
Hourly	J2EE	Load Data
Hourly	Oracle Storage	Load Data
Every 8 hours	PMDB	Summarize Data
Hourly	Oracle	Collect Instance Statistics
Hourly	Oracle	Analyze Database Behavior (SmarTune)
Hourly	SQL Server	Collect DB File Statistics
Slice	Oracle	Load Data
Slice	Web	Load Data
Slice	SQL Server	Load Data
Slice	SQL Server	Collect Instance Statistics
Slice	SQL Server	Explain New Statements
Slice	Sybase	Load Data
Slice	Sybase	Collect Instance Statistics
Slice	Sybase	Explain New Statements
Slice	Sybase Replication Server	Load Data
Slice	SAP	Load Data
Slice	WebSphere MQ	Load Data
Slice	Tuxedo	Load Data
Slice	Oracle Applications	Load Data
Slice	Network	Load Data
Slice	Operating System	Load Data
Slice	Insight SmartLink	Load Data

The following table lists processes that cannot be scheduled per instance.

Table 5-4 Precise processes not scheduled per instance

Technology	Process
PMDB	Calculate Baselines
SAP	Organizational Mapping
WebSphere MQ	Load Data
Tuxedo	Load Data
Oracle Applications	Load Data
Other	Load Data

To schedule a process

1. Select a process in the Main area table and click **Scheduling**.
2. In the Scheduling dialog box, perform any of the following tasks:
   - Clear the **Run this process in the PMDB** check box to disable the process. This action removes the process from the PMDB Scheduler, but you can still run the process manually. See Manually running a process.
   - Change the time that the process is scheduled to run. Each process runs at a predefined interval by default: Every 15 minutes (slice), hourly, daily, or weekly. You cannot change this interval, but you can set the exact time that the process runs, for example every 2 hours, 15 minutes after the hour.
   - Schedule the process per instance (only available for some processes). This mechanism lets you perform the following tasks:
     - Disable a process for a specific instance by clearing the relevant check box in the first table column.
     - Set the interval of an instance within the process. For example, if you set the Run Every... column to **2**, the instance executes every 2 minutes (or other) run of the process.
3. Click **OK**.

Batch processes in Precise

The following sections provide a description of all existing Precise batch processes. The following table lists all batch processes on the Warehouse workstation.

Table 5-5 Batch processes on the Warehouse workstation

Process	Description
Summarize Data	Aggregates the loaded data into daily, weekly, and monthly summary tables.

Process	Description
Purge Data	Deletes old and unnecessary data from the statistics tables in the Warehouse workstation. ⓘ Although you can set the purge settings per instance, it is recommended to use the same settings for all instances in the same AppTier. Otherwise, a 'No Data Found' message appears when you launch from one product to another, and inconsistent information may be displayed in cross-AppTier products like Insight and Report Manager.
Warehouse Maintenance	Maintains the Warehouse workstation database. This process deletes old entries from the job history table and analyzes statistics and summary tables to update the database statistics. It also rebuilds and defrags indexes and maintains the normalization tables.
Calculate Baselines	Tracks the exceptions of currently measured counters that are compared to their historical values.
Load Data	Loads the data that is collected by a product into the database tables. This information lets you understand the behavior of your applications by examining their performance over time, identifying problematic time periods, and enabling you to drill down to the cause of the performance problem. Each product has its own Load Data process.
Organizational Mappings	A daily process that maps each organization, as it appear in the SAP system, to the physical site where its employees are located. Each organization can be mapped to more than one location. This mapping data is used to show SAP workload statistics according to locations.

The following table lists all Precise for Oracle batch processes.

Table 5-6 Precise for Oracle batch processes

Process	Description
Explain Statements	The PMDB can explain SQL statements that are loaded from the Precise for Oracle short-term history. This process explains new statements that have not yet been explained, or statements that have changed one of their dependent jobs. An object is considered changed either if its definition changes (for example when an index is created) or when its statistical data changes (for example the number of 'dirty' blocks in a table that is increased after running the Oracle Analysis command).
Collect Instance Definitions	Collects the changes in the instance definition, such as changes of the init.ora file or database parameters.
Collect Instance Statistics	Collects data on activity statistics of the Oracle server.

Process	Description
Collect Schema Changes	Collects the schema definitions and statistics from the Oracle database, compares them to the definitions that are collected in the previous execution of the process, and stores changes in definitions.
Collect Objects Statistics Changes	Collects the statistics changes for tables, indexes, and columns (identified based on DBMS_STATS execution). Also collects space utilization information for tablespaces.
Identify Changes (SmarTune)	Analyzes the changes to schema or instance settings to identify changes that affected the resource consumption or the response time (or both), either on the application level or even on the instance level.
Analyze Database Behavior (SmarTune)	Analyzes the instance behavior to detect places where your instance spends more time than expected.
Purge Data	Deletes old and unnecessary data that is related to instance definitions and schema changes.

The following table lists all Precise for SQL Server batch processes.

Table 5-7 Precise for SQL Server batch processes

Process	Description
Explain statements	Generates the execution plans for the statements that are collected by the Precise for SQL Server Collector agent and saves the execution plan information in the Warehouse workstation to enable you to tune these statements. This way, you can understand the execution plan that SQL Server chose for the statement, identify problematic operations in the execution plan, and analyze the objects that is referenced in this statement.
Collect Instance Statistics	Loads the Instance Statistics counters that are sampled by the Precise for SQL Server Collector agent into the Warehouse workstation. These performance counters enable you to analyze various aspects of the performance of the SQL Server instance and the overall system (such as CPU, paging, I/O, Network) over time.
Collect DB File Statistics	Samples SQL Server and collects I/O statistics for database files and saves the information that is sampled in the Warehouse workstation. This information lets you analyze the I/O activity of the database files over time and discover which files show heavy I/O activity so that you can make a sound judgment as to how these files should be dispersed amongst the various disks.

Process	Description
Collect Space Utilization	Samples SQL Server to collect information on the space that is allocated and used by the tables, indexes, and database files and saves this information in the Warehouse workstation. This information lets you easily keep track of object growth over time and decide how to disperse database files over various disks and how to relate objects to FILEGROUPS in the specified database.
Collect Schema Changes	Identifies the instance and database configuration changes, schema changes, and job changes that are performed in the SQL Server instance and saves the changes in the Warehouse workstation. This information lets you track and examine the changes that are performed on the instance and its entities and to understand how the changes affected the application's performance.
Purge Internal Data	Purges data as part of maintenance, such as purging an Availability log file.
Explain New Statements	Similar to Explain Statements, but it is executed every 15 minutes and generates execution plans for a few of the most resource-consuming statements. This way, the execution plan information becomes available as soon as possible, and not only after the Explain Statements have been carried out (which would make the Execution plan information only available after a one-day delay).
Collect Job Statistics	Samples SQL Server job runs and saves this information in the Warehouse workstation. This information lets you examine job runs over time, identify problematic time periods, and view the general trends in the job runs.

The following table lists all Precise for Sybase batch processes.

Table 5-8 Precise for Sybase batch processes

Process	Description
Explain statements	Generates the execution plans for the statements that are collected by the Precise for Sybase Collector agent and saves the execution plan information in the PMDB to enable you to tune these statements. This way, you can understand the execution plan that Sybase chose for the statement, identify problematic operations in the execution plan, and analyze the objects that are referenced in this statement.
Collect Instance Statistics	Loads the Instance Statistics counters that are sampled by Sybase Adaptive Server Enterprise (ASE) into the PMDB. These performance counters enable you to analyze various aspects of the performance of the Sybase instance (such as CPU, paging, I/O, Network) over time.

Process	Description
Collect Space Utilization	Samples Sybase to collect information on the space that is allocated and used by the tables, indexes, segments, and database devices and saves this information in the PMDB. This information lets you easily keep track of object growth over time and decide how to disperse database files over various disks and how to relate objects to devices in the specified database.
Purge Internal Data	Purges internal data as part of maintenance.
Explain New Statements	Similar to Explain Statements, but it is executed every 15 minutes and generates execution plans for a few of the most resource-consuming statements. This way, the execution plan information becomes available as soon as possible, and not only after the Explain Statements have been carried out (which would make the Execution plan information only available after a one-day delay).
Sybase Replication Server: Load Data	Loads the outbound processing Statistics counters that are sampled by the Sybase Replication Server Collector agent into the PMDB. These counters enable you to analyze the outbound processing phase of the replication system, and locate the specific component(s) which may be causing the latency in the replication system.

The following table lists Precise SmartLink processes

Table 5-9 Precise SmartLink processes

Process	Description
Load Data	Loads the data that is collected by a product into the database tables. This information lets you understand the behavior of your applications by examining their performance over time, identifying problematic time periods, and enabling you to drill down to the cause of the performance problem. ⓘ The Load Performance parameter for Precise SmartLink does not have a "Technology" value in the Display Warehouse Processes Information table in **AdminPoint>Warehouse**. To change the Load Data slice size, see Modifying the Precise Load Data slice size.

Modifying the Precise Load Data slice size

Most process are specific to one technology. The Load Data process that does not have a Technology value is shared by SmartLink and J2EE. All changes to this value affect the slice size not only in SmartLink tabs, but also in all J2EE instances in that environment.

To modify the Precise SmartLink/J2EE Load Data Performance slice size

1. Go to **AdminPoint>Warehouse>Processes**.

2. From the table, select the Load Data process with an empty Technology value. Click **Parameters**.
3. Select the relevant environment and click **Edit**.
4. Mark the **Use Defaults** check box to enable editing.
5. From the drop-down menu, select the desired slice size.
6. Click **OK>OK**.

> ⓘ Changing the **Load Data** value from "300" to "30" may take up to 10 minutes to take affect in Precise for J2EE.

About the PMDB Summarize Data process parameters

The table below describes the configurable process parameters and default values in the PMDB Summarize Data process.

> ⓘ The following set of parameters is defined for each technology, server, and instance which is installed in the Precise system.

Table 5-10 Configurable process parameters and their default values

Parameter	Default value
Summarize to daily level only top *n* rows	2000 rows
Summarize to weekly level only top *n* rows	3000 rows
Summarize to monthly level only top *n* rows	3000 rows

About viewing the status of the PMDB

Under the Warehouse menu, the view shows information on the PMDB load status of each instance in your Precise installation. The Status table in the Main area includes the following columns:

- **Server**. The name or IP address of the server on which the instance runs.
- **Technology**. The technology type of the instance.
- **Instance**. The name of the instance.
- **Unmonitored**. Indication by an icon when the instance is unmonitored.
- **First Loaded**. The date and time the first instance information was loaded into the PMDB.
- **Most Recently Loaded**. The date and time the latest instance data was loaded into the PMDB.
- **Status**. Icon indicating the status of the last load.

Using the Warehouse command buttons, you can get more detailed information on the actual PMDB tables that hold information on the specific instance. You can select any instance from the list and click Details to receive First Load and Most Recent Load information about specific tables in the PMDB.

You can also use the Warehouse Status Command selector to refresh the information that is displayed on the screen.

Viewing PMDB tables

You can examine additional information about the PMDB tables that are loaded for each instance, for example to understand their summary level.

To view the PMDB tables

1. Select an instance in the Main area table and click **Tables**.
2. In the PMDB Tables dialog box, investigate all relevant data.
3. Click **Close**.

Viewing the PMDB processes

You can examine additional information about the processes that are involved in loading information for this instance, for example to find out which tables were affected during the last execution of a process, which process failed during its last run, or when the next process is scheduled.

To view the run history

1. Select an instance in the Main area table and click **Processes**.
2. In the PMDB Processes dialog box, the top table lists all processes involved in loading information for this instance. Select the process you want to focus on, such as a process that failed during its last run (indicated by the status icon).
   The last run of this process affects the bottom table populated with detailed information on the PMDB tables. If an error occurred, focus on the details in the **Error Message** column.
3. Investigate all relevant data. Then click **Close**.

Utilizing the PMDB

This section includes the following topics:

- Prerequisites for configuring the PMDB in an SQL Server
- Prerequisites for configuring the PMDB in an Oracle Server
- How to disable the recycle bin in Oracle
- About backing up the database and creating archiving
- About maintenance operations
- About changing the size of tables and index extents

Prerequisites for configuring the PMDB in an SQL Server

Install the Precise environment before configuring the SQL Server-based PMDB. Before applying the information in this section, refer to the Installing Precise Components.

For an SQL Server-based PMDB, use an MS-SQL dedicated instance.

To avoid using the MS-SQL fully automatic configuration function, change the following parameters:

- **Maximum server memory (MB)**. Should be 80% of the server's RAM.
- **Minimum memory per query (KB)**. Should be 512 instead of 1024.

An example to configure a parameter is the following command from the Query Analyzer:

```
EXEC sp_configure 'max server memory (MB)', [new value] RECONFIGURE WITH OVERRIDE
GO
```

Maintenance operations

Check the table and index status by running the following command:

```
dbcc showcontig ({table name}) with all_indexes
```

The results should be that the "Extent Scan Fragmentation" is less than 50 and the "Logical Scan Fragmentation" is low; otherwise, it will damage the index scans.

If one of the above conditions is confirmed, perform a reorganization on the indexes with the following command:

```
dbcc dbreindex ('{databasename.tableName}')
```

> ⓘ   Be aware that the index and the table will not be available during the rebuild.

Backing up the database and creating archiving

Use the Database Maintenance Plan function to create a backup plan for the database. Verify that the SQL Server agent is automatically started at database startup.

If you choose the simple recovery mode for the database (the transaction log can be deleted after a commit or checkpoint), the recovery will not be able to work to a point in time, but only to the backup time.

If you choose a full recovery mode for the database, verify that you also backup the transaction log and shrink the database periodically, to stop the transaction log from growing infinitely.

If you do not create a backup, information will be lost when you have a database crash and you will not be able to recover data.

Additional DBA settings for an SQL Server-based PMDB

To improve performance, the DBA on site must check that the settings in the following table contain the correct values.

Table 1 Additional DBA settings for the SQL Server-based PMDB

Setting	Description	Value
Size of the datafile	The default size when the PMDB is created.	2 GB
Size of the log file	The default size when the PMDB is created.	500 MB
Auto Extent size for datafiles and logs	The size of the new segment to be allocated.	30% of initial datafile (for example, if the datafile is initially 1 GB, the auto extent size should be 300 MB).
Tempdb location	Put the `tempdb` database on a fast I/O system.	

Prerequisites for configuring the PMDB in an Oracle Server

Install the Precise environment before configuring the Oracle-based PMDB. Before applying the information in this section, refer to the Precise Installation Guide. For an Oracle-based PMDB, use an Oracle dedicated instance.

Verify that you are creating the database with the proper `NLS_CHARACTERSET`. When the monitored Oracle instances use a multiple byte character set, use UTF8. `AL32UTF8` is not supported by the PMDB.

The configuration that is recommended when the PMDB is based on an Oracle database can be found in the following paragraphs.

If you install the PMDB on an Oracle legacy database, verify that the database block size is at least 8 KB. If not, try to install the PMDB on a different Oracle instance.

The following table specifies the database block size prerequisite.

Table 2 Oracle Server Database block size prerequisite

Parameter	Value	Improvement
db_block_size	Minimum 8 KB (16 KB recommended)	This value defines the Oracle block size. The block size is set when creating the database and cannot be changed afterwards. A value of at least 8 KB ensures that the PMDB can perform smoothly. A lower value may cause serious performance problems. If you are installing on a legacy database, verify that the database block size is 8 KB or higher. If not, try to install the PMDB on a different Oracle instance. ⓘ The default block size on Windows NT is 2 KB, which is too small. To review this value, check the `db_block_size` parameter in the `init.ora` file.

Required INIT.ORA changes

To improve performance, change some parameter values and verify that the values match those listed in the appropriate table.

The parameter values can be changed on a new and on an existing Oracle instance. The following tables describe the general and Oracle version specific parameters.

Table 3 General parameters for the Oracle-based PMDB

Parameter	Value	Improvement / Remarks
log_checkpoint_timeout	900	
open_cursors	300	Verify that you have enough open cursors for the application to execute.
Processes	300	Verify that you have enough processes for the application to execute.

Table 4 Oracle 9i parameters

Parameter	Value	Improvements / Remarks
session_cached_cursors	300	

Parameter	Value	Improvements / Remarks
DB-cache_advice	Verify that this parameter is set to `off` because the default value is on.	This parameter checks the work on the database and helps determine the recommended size of the Buffer pool. But this damages performance, so verify that it is on only for a short period.

Table 5 Oracle 10g parameter

Parameter	Value	Improvement / Remarks
sga_target	Automatic memory management. 1/4th of the physical memory.	Replaces the following parameters: • db_cache_size • shared_pool_size • large_pool_size • java_pool_size Verify that they are initialized with either 0 or the minimum value you need.

INIT.ORA example

The following is an example of the `init.ora` file for an Oracle 12.2 instance.

```
db_name='ORCL'  memory_target=1G processes=300 db_block_size=8192
db_domain='' db_recovery_file_dest='<ORACLE_BASE>/flash_recovery_area'
 db_recovery_file_dest_size=2G diagnostic_dest='<ORACLE_BASE>'
dispatchers='(PROTOCOL=TCP) (SERVICE=ORCLXDB)'
 open_cursors=300 remote_login_passwordfile='EXCLUSIVE'
 undo_tablespace='UNDOTBS1'
 undo_retention=900 #  15 minutes
 # You may want to ensure that control files are created on separate physical
 # devices
 control_files =  (ora_control1, ora_control2)
 compatible ='12.0.0'
```

How to disable the recycle bin in Oracle

Our recommendation is to disable the recycle bin and this section describes the background and the action on how to disable the recycle bin.

The recycle bin is actually a data dictionary table containing information about dropped objects. Dropped tables and any associated objects such as indexes, constraints, nested tables, and the likes are not removed and still occupy space. They continue to count against user space quotas, until specifically purged from the recycle bin or the unlikely situation where they must be purged by the database because of tablespace space constraints.

The following information is taken from the *Oracle Database Administration Guide*. In version 10.0 and 10.1 use the following command:

```
alter system set "_recyclebin"=FALSE scope=BOTH;
```

In version 10.2, use the following command:

```
alter system set recyclebin=off scope=BOTH
```

About backing up the database and creating archiving

The PMDB holds information for up to 3 years by default. If you want to backup this information, you should enable the Redo log archiving, and create a backup using RMAN. For more information, refer to the official Oracle Backup and Recovery documentation. If you do not create a backup, information will be lost when you have a database crash and you will not be able to recover data. If you create a backup without using the archive mode, you will be able to recover data to the time of the backup. All new operations (after the logs were archived) will be lost.

Enabling Automatic Undo Management (AUM)

To enable Automatic Undo Management (AUM), you must create a new tablespace and set the `init.ora` parameters. For additional information on how to enable AUM, see the Oracle Database Administrator's Guide, "Managing Undo Space."

To enable Automatic Undo Management (AUM)

1. Create the `undo tablespace` by running the following command:
   `CREATE UNDO TABLESPACE tablespace_name DATAFILE 'datafile name' SIZE 32M REUSE AUTOEXTEND ON EXTENT MANAGEMENT LOCAL`
2. Set the following `init.ora` parameters:
   - undo_management = AUTO
   - undo_tablespace = tablespace_name
   - undo_retention = 900

If the `undo tablespace` size is too large, you may decrease the number of seconds.

About maintenance operations

Every week the maintenance process analysis the PMDB, but to improve the database performance, the DBA must check and perform a rebuild to the indexes using the `alter index {name} rebuild` command.

About changing the size of tables and index extents

The PMDB uses the `products\dbms\files\tables_definition\ps_00_db_entities.xml` file to create the tablespaces for the tables and the indexes. To change the default sizes you can change the file before the PMDB installation or alter the tablespaces after the installation.

The following table shows the default extents sizes.

Table 6 Default extents sizes

Tablespace	Size
Large table	2 MB
Medium table	512 KB
Small table	80 KB
Large index	1 MB
Medium index	256 KB

Tablespace	Size
Small index	80 KB

Configuring PMDB process parameters

With the Load Data process parameters you are able to filter the Load Data results better, providing you with a better overview of the actual data. After selecting this parameter, you can select the instance for which you want to make changes to the default settings.

The following parameters that can be edited in AdminPoint>Warehouse Processes. Technology is the basis for the division of the process parameters:

- DB2
- J2EE
- Microsoft .NET
- Oracle
- Operating system
- Other
- PMDB
- SAP
- SQL Server
- Sybase
- Tuxedo
- Web Load Data
- WebSphere MQ

> ⓘ PMDB processes will not be executed if any of the following collector instances is down: SQL Server, Oracle, Sybase, or DB2.

DB2

About the DB2 Explain Statements process parameters

The DB2 Explain Statements process automatically explains the top *n* statements. By default, this process is run only once a night.

The following table describes the configurable process parameter and default value in the DB2 Explain Statements process.

Table 7-1 Configurable process parameter and its default value

Parameter	Default value
Explain only Top-N Statements (based on their In DB2 time)	100
Timeout if the Explain Process takes more than n minutes	60
Explain process stops after N Statements fail to be explained	100

About the DB2 Explain New Statements process parameters

The DB2 Explain New Statements process automatically explains the top *n* statements. By default, this process is run every 15 minutes.

The following table describes the configurable process parameter and default value in the DB2 Explain Statements process.

Table 7-2 Configurable process parameter and its default value

Parameter	Default value
Explain only Top-N Statements (based on their In DB2 time)	30

About the DB2 Load Data process parameters

The following table describes the configurable process parameter and default value in the DB2 Load Data process.

Table 7-3 Configurable process parameter and its default value

Parameter	Default value
Load only Top-N Programs information	300
Load only Top-N Statements information	500

About the DB2 Purge Internal Data process parameters

The DB2 Purge Internal Data process automatically purges old internal data from the PMDB tables. By default, this process is run once a week.

The following table describes the configurable process parameter and default value in the DB2 Purge Internal Data process.

Table 7-4 Configurable process parameter and its default value

Parameter	Default value
Purge plans data older than *n* days	14
Save the last n plans for each Explain Statement	3

J2EE

About the J2EE Load Data process parameters

The following table describes the configurable process parameter and default value in the J2EE Load Data process.

Table 7-5 Configurable process parameter and its default value

Parameter	Default value
Collect only Top-N records	1000
Number of executions over total service time	Low

Parameter	Default value
Load Data	30

Microsoft .NET

About the Microsoft .NET Perform SmarTune Analysis process parameters

The following table describes the configurable process parameters and default values in the Perform SmarTune Analysis process.

Table 7-6 Configurable process parameters and their default values

Parameter	Default value
Too many queued ASP.NET requests	Maximum request queue length n queued items (5).
Long wait time for ASP.NET requests	Maximum request wait time n milliseconds (1000).
Incorrect ASP.NET API cache usage - low hit ratio	Minimum cache hit ratio n percent (80).
Incorrect ASP.NET API cache usage - high turnover rate	Maximum cache turnover ratio n turnovers/second (100).
Incorrect ASP.NET output cache usage - low hit ratio	Minimum cache hit ratio n percent (80).
Incorrect ASP.NET output cache usage - high turnover rate	Maximum cache turnover ratio n turnovers/second (100)
Too many ASP.NET errors	Maximum errors per request ratio (5).
Too many unhandled ASP.NET exceptions	Maximum number of unhandled exceptions (0).
Too much time in garbage collection	Normal CPU utilizations n percent (70). Maximum time in garbage collection for low CPU utilization n percent (5). Maximum time in garbage collection for high CPU utilization n percent (10).
Object allocation problem	Maximum ratio garbage collection generations n percent (20).
Too many garbage collection calls by user	Maximum number of induced garbage collection calls (0)
Too many exceptions thrown	Maximum number of thrown exceptions per second (50).
Contention rate too high	Maximum number of unsuccessful locks per second (50).

Parameter	Default value
Incorrect connection string usage	Maximum number of different connection strings to database (10).
Slow database open connection	Maximum database open connection time n seconds (2).
Slow database requests execution	Maximum database execution time n seconds (10).
Slow page load	Maximum page load time n seconds (5).
Slow Web service invocation	Maximum Web service invocation time n seconds (5).
Slow XML document load	Maximum XML load time n seconds (2).
Incorrect garbage collection usage	Maximum number of garbage collection calls (0).
Slow serialization	Maximum serialization time per invocation time ratio n percent (10).
Slow de-serialization	Maximum de-serialization time per invocation time ratio n percent (10).
Too many calls for serialization	Maximum number of serialization invocations per second (100).
Too many calls for de-serialization	Maximum number of de-serialization invocations per second (100).
Incorrect usage of the Finalize method	Maximum number of calls to the finalized method (0).
Too many method invocations	Maximum method time n seconds (0.3). Maximum number of method invocations (100).
Too much time in lock	Maximum time in lock n seconds (0.2).

Oracle

About the Oracle Collect Bind Variables process parameter

The Oracle Collect Bind Variables process collects bind sets that were used while the statements were run (applicable for Oracle 10.1.0.4 and later). You can specify which mandatory statements to collect binds for by specifying their statement hash values, separated by a semicolon.

If this process is not run, no data is displayed in the Bind Variables view in the SQL tab and you are not able to choose a real set to run the statement with, in the Run Statement dialog box and the Bind Variables tab is disabled. By default, this process is run once an hour.

The following table describes the configurable process parameters and default values in the Oracle Collect Bind Variables process.

Table 7-7 Configurable process parameters and their default values

Parameter	Default value
Maximum number of statements to collect binds for:	20 ⓘ Increasing this parameter influences resource consumption and run time.
Maximum number of bind sets per statement to collect:	3 ⓘ Increasing this parameter influences resource consumption and run time.
List of mandatory statements to collect binds for:	No default value

About the Oracle Collect Instance Statistics process parameters

The Oracle Collect Instance Statistics process collects segment statistics. If you decide not to collect segment statistics, no active objects are displayed in the Objects tab.

The following table describes the configurable process parameters and default values in the Oracle Collect Instance Statistics process.

Table 7-8 Configurable process parameters and their default values

Parameter	Default value
Collect segment statistics every *n* hours	4 ⓘ "0" means that the segment statistics should not be collected.
Maximum number of segments to collect *n*	500
List of mandatory segments *n* (such as EMP_TBL, EMP_DETP_IDX)	No default value

About the Oracle Explain Statements process parameters

The Oracle Explain Statements process automatically explains the top *n* statements and parses the statements that were explained. By default, this process is run once a night.

The following table describes the configurable process parameters and default values in the Oracle Explain Statements process.

Table 7-9 Configurable process parameters and their default values

Parameter	Default value
Explain only Top-N Statements (based on their In Oracle time)	1000
Parse Top-N Statements (based on their In Oracle time)	1000
Timeout if Explain takes more than *n* seconds	60
Save the last *n* Explain plans for each explained statement	3

About the Oracle Load Data process parameters

The following table describes the configurable process parameter and default value in the Oracle Load Data process.

Table 7-10 Configurable process parameter and its default value

Parameter	Default value
Load only Top-N Programs information	300
Load only Top-N Statements information	500
Load only Top-N Objects information	700

About the Oracle Object Statistics Changes process parameters

The following table describes the configurable process parameters and default values in the Oracle Object Statistics Changes process.

Table 7-11 Configurable process parameters and their default values

Parameter	Default value
Keep track on changes in column statistics	Cleared
Collect database size statistics every *n* days	1

About the Oracle Perform SmarTune Analysis on Changes process parameter

The following table describes the configurable process parameter and default value in the Oracle Perform SmarTune Analysis on Changes process.

Table 7-12 Configurable process parameter and its default value

Parameter	Default value
Analyze the impact of changes in the last *n* days	30

About the Oracle Purge Data process parameters

The following table describes the configurable process parameters and default values in the Oracle Purge Data process.

Table 7-13 Configurable process parameters and their default values

Parameter	Default value
Purge schema changes data older than *n* weeks	52
Purge database structure data older than *n* weeks	26

About the Oracle Real Execution Plans process parameter

The Oracle Real Execution Plans process collects real execution plans stored in Oracle (applicable for Oracle 9i and later) system tables. If this process is not run, no real execution plan information is displayed in the SQL tab. By default, this process is run once an hour.

The following table describes the configurable process parameter and default values in the Oracle Actual Execution Plans process.

Table 7-14 Configurable process parameter and its default value

Parameter	Default value
Maximum number of statements to collect actual execution plans for	500 ⓘ Increasing this parameter influences resource consumption and run time.

About the Oracle Applications Load Data process parameters

The following table describes the configurable process parameter and default value in the Oracle Applications Load Data process.

Table 7-15 Configurable process parameter and its default value

Parameter	Default value
Collect only Top-N records	1000
Number of executions over total service time	Low

Operating system

About the Operating System Load Data process parameters

The following table describes the configurable process parameters and default values in the Operating System Load Data process.

Table 7-16 Configurable process parameters and their default values

Parameter	Default value
Collect only Top-N records	1000
Size of memory over total CPU time	Low
Include Process ID details in time slice summary level	Cleared
Include Command and Process ID only got processes with CPU time greater than n seconds per time slice	10
Mask command according to	No default value

Other

About the Other Load Data process parameters

The following table describes the configurable process parameter and default value in the Other Load Data process.

Table 7-17 Configurable process parameter and its default value

Parameter	Default value
Collect only Top-N records	1000
Number of executions over total service time	Low
Mask Client IP according to	No default value

PMDB

About the PMDB Purge Data process parameters

The following table describes the configurable process parameters and default values in the PMDB Purge Data process. The following set of parameters is defined for each technology, server, and instance that is installed in the Precise system.

Table 7-18 Configurable process parameters and their default values

Parameter	Default value
Purge slice data older than	4 weeks
Purge hour data older than	4 weeks
Purge daily data older than	3 months

Parameter	Default value
Purge weekly data older than	12 months
Purge monthly data older than	24 months

About the PMDB Calculate Baselines process parameters

The following table describes the configurable process parameters and default values in the PMDB Calculate Baselines process.

Table 7-19 Configurable process parameters and their default values

Parameter	Default value
Retrieve data from the *n* Summary tables	Daily
Calculate based on last *n* day(s)	64

About the PMDB Maintenance (Weekly) process parameters

The following table describes the configurable process parameters and default values in the PMDB Maintenance (Weekly) process.

Table 7-20 Configurable process parameters and their default values

Parameter	Default value
Purge data of instances that are no longer monitored	Cleared
Purge job history after *n* days	7
Defragmentation threshold after *n* percent	50 If the value is 100, no defragmentation is performed.

About the PMDB Summarize Data process parameters

The following table describes the configurable process parameters and default values in the PMDB Summarize Data process.

The following set of parameters is defined for each technology, server, and instance that is installed in the Precise system.

Table 7-21 Configurable process parameters and their default values

Parameter	Default value
Summarize to daily level, only for top *n* rows	2000 rows
Summarize to weekly level, only for top *n* rows	3000 rows

Parameter	Default value
Summarize to monthly level, only for top *n* rows	3000 rows

About the PMDB Daily Purge Data process parameters

The following table describes the configurable process parameters and default values in the PMDB Daily Purge Data process.

The following set of parameters is defined for each technology, server, and instance that is installed in the Precise system.

Table 7-22 Configurable process parameters and their default values

Parameter	Default value
Purge short slice data older than *n* days	1 day

About the PMDB Load Data process parameters

The following table describes the configurable process parameters and default values in the PMDB Load Data process. The following set of parameters is defined for each technology, server, and instance that is installed in the Precise system.

Table 7-23 Configurable process parameters and their default values

Parameter	Default value
Use Defaults	30 seconds

SAP

About the SAP Load Data process parameters

The following table describes the configurable process parameter and default value in the SAP Load Data process.

Table 7-24 Configurable process parameter and its default value

Parameter	Default value
Load only transactions with service time greater than or equal to n seconds per time slice	0

About the SAP Organizational Mapping process parameters

The following table describes the configurable process parameter and default value in the SAP Organizational Mapping process.

Table 7-25 Configurable process parameter and its default value

Parameter	Default value
Maps each organization, as it appears in the SAP system, to the physical site where its employees are located. Each organization can be mapped to more than one location. This mapping data is used to show SAP workload statistics according to locations.	Daily

SQL Server

About the SQL Server Collect Operational Statistics process parameters

The following table describes the configurable process parameter and default value in the SQL Server Collect Operational Statistics process.

Table 7-26 Configurable process parameter and its default value

Parameter	Default value
Maximum number of objects to collect:	500

About the SQL Server Collect Schema Changes process parameters

The following table describes the configurable process parameters and default values in the SQL Server Collect Schema Changes process.

Table 7-27 Configurable process parameters and their default values

Parameter	Default value
Load instance definition changes	Selected
Load SQL Server Agent jobs definition changes	Selected
Load database schema changes	Selected
Load database schema changes in syscolumns	Selected
Load database schema changes in sysindexkeys	Selected

About the SQL Server Collect Space Utilization process parameters

The following table describes the configurable process parameters and default values in the SQL Server Collect Space Utilization process.

Table 7-28 Configurable process parameters and their default values

Parameter	Default value
Load DB files size information	Selected
Load table and indexes space information	Selected
Load partition size information (for 2005 only)	Cleared

About the SQL Server Explain Statements process parameters

The following table describes the configurable process parameters and default values in the SQL Server Explain Statements process.

Table 7-29 Configurable process parameters and their default values

Parameter	Default value
Explain only the to n batches (based on their In MS-SQL time)	1000
Do not explain statements that were explained in the last n days	14

About the SQL Server Explain New Statements process parameters

The following table describes the configurable process parameter and default value in the SQL Server Explain New Statements process.

Table 7-30 Configurable process parameter and its default value

Parameter	Default value
Explain only Top-N batches (based on their In MS-SQL time)	30

About the SQL Server Load Data process parameters

The following table describes the configurable process parameter and default value in the SQL Server Load Data process.

Table 7-31 Configurable process parameter and its default value

Parameter	Default value
Load only Top-N Statements information	500

About the SQL Server Perform SmarTune Analysis process parameters

The following table describes the configurable process parameters and default values in the SQL Server Perform SmarTune Analysis process.

Table 7-32 Configurable process parameters and their default values

Parameter	Default value
Check the following	Findings of Top-N Statements (20)
	Findings of Top-N Objects (10)
	History activity for last n days (7)
	Minimum cost of heavy operators n percent (1)
Top statements should have	Minimum In MS-SQL n minutes/hour (1)
Scalability change for top statements	Increase in total executions of n percent (10)
Considered problematic for top statements	In MS-SQL increased at least by n percent (20)
	Lock wait time exceeds n percent (10)
Top objects should have	Minimum n pages (100)
	Minimum IN MS-SQL n minutes per hour (15)
Major growth of an object defined by	Increase of n rows (10000)
	Increase of n pages (20)
Scalability change for top objects	Increase in total executions of n percent (10)
Considered problematic for top objects	In MS-SQL increased by at least n percent (10)
	Lock wait time exceeds n percent (20)
Instance is locked when	Lock wait time exceeds n percent (10)
Tempdb is considered a bottleneck when	Tempdb wait time exceeds n percent (10)
Buffer cache is considered to be too small when	Buffer cache hit ratio counter is less than n percent (90)
	Number of buffers written by lazy writer is more than n buffers per second (20)
	Data pages stay in buffer less than n seconds (300)
	SQL Server page faults is less than n per second (20)
	I/O wait time exceeds n percent (5)
Other applications influence SQL Server memory resources	SQL Server page faults/sec. exceeds n (20)
	Non SQL Server page faults/sec. is less than n (20)
	SQL Server memory is less than n percent of total memory (30)

Parameter	Default value
Other applications influence SQL Server CPU resources	SQL Server CPU is less than n percent (20) Non SQL Server CPU exceeds n percent (50) Minimum CPU queue length n (2)
Transaction log is considered a bottleneck when	Log wait time exceeds n percent (10) The average log flushes/sec. is greater than n (100) The average log flush waits/sec. is greater than n (10) Log flush wait time exceeds n minutes per hour (5)
Statement Findings	The access plan of the statement contains heavy operators (selected) The access plan of the statement has issued a missing indexes warning (selected) Statements accessing the object have missing statistics warnings (selected) The average In MS-SQL time has increased due to a schema change (selected) The average In MS-SQL time increased due to a growth in statement executions (selected) The average In MS-SQL time increased due to a growth in tables (selected) The average In MS-SQL time has increased (but the increase is not due to schema changes, table size or scalability) (selected) The total In MS-SQL time of the statement is consistently high (selected) The statement's lock wait time is very high (selected)

Parameter	Default value
Object Findings	Statements accessed the object using heavy operators (selected)
	Statements accessing the object have missing indexes warnings in their access plan (selected)
	Statements accessing the object have missing statistics warnings in their access plan (selected)
	Total In MS-SQL time has increased due to a schema change (selected)
	Total In MS-SQL time has increased due to a major growth in table size (selected)
	Total In MS-SQL time has increased due to major growth in total executions of statements (selected)
	Total In MS-SQL time has increased (but the increase is not due to schema changes, table size or scalability) (selected)
	The object has unnecessary indexes (containing many updates that are not in use by any SELECT statement) (selected)
	The object's lock wait time is very high (selected)
Instance Findings	Extensive lock wait time (selected)
	Tempdb is a bottleneck (selected)
	Buffer cache is too small (selected)
	Other applications influence SQL Server memory resources (selected)
	Other applications influence SQL Server CPU resources (selected)
	The transaction log file is a bottleneck (selected)

About the SQL Server Purge Internal Data process parameters

The following table describes the configurable process parameters and default values in the SQL Server Purge Internal Data process.

Table 7-33 Configurable process parameters and their default values

Parameter	Default value
Preserve execution plan details	6 months
Preserve access path and cost changes history	18 months
Preserve schema change log	6 weeks

Sybase

About the Sybase Collect Space Utilization process parameters

The following table describes the configurable process parameters and default values in the Sybase Collect Space Utilization process.

Table 7-34 Configurable process parameters and their default values

Parameter	Default value
Load devices space information	Selected
Load table and indexes space information	Selected
Load segments space information	Selected

About the Sybase Explain Statements process parameters

The following table describes the configurable process parameters and default values in the Sybase Explain Statements process.

Table 7-35 Configurable process parameters and their default values

Parameter	Default value
Explain only Top-N Batches (based on their In Sybase time)	1000
Do not explain statements that were explained in the last n days	14

About the Sybase Explain New Statements process parameters

The following table describes the configurable process parameter and default value in the Sybase Explain New Statements process.

Table 7-36 Configurable process parameter and its default value

Parameter	Default value
Explain only TopN Batches (based on their In Sybase time)	10

About the Sybase Load Data process parameters

The following table describes the configurable process parameter and default value in the Sybase Load Data process.

Table 7-37 Configurable process parameter and its default value

Parameter	Default value
Load only Top-N Statements information	500

Tuxedo

About the Tuxedo Load Data process parameters

The following table describes the configurable process parameter and default value in the Tuxedo Load Data process.

Table 7-38 Configurable process parameter and its default value

Parameter	Default value
Collect only Top-N records	1000
Number of executions over total service time	Low

Web Load Data

About the Web Load Data process parameters

The Web Load Data process parameters dialog box contains three columns that cannot be configured. The Name (with Instance and server name or Cluster name), Type (with Instance or Cluster), and Server (with the name of the server) columns enable you to see that the row you select is an instance or a cluster. When you select a cluster, the Cluster Instances button will become active and you are able to see more information on the instances within the selected cluster in the list shown in the Cluster Instances dialog box. Setting parameters at the cluster level affects all instances in that cluster. It is impossible to set parameters for an instance within a cluster individually.

The following table describes the configurable process parameters and default values in the Web Load Data process.

Table 7-39 Configurable process parameters and their default values

Parameter	Default value
Collect only Top-N URL records	1000
Number of executions weight over average URL service +network time	Medium
Consider SLA breach when filtering data	Selected
Collect only Top-N page records	100
Number of executions over average page response time	Medium
Consider SLA breach when filtering data	Selected

Parameter	Default value
Collect Client IP	Cleared
Collect country	Cleared
Collect state	Cleared
Collect city	Cleared
Mask Client IP according to	No default value
Mask Private IP according to	No default value

WebSphere MQ

About the WebSphere MQ Load Data process parameters

The following table describes the configurable process parameters and default values in the WebSphere MQ Load Data process.

Table 7-40 Configurable process parameters and their default values

Parameter	Default value
Collect only Top-N records	1000
Number of executions over total service time	Low
Collection mode:	Basic
In extended collection mode, collect from the message body the first n bytes	250
Include only messages with service time greater than n seconds per time slice	1
Include only messages that pass through queues that met one of the patterns in	No default value
Include only users that met one of the patterns in	No default value
Include only messages of applications that met one of the patterns in	No default value
Include only messages that their body met one of the patterns in	No default value

Configuring Precise for J2EE

This section includes the following topics:

- About configuring Precise for J2EE
- About configuring registry settings
- About configuring Precise for J2EE features
- About configuring EJB 3.0 monitoring
- About configuring JMS monitoring
- About configuring findings settings
- About configuring HTTP query parameters and Java method arguments capturing
- About file system security
- About monitoring settings for a J2EE remote instance

About configuring Precise for J2EE

> ⓘ Precise for J2EE v9.0 introduces the Monitor Settings dialog box to the user interface. This dialog box is recommended for configuring most of the common Precise for J2EE instrumentation. For details, see About configuring Precise for J2EE settings in the Precise for J2EE User Guide. In some cases, however, it may be preferable to manually perform specific instrumentation. This section describes these cases.

In Precise for J2EE v9.0, the installation and configuration directories for all instances (for all installation types) are centrally located in the registry, installed on a central JVM defined as the host, with the following path: `products\i3fp\registry\`. Whenever the JVMs are restarted, they are configured according to the information found in the XML files located in the registry folder. As a result, any change made to a file in the registry will take affect after a JVM restart.

In addition to the registry configuration files, local directories still exist on each JVM. The file system path of the local Precise installation directory is `<i3_root>`. `<i3_root>` may also be called the Precise root directory, such as `PreciseSoftware`.

The Precise for J2EE products directory path is `\<i3_root>\products\j2ee`.

The JVM identifier that was chosen or given (depending on the installation type) during installation is referred to as *JVMID*. The *JVMID* directory is used to store JVM-specific settings. It is referred to within the file system paths, such as `\i3\products\j2ee\config\JVMID`. For example, if you choose PetStore as a *JVMID*, the *JVMID* directory would be `\i3\products\j2ee\config\PetStore`.

Any changes made on a local directory will immediately take effect, but only until the next JVM restart, when the JVM will reload the configuration found in the registry and override the local changes.

About configuring registry settings

This section describes the registry settings configuration process.

Registry structure and inheritance

Some configurations are relevant per instance (e.g. collect-post-parameters) and some are relevant for the J2EE collector in general (e.g. preferred-data-port). Others are relevant for the whole product (e.g. user-agents). Instance configuration resides in the instance registry file:

`products\i3fp\registry\instances\j2ee\<server-name>\<instance_name>\<settings>.xml`

> ⓘ You can set instance configuration in the `products\i3fp\registry\instances\j2ee\<server-name>\<settings>.xml` file if you want to set the same setting for all instances on the machine. Consider however, that settings in the instance registry will override the settings in the server registry.

J2EE collector configuration resides in the registry file:

`products\i3fp\registry\instances\j2ee\<server-name>\<settings>.xml`

> ⓘ You can set the configuration for all J2EE collectors at once by modifying the file `products\i3fp\registry\products\j2ee\focal-point.xml`
>
> Consider however, that settings in the server registry will override the settings in the `focal-point.xml` file.

Product level configuration resides in the registry file: `products\i3fp\registry\products\j2ee\focal-point.xml`

J2EE collector behavior is set via the registry entry `<j2ee-collector>`

Cluster configuration resides in the registry file:

`products\i3fp\registry\clusters\j2ee\<cluster-name>\<settings>.xml`

Instance registry

Configuring the instance registry only affects the specific instance. Every element in the instance registry can also be specified in the server registry file, in which case it affects all the instances on that server.

To change the instance registry

1. On the relevant Precise node (proxy), manually change the products\i3fp\registry\instances\j2ee*<server-name>**<instance_name>**<settings>*.xml file.
2. Save the file.
3. Run the `update-main-registry` CLI command on the same Precise node.
   - **UNIX**. ./infra/bin/psin_cli.sh -action update-main-registry -i3-user admin -i3-clear-password admin -registry-path /registry/clusters/j2ee/<server-name>/<instance-name>
   - **Windows**. infra\bin\psin_cli.bat -action update-main-registry -i3-user admin -i3-clear-password admin -registry-path /registry/clusters/j2ee/<server name>/<instance name>

This will force a registry sync from the node's FocalPoint to the main FocalPoint.

Cluster registry

Configuring the cluster registry affects all instances of the specific cluster.

> ⓘ All instances in the cluster need to be restarted via AdminPoint after changing the cluster registry.

To change the cluster registry:

1. On the relevant Precise node (proxy), manually change the products\i3fp\registry\clusters\j2ee*<cluster-name>**<settings>*.xml file.
2. Save the file.
3. Run the `update-main-registry` CLI command on the same Precise node.

- **UNIX**. ./infra/bin/psin_cli.sh -action update-main-registry -i3-user admin -i3-clear-password admin -registry-path /registry/clusters/j2ee/<cluster-name>
- **Windows**. infra\bin\psin_cli.bat -action update-main-registry -i3-user admin -i3-clear-password admin -registry-path /registry/clusters/j2ee/<cluster_name>

This will force a registry sync from the node's FocalPoint to the main FocalPoint.

> ⓘ Cluster registry changes override the FocalPoint registry changes.

About configuring Precise for J2EE features

This section includes configuration details for exception seeking and data filtering.

About configuring exception seeker options

Precise for J2EE catches exceptions during a time slice, based on its instrumentation configuration and filtering criteria. Filtering can be performed at the JVM level (that is: instrumentation and collector.xml level) and the GUI level to configure portlets.

Exceptions filtered at the JVM level will not be sent to the Precise for J2EE FocalPoint. View the exceptions on the Precise for J2EE Dashboard tab.

Configuring instrumentation parameters (JVM)

Perform the following steps to configure instrumentation parameters.

To configure instrumentation parameters (JVM)

1. In any Precise for J2EE tab, go to Settings>Monitor Settings.
2. In the Monitor Settings dialog box, mark the "Monitor Exceptions" check box.
3. Restart the JVM.

> ⓘ The changes take place at the instrumentation level. Therefore, the changes will only occur after restarting the JVM.

If you have a large amount of exceptions, focus on a specific exception by defining a filter on an exception class name (JVM).

To define a filter on an exception class name

1. Open the collector.xml file in the registry folder. Use this file to define a filter on an exception class name, using regular expressions.
2. Edit the following tag:
   `<exceptionFilterByName></exceptionFilterByName>`
   The default filter tag is:
   `<exceptionFilterByName>.*</exceptionFilterByName>`
3. Run the relevant CLI command as it appears in step 3 of Instance registry or Cluster registry.

About configuring data filtering

To reduce the quantity of collected data and propagate mainly meaningful data, a filtering mechanism is used. This mechanism is used to filter nodes and invocations before loading the data into the PMDB.

About the data filtering stages

The Precise for J2EE data filtering mechanism contains the following three stages:

1. **Top N branches filtering**. Sorts each invocation tree by service time, only keeping the top n entry points (with their underlying call tree), as defined in the filtering configuration.
2. **Filtering by service time**. For each entry point, all underlying branches (including locked methods) with a service time lower than the defined percent of the entry point's service time are filtered.

   > ⓘ No information regarding the filtered branch's SQLs or exit points will be displayed.

3. **Filtering by work time**. Filters branches (only branches, and keeps their underlying call tree) with a work time lower than the defined threshold.

   > ⓘ Locked methods are always displayed.

   It is highly recommended not to alter the default data filtering values. Changes to the default values should only be made in extreme cases that either require maximum data, such as POC environments, or in cases of data overflow where stricter data filtering must be applied.

Configuring data filtering

To configure data filtering:

1. Open the collector.xml file in the registry folder.
2. Add the following element:
   ```
   <agent-config>
       <vendor>undefined (version undefined)</vendor>
       <memloginterval>100</memloginterval>
       <agginterval>30</agginterval>
       <topnsql>5</topnsql>
       <minSQLThreshold>100</minSQLThreshold>
       <loggerEnabled>true</loggerEnabled>
       <dirconnenabled>false</dirconnenabled>

       <dirconnport>20764</dirconnport>
       <exceptionFilterByName>.*</exceptionFilterByName>

       <filteringEnabled>true</filteringEnabled>
       <filteringMaxInvocationTrees>100</filteringMaxInvocationTrees>

       <filteringPercent>1</filteringPercent>
       <filteringThreshold>5</filteringThreshold>

       <useStartupClass>false</useStartupClass>

   </agent-config>
   ```
3. Run the relevant CLI command as it appears in step 3 of Instance registry or Cluster registry.

The following table lists the available parameters:

Table 8-1 Data filtering configuration parameters

Parameter	Description
filteringEnabled	Activates and deactivates data filtering. Type: Boolean Default value: true
filteringMaxInvocationTrees	The maximum number of entry points to keep, based on service time. Type: Integer Default value: 100 ⓘ For more information, see About the data filtering stages.
filteringPercent	The minimum percent value of the entry point's service time necessary to keep a node. Type: Integer Default value: 1 ⓘ For more information, see About the data filtering stages.
filteringThreshold	The minimum work time value (milliseconds) threshold to keep a node. Type: Integer Default value: 5 ⓘ For more information, see About the data filtering stages.

Keeping exceptions of filtered invocations

After selecting the data filtering method, the user can choose to mark the option: "Keep exceptions for filtered methods". The exceptions will be included in the total per JVM, even though their source is not displayed.

About configuring EJB 3.0 monitoring

Precise for J2EE provides default monitoring for EJB 3.0 annotations. Metadata annotations are a key element in the developmental simplification of EJB 3.0 applications. Metadata annotations are used by the developer to specify expected requirements on container behavior, to request the injection of services and resources, and to

specify object/relational mappings. Metadata annotations may be used as an alternative to the deployment descriptors that were required by earlier versions of the EJB specification.

While the EJB 3.0 specification allows (and promotes) defining EJBs using annotations, it is also still possible to use an XML deployment descriptor (the previous EJB 2.0 method). Because this is not automatically monitored by Precise, the following procedure must be run for the appropriate application server, as follows:

WebLogic

To monitor EJB3.0 in a WebLogic environment using a XML deployment descriptor file only (no annotations)

1. Add the following entry to the <i3_root>/products/i3fp/registry/clusters/j2ee/<cluster-name>/ InstrumenterConfigLis t.xml file:
   <config-file>
       WeblogicEjb3Config-FD.xml
   </config-file>
2. Run the relevant CLI command as it appears in step 3 of Instance registry or Cluster registry.

WAS

To monitor EJB3.0 in a WAS environment using a XML deployment descriptor file only (no annotations)

1. Add the following entry to the <i3_root>/products/i3fp/registry/clusters/j2ee/<cluster-name>/ InstrumenterConfigLis t.xml file:
   <config-file>
       WebSphereEjb3Config-FD.xml
   </config-file>
2. Run the relevant CLI command as it appears in step 3 of Instance registry or Cluster registry.

Jboss

To monitor EJB3.0 in a Joss environment using a XML deployment descriptor file only (no annotations)

1. Add the following entry to the <i3_root>/products/i3fp/registry/clusters/j2ee/<cluster-name>/ InstrumenterConfigLis t.xml file:
   <config-file>
       JbossEjb3Config-FD.xml
   </config-file>
2. Run the relevant CLI command as it appears in step 3 of Instance registry or Cluster registry.

About configuring JMS monitoring

By default, Precise for J2EE instrumentation of JMS calls is disabled, and must be manually configured to enable monitoring for transactions that use synchronous JMS. Since JMS is an asynchronous API, to configure this functionality the thread that initiates the JMS calls on the client-side must contain a method which calls both the JMS send and receive operations, and represents the bounds of the "synchronous JMS call".

For example, the following method could be configured as a JMS exit point:

```
public String sendSyncMessage (Message msg) {
// Do some things... MessageProducer.send(msg);
// Do some things...
Message reply = MessageConsumer.receive();
return reply.getText();
```

1. In the `<i3_root>\products\j2ee\config` folder of the instances that sample JMS client and server calls, add the following element to the `j2ee\config\[instance_id]\instrumenterconfiglist.xml` file:
   <config-file>
       JMS_Calls_AOP.xml
   </config-file>
2. In the j2ee instrumenter config folder, apply the following change to the `j2ee\config\instrumenter\JMS_Calls_AOP.xml` file:
   aspect.wrapper.target.class: jms.wrapper
   aspect.wrapper.target.method:
   sendSyncMessage
   aspect.wrapper.target.signature: *
3. Stop and restart the monitored JVMs.

About configuring findings settings

The Precise for J2EE Dashboard and Activity tabs include the findings area. This area displays specific problematic issues detected in the J2EE activity monitored by Precise for J2EE. The findings are displayed according to the definitions configured in the `findings-config.xml` file located in the registry file.

To modify the findings parameter settings

1. Open the `findings-config.xml` file from the following path on the FocalPoint server:
   `<i3_root>\products\i3fp\registry\products\j2ee\findings-config.xml`
2. Modify the desired configurable parameters. For more information regarding these parameters, see Table 8-2 Configurable parameters in the findings-config.xml file.
3. Run the relevant CLI command as it appears in step 3 of Instance registry or Cluster registry.
4. Restart the Precise for J2EE FocalPoint.

The following table describes the configurable findings parameters.

> ⓘ Configuring a parameter not mentioned in the Configurable parameters in the findings-config.xml file table may lead to errors in the displayed findings, or a general failure to load the findings. To configure such parameters, contact Precise Customer Support.

Table 8-2 Configurable parameters in the findings-config.xml file

Parameter	Description
max-findings-to-display	The maximum number of findings to be displayed in the user interface. Type: Integer Default value: 7
timeout-in-seconds	The time limit (in seconds) that the findings query will run. Type: Integer Default value: 20

Parameter	Description
enabled	The finding will only be displayed if it is enabled. Type: Boolean Default value: true
tab	The finding will only be displayed in the specific tab (activity or dashboard) if it is enabled. Type: Boolean Default value: true
topN-to-display	The maximum number of times a specific type of findings will be displayed (with different contexts). Type: Integer Default value: 2
thresholds	Each finding's severity is calculated based on finding specific thresholds. Type: Integer

About configuring HTTP query parameters and Java method arguments capturing

Under certain circumstances, you may want to capture the HTTP servlet request query arguments and select Java method arguments. This section describes the configuration steps required to enable these features in Precise for J2EE.

A portal typically uses HTTP servlet request query string parameters to identify user services. Usually, very few servlets perform many different functions. The functions are differentiated by various query string parameter name=value pairs that are submitted in the URL. Additionally, URI paths may be independent of the query string parameters. Capturing the HTTP servlet request query string parameters to identify portal services is described in this section. In the following hyperlink, the substring s=VRTS is an HTTP query string parameter:

http://finance.yahoo.com/q?s=VRTS

Application services, such as BEA's Tuxedo, provide a Java API to select the desired service. Application programmers usually select the desired service by passing to the API an object, or string selector, describing their desired service. Capturing the Java parameters that are used to identify Tuxedo services is described in this section.

About getting started

You must first install Precise for J2EE and be able to collect and view application data in the Precise for J2EE user interface.

About capturing HTTP servlet query parameters

The configuration and interpretation of capturing HTTP query parameters is described in the following sections:

- About configuring parameter capture
- About using parameter capture

About configuring parameter capture

To enable HTTP Servlet Query parameter capture, add the following system property to your application server's Java command line:

```
-Dindepth.j2ee.http.request.query.parameters=[options]
```

The value of the property (its [options]) controls which parameters are captured. A comma-separated list of parameter names is used to identify one or more specific parameters. A single asterisk is used to represent all parameters.

In Precise for J2EE, the captured parameter names and values are appended to the URL of HTTP type invocations. A colon separates the method name from the parameter list. Each parameter is separated from its value by an equal (=) symbol. Commas separate parameter name/value pairs. Commas replace Ampersands. A maximum combined method name and parameter length of 128 characters is captured and displayed. See About using parameter capture.

In addition, to print the entire HTTP query string and URI in the application server's console window, add the following system property to your application server's Java command line. This feature is useful for learning about the applications usage of specific query arguments:

```
-Dindepth.j2ee.verbose.parameters=true
```

In this example Java command line, we enable HTTP Servlet Query parameter capture for action and itemld parameters that are used in the Java PetStore application running on WebLogic 9 by adding the following:

```
…
-Dindepth.j2ee.verbose.parameters=true
-Dindepth.j2ee.http.request.query.parameters=action,itemid
…
```

In a PeopleSoft v8 demo example, we found this property value sufficient to capture interesting parameters that differentiated executions of each screen:

```
-Dindepth.j2ee.http.request.parameters=FolderRef,Page,PortalActualURL,cmd
```

About using parameter capture

Typically, we start by using the following configuration:

```
-Dindepth.j2ee.http.request.parameters=* and
-Dindepth.j2ee.verbose.parameters=true
```

When we use this configuration all parameters on the console are printed, enabling us to inspect and learn which parameters are significant as we navigate through the application.

After we learn which parameters are important to us, we focus the capture on a few parameters by specifying them individually in a list, for example
```
-Dindepth.j2ee.http.request.query.parameters=param1,param2,param5,param7
```

> ⚠️ Capturing all parameters can use significant system resources. As a result, capturing all parameters all the time is not recommended in production environments.

Figure 8-1 HTTP SRs using
`–Dindepth.j2ee.http.request.query.parameters=param1,param2,param5,param7`
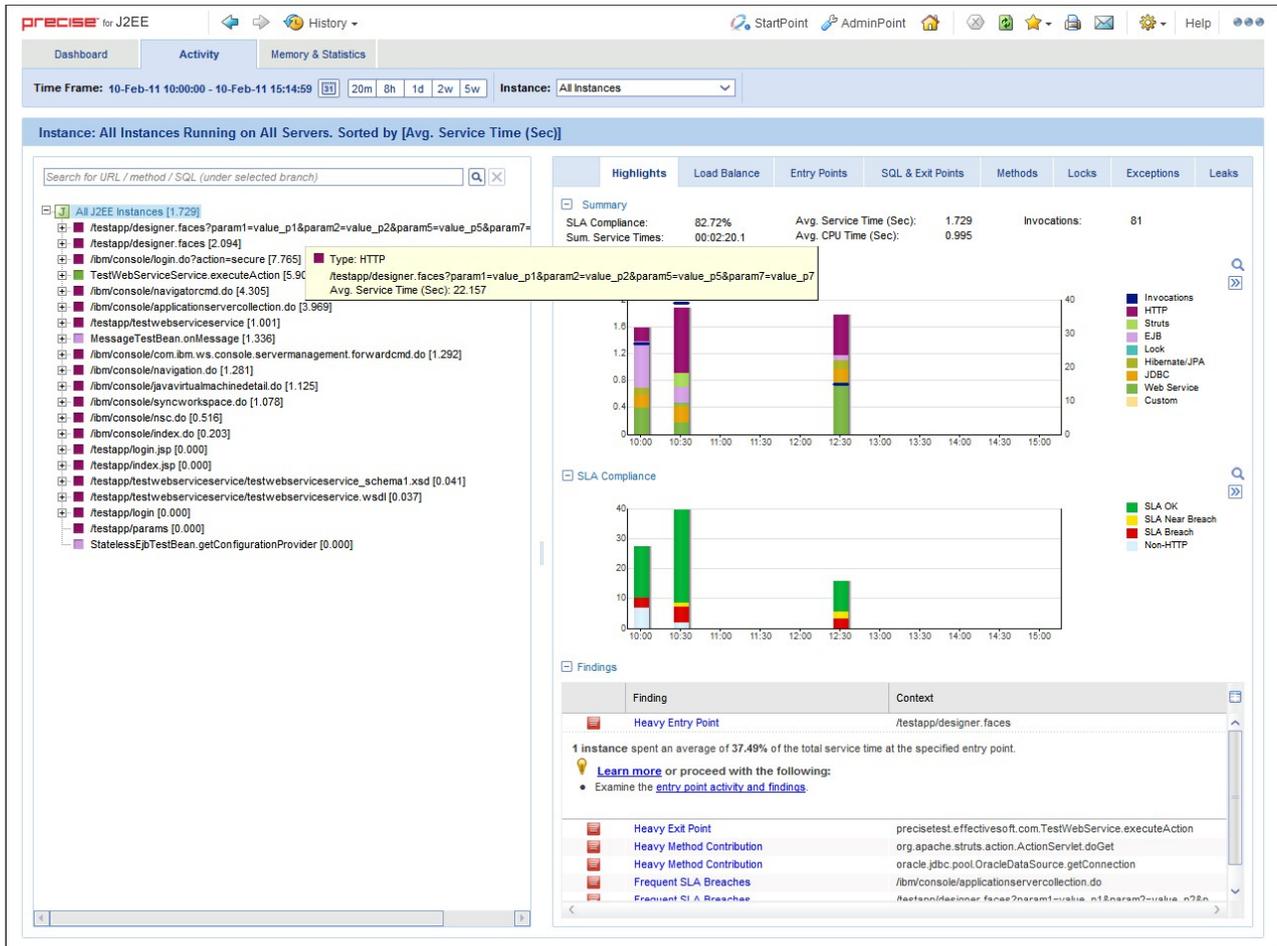
About capturing Java method arguments

The following example illustrates the configuration and interpretation of captured Java method arguments. This example describes the instrumentation of Tuxedo service names from BEA's Jolt API as declared in the Jolt.xml instrumentation configuration file included in Precise for J2EE. To enable the Tuxedo service name capture, add the following element to your <i3_root>/products/i3fp/registry/clusters/j2ee/<*cluster-name*>/InstrumenterConfigList.xml file:

```
<!--
BEA Jolt custom instrumentation
-->
<config-file>
    Jolt.xml
</config-file>
```

To enable Java method argument capture, add a custom instrumentation element to an enabled instrumentation configuration file. The following example shows one of these elements that are configured for bea.jolt.JoltRemoteService. The example is taken from the Jolt.xml instrumentation configuration file that is distributed with Precise for J2EE. In this example, we display the Tuxedo service name whenever the bea.jolt.JoltRemoteService.call method is executed. A complication is that the Tuxedo service name is specified in the bea.jolt.JoltRemoteServiceconstructor (init) method and not passed directly to each execution of call.

Following is an example for the Java Method argument capture, custom instrumentation element, for bea.jolt.JoltRemoteService:

```
<instrumenter-config>
    <custom-config>
        <java-classes>
            <java-class>
            <class-name>
                bea.jolt.JoltRemoteService
            </class-name>
            <methods>
            <capture-method>
            <name>
            init
            </name>
            <capture-param-index>
            0
            </capture-param-index>
            </capture-method>
            <method>
                <name>call</name>
            </method>
            </methods>
            </java-class>
        </java-classes>
    </custom-config>
</instrumenter-config>
```

The <capture-method> element that is shown in this example identifies the Java method to be used to capture the data. The captured data is displayed in Precise for J2EE for methods specified by the <method> elements that share the same <java-class> element as the <capture-method> element. For the bea.jolt.JoltRemoteService.call case, since the Tuxedo service name is actually specified in the constructor (init) method, init is specified as the <capture-method> element.

The <capture-param-index> element defines the index count of the Java method argument to capture. The first argument is numbered zero, the second argument is numbered 1, and so on. The captured argument is converted to a string, using toString(), and appended (separated by a colon) to the method name of interest.

If you use the <capture-param-index> element without the <capture-method> element, the argument value of the index number that the <capture-param-index> element specifies is displayed following the associated method name. In other words, when <capture-method> is not used and <capture-param-index> is used, the argument that is specified by the <capture-param-index> element is both captured and appended to the method that is entered in the <name> element. For example, the following configuration appends the doProcess method's first argument value to the doProcess method name that is displayed in Precise for J2EE.

```
<methods>
    <method>
        <name>doProcess</name>
            <capture-param-index>0</capture-param-index>
```

```
    </method>
  </methods>
```

About file system security

Precise for J2EE requires two types of system users: the Precise user and the user who starts the application server that Precise for J2EE monitors. These users are created when you install Precise for J2EE. While the Precise user must have read, write, and execute permissions for all subdirectories in the `<i3_root>` and the `<i3_root>/products/j2ee` directories, you can restrict the permissions for some subdirectories for the application server user or group.

> ⓘ If the Precise user and the application server user are the same, give the application server user the same permissions as the Precise user.

The following table shows the minimum permissions that you can set for the application server user or group for the `<i3_root>` dir

Table 8-3 Minimum permissions for the `<i3_root>` directory for the application server user

Directory	Minimum permissions
`<i3_root>/distribution`	read, execute
`<i3_root>/infra`	read, execute
`<i3_root>/java`	read, execute
`<i3_root>/products`	read, execute
`<i3_root>/support`	read, write, execute
`<i3_root>/logs`	read, write, execute

The following table shows the minimum permissions that you can set for the application server user or group for the `<i3_root>/products/j2ee` directory.

Table 8-4 Minimum permissions for the `<i3_root>/products/j2ee` directory for the application server user

Directory	Minimum permissions
`<i3_root>/products/j2ee`	read, execute
`<i3_root>/products/j2ee/archive`	read, execute
`<i3_root>/products/j2ee/cache`	read, execute
`<i3_root>/products/j2ee/config`	read, write, execute
`<i3_root>/products/j2ee/install`	read, execute
`<i3_root>/products/j2ee/lib`	read, execute

Directory	Minimum permissions
`<i3_root>/products/j2ee/samples`	read, execute
`<i3_root>/products/j2ee/support`	read, execute
`<i3_root>/products/j2ee/bin`	read, execute
`<i3_root>/products/j2ee/classes`	read, execute
`<i3_root>/products/j2ee/etc`	read, write, execute
`<i3_root>/products/j2ee/installed`	read, execute
`<i3_root>/products/j2ee/logs`	read, write, execute
`<i3_root>/products/j2ee/smartlink`	read, write, execute
`<i3_root>/products/j2ee/tac`	read, write, execute

About monitoring settings for a J2EE remote instance

This section describes the following monitoring settings:

- Time slice size
- Using Web patterns
- Enabling/Disabling JMX data collection
- Enabling/Disabling Exception Seeker
- Enabling/Disabling Leak Seeker

For information about installing a remote J2EE instance, see the *Installing the J2EE remote instance* in the *Precise CLI Utilities Reference Guide*.

Time slice size

Currently 2 time slice sizes are supported: 30 seconds, 5 minutes. To change the slice size setting for a monitored JVM

1. Open the *<LOCAL-LIB>*/products/j2ee/config/*<JVM-NAME>*/collector.xml file for editing.
2. Edit the value of the <agginterval> node and specify one of the following supported values in seconds: **30** or **300**.

For example: <agginterval>30</agginterval>

1. Save the file.
2. Restart the JVM for the changes to take effect.

Using Web patterns

Perform the following steps to apply URL patterns defined for the Web instance. To define a pattern

1. Make sure Precise for Web is installed on the server of the remotely monitored JVM.

2. Copy the `TacPaterns.xml` file from `products/j2ee/config/` on the Precise FocalPoint server to `products/j2ee/config/` folder inside the extracted ZIP folder on the JVM side. If the file does not exist, restart the Precise FocalPoint.
3. Restart the JVM to start using the patterns defined by the `TacPaterns.xml` file.

Enabling/Disabling JMX data collection

To enable/disable JMX data collection, choose the corresponding the Application Server vendor below and perform the procedure described for the vendor.

Tomcat <VERSION>

Enable the Application Server Metrics

To enable Precise for J2EE to collect Application Server Metrics, a startup class must be loaded and executed on the server being monitored.

To configure the startup class

1. Copy the *<PRECISE_HOME>*/products/j2ee/lib/indepthmetric.jar file into the `<TOMCAT_HOME>/lib` folder.
2. Add the following xml entry to the `<TOMCAT_HOME>/conf/server.xml` file just below the other <Listener> entries: <Listener className="com.precise.javaperf.extensions.tomcat.JMXMetricsLoaderListener"/>

Choose a default list of metrics

Copy the *<PRECISE_HOME>*/products/j2ee/etc/metrics/TOMCAT/*<VERSION>*/measurements.xml file to the *<PRECISE_HOME>*/products/j2ee/config/*<JVM_ID>* folder.

WebLogic <VERSION>

Choose a default list of metrics

Copy the *<PRECISE_HOME>*/products/j2ee/etc/metrics/WEBLOGIC/*<VERSION>*/measurements.xml file to the *<PRECISE_HOME>*/products/j2ee/config/*<JVM_ID>* folder.

WebSphere <VERSION>

Choose default list of metrics.

Copy the *<PRECISE_HOME>*/products/j2ee/etc/metrics/WEBSPHERE/*<VERSION>*/measurements.xml file to the *<PRECISE_HOME>*/products/j2ee/config/*<JVM_ID>* folder.

Jboss <VERSION>

Enable the Application Server Metrics

To enable Precise for J2EE to collect Application Server Metrics, a startup class must be loaded and executed on the server being monitored.

To configure the start up class

1. Copy the *<PRECISE_HOME>*/products/j2ee/etc/metrics/JBOSS/*<VERSION>*/Precise for J2EE-startupclass.sar file into the *<JBOSS_HOME>*/server/*<YOUR_SERVER>*/deploy folder.
2. Choose a default list of metrics

Copy the *<PRECISE_HOME>*/products/j2ee/etc/metrics/JBOSS/*<VERSION>*/measurements.xml file to the *<PRECISE_HOME>*/products/j2ee/config/*<JVM_ID>* folder.

Enabling/Disabling Exception Seeker

To enable the Exception Seeker feature

1. Open the <LOCAL-LIB>/products/j2ee/config/<JVM-NAME>/InstrumenterConfigList.xml file for editing.
2. Add the following XML node under the root node:
   ```
   <config-file>
     ExceptionSeeker.xml
   </config-file>
   ```
3. Restart the JVM.

To disable the Exception Seeker feature

1. Delete this node inside the `InstrumenterConfigList.xml`.
2. Restart the JVM.

Enabling/Disabling Leak Seeker

To enable the Leak Seeker feature

1. Open the *<LOCAL-LIB>*/products/j2ee/config/*<JVM-NAME>*/InstrumenterConfigList.xml file for editing.
2. Add the following XML node under the root node:
   ```
   <config-file>
     LeakSeeker.xml
   </config-file>
   ```
3. Restart the JVM.

To disable the Leak Seeker feature

1. Delete this node inside the `InstrumenterConfigList.xml` file.
2. Restart the JVM.

Configuring Precise for Web

This section includes the following topics:

About configuring Precise for Web Collectors

To configure the Precise for Web Collectors, you can perform one of the following tasks:

1. Add and change parameters to the Precise registry file.
2. Add parameters to the Web server's configuration mechanism.
3. Add and change parameters to the dynamic instrumentation agent via the `instrument.xml` file.

Configuring registry settings

This section describes the registry settings configuration process.

Registry structure and inheritance

Some configurations are relevant per instance (e.g. collect-post-parameters) and some are relevant for the Web Collector in general (e.g. preferred-data-port). Others are relevant for the whole product (e.g. user-agents). Instance configuration resides in the instance registry file:

products\i3fp\registry\instances\www*<server-name>**<instance_name>*\settings.xml

> ⓘ You can set instance configuration in the `products\i3fp\registry\instances\www\<server-name>\settings.xml` file if you want to set the same setting for all instances on the machine. Consider however that settings in the instance registry will override the settings in the server registry.

Web Collector configuration resides in the registry file:

products\i3fp\registry\instances\www*<server-name>*\settings.xml

> ⓘ You can set the configuration for all Web Collectors at once by modifying the file `products\i3fp\registry\products\indepth-web\focal-point.xml` Consider however that settings in the server registry will override the settings in the `focal-point.xml` file.

Product level configuration resides in the registry file: `products\i3fp\registry\products\indepth-web\focal-point.xml` Web Collector behavior is set via the registry entry <web-collector>

Cluster configuration resides in the registry file:

products\i3fp\registry\clusters\www\<cluster-name>\settings.xml

Instance registry

Configuring the instance registry only affects the specific instance. Every element in the instance registry can also be specified in the server registry file, in which case it affects all the instances on that server.

To change instance registry

1. On the relevant Precise node (proxy), manually change the products\i3fp\registry\instances\www**<server-name>****<instance_name>**\settings.xml file.
2. Save the file.
3. Run the `update-main-registry` CLI command on the same Precise node.
   - **UNIX**. /infra/bin/psin_cli.sh -action update-main-registry -i3-user admin -i3-clear-password admin -registry-path /registry/clusters/www/*<server_name>*/*<instance_name>*
   - **Windows**. infra\bin\psin_cli.bat -action update-main-registry -i3-user admin -i3-clear-password admin -registry-path /registry/clusters/www/*<server_name>*/*<instance_name>*

This will force a registry sync from the node's FocalPoint to the main FocalPoint.

Cluster registry

Configuring the cluster registry affects all instances of the specific cluster.

> ⓘ All instances in the cluster need to be restarted via AdminPoint after changing the cluster registry.

To change the cluster registry:

1. On the relevant Precise node (proxy), manually change the products\i3fp\registry\clusters\www*<cluster-name>*\settings.xml file.
2. Save the file.
3. Run the `update-main-registry` CLI command on the same Precise node.
   - **UNIX**. ./infra/bin/psin_cli.sh -action update-main-registry -i3-user admin -i3-clear-password admin -registry-path /registry/clusters/www/*<cluster_name>*
   - **Windows**. infra\bin\psin_cli.bat -action update-main-registry -i3-user admin -i3-clear-password admin -registry-path /registry/clusters/www/*<cluster_name>*

This will force a registry sync from the node's FocalPoint to the main FocalPoint.

> ⓘ Cluster registry changes override the FocalPoint registry changes.

connection-params (in general section)

The following table describes the registry elements for the connection-params (in the general section).

Table 1 connection-params (in general section)

Registry element	Description
site-name	Specifies the name of the Web application (context root). Only relevant for J2EE Web servers and only when the filter is not installed on the root Web application.

Registry element	Description
filter-connect-ext-cmd-line	Additional miscellaneous command line parameters for the psi_web_filter_connect utility which is used to send management URLs to the filter.
authorized-ips	Comma separated IPs list that specifies the IPs from which the Precise for Web agent management pages can be browsed. Other IPs that will try to access this status will be blocked. (127.0.0.1 is added by default).

ssl (in connection-params, in general section)

The ssl element in the registry is a "container" element that contains the use-client-certificate, cipher-list, and log-file elements. It also contains another "container" element: client-certificate. This "container" element contains the path, password, and key-path elements.

The following table describes the registry elements that can be used for the ssl instance registry.

Table 2 ssl elements

Registry element	Child element	Description
use-client-certificate		Specifies the use of a client certificate when Precise for Web tries to connect using Secure Socket Layer (SSL). Default value: false.
cipher-list		Specifies additional ciphers that OpenSSL uses on top of the default ciphers for SSL version 2 and 3 and Transport Layer Security (TSL) version 1. For more information, see the following URL: http://www.openssl.org/docs/apps/ciphers.html#cipher_suite_names.
log-file		Specifies the file that OpenSSL logging is written to.
client-certificate		Specifies client certificate details, if the monitored Web server requires it. Container for path, password, and key-path elements.
	path	Specifies the full path of the client certificate.
	password	Specifies the password to be used to decrypt the certificate file. The password itself should be encrypted using the Encrypt command that is part of the Precise CLI Utility. For more information, see the *Precise CLI Utility Reference Guide*.

Registry element	Child element	Description
	key-path	Specifies the path to the private key if it is different than the path to the certificate.

user-authentication (in connection-params, in general section)

The following table describes the registry elements for the user-authentication (in connection-params, in general section).

Table 3 user-authentication (in connection-params, in general section)

Registry element	Description
user	Specifies the user name for authentication.
password	Specifies the user password for authentication. The password itself should be encrypted using the Encrypt command that is part of the Precise CLI Utility. For more information, see the *Precise CLI Utility Reference Guide*.
domain	Specifies the domain name for authentication.
method	The authentication method. One of the following methods should be used: • basic • digest • ntlm

Server-side

The following table describes the registry elements for the server-side .

Table 4 Registry elements for server-side

Registry element	Description
ext-list	Specifies the extensions used, such as .html, .jsp, and .asp. For example: <ext-list>html,jsp,asp</ext-list> Default values: html, swe, asp, aspx, htm, jsp, php, gbl, cfm, pl, php3, do, IScript_PT_NAV_PAGELET, I_Script_AppHP, IScript_UniHeader_Frame, IScript_PT_NAV_INFRAME, iscript_apphp, iscript_timeoutwarning
ext-list-separator	Specifies the separator of the extension list. Default value: , (a comma)

Registry element	Description
stat-extension	Specifies the URL of the statistics module. Add this registry element only if you know that the statistics URL differs from the default URL for the specific Web server type. Default values: Apache: server-status?auto iPlanet: .perf WebLogic: weblogicstat WebSphere: websphere_stats
filter-trace	Specifies whether filter logging is used. Default value: false
filter-session-cookie	Specifies the name of the session cookie used by the application. Define this value to improve client-side correlation in the following Web servers: WebLogic, WebSphere, Tomcat, Oracle AS, Sun One, IIS, Apache 2.x. Default value: JSESSIONID
max-i4w-cookie-size	Specifies the maximum size of a Precise for Web cookie (in bytes). Default value: 2048
filter-ignore-files	A comma-separated list of files that the filter has to ignore. If a file contains any of these patterns, it is ignored. Only relevant for J2EE filters.
filter-use-encoding	Specifies the translation of URLs and URL parameters to unicode format. Set this parameter if URLs or parameters may contain non-ASCII characters. Default value: false
filter-encoding-name	Specifies the name of the URL encoding mechanism. Set this parameter if the server handles URLs that use neither UTF-8 nor local encoding, which are the default encoding types.
web-parameter-delimiters	A list of URL parameters delimiters. These delimiters are used to separate the parameters of a URL from the URL itself. For example, the URL: "index.html?param1=abc" has 1 parameter 'param1'. The parameters delimiter in this URL is '?'. Use this element if your application uses non-standard delimiters (like ';'). To specify more than one delimiter, concatenate all delimiters into one string, for example: <parameter-delimiters>?;.</parameter-delimiters> sets 3 delimiters: '?', ';' & '.' If more than 1 delimiter is used, the first one that is encountered in the URL is used. Default value: ? (question mark)

Registry element	Description
collect-post-parameters	If this registry element is set to true, the filter collects parameters that are passed in the request body (as opposed to parameters that are passed with the URL). For enabling the post-parameters collection on an IIS Web server, another action is needed (besides changing this flag) - For more information, see the *Adding post-parameters collection for IIS6* and *Adding post-parameters collection for IIS7* sections in the Installing Web Tier Collectors section of the *Precise Installation Guide*. Default value: false (for Siebel this is set to "true" (automatically)) ⓘ This feature is not supported for Apache 1.3 ⓘ Only Form post-parameters are supported. This means: Only submitted page forms of which the POST request contains the header `Content-Type: application/x-www-form-urlencoded` are supported. The post-parameters are passed as: `key1-value1&key2=value2&...` Any other type of post-parameters are not monitored. For example - if the POST data is XML. ⓘ In any case no more than 1500 characters of POST data will be collected.
apache-pid-file-path	Sets the path for apache .pid file (commonly: httpd.pid). Should be set only when the .pid file path is not specified in the configuration file or in any other case, when the statistics agent fails to resolve it.

user-defined-transaction-name

By configuring the elements user-defined-transaction-name, you can change the collection method of the transactions. This is valid for Precise for Web and Precise TPM. The collection method can be represented by one of the following tags:

Table 5 Elements

Elements	Description
cookie-name	Collects the value of the specified cookie.
req-header-name	Collects the value of the specified request header.
parameter-name	Collects the value of the specified GET parameter (not relevant for POST parameters).

Elements	Description
dom-element	Collects the value of the specified dom element. Applicable only if the client.-side agent is installed.
display-mode	The way the transaction name is displayed in Precise for Web and Precise TPM.

When configuring a user defined transaction name, you will need to specify the display mode between the <display-mode> tags. Possible values are:

- **prefix**. This will place the value in the before the original transaction name.
- **override**. This will replace the original transaction name with the configure transaction name.

> (i) When you configure a user defined transaction name, verify that it is a field with little variance (not many different values). For example, the cookie APP_VIEW is a good candidate to be used. In contrary to the SESSION_ID, which is a bad candidate, as it has a very large range of values. When you have a very large range of values, the Precise for Web schema tables grow very large.

To add a cookie - example

- If you have a cookie called USER_NAME in your application, you may add it to the user interface (as User Name), by adding the following UserDefined element:

  <user-defined-transaction-name>
      <cookie-name>USER_NAME</cookie-name>
      <display-mode>prefix</display-mode>
  </user-defined-transaction-name>

To add a dom element - example

- If you want to add the Title (DOM element) of your Web pages to the user interface (as My Title), you can add the following UserDefined element:

  <user-defined-transaction-name>
      <dom-element>document.title</dom-element>
      <display-mode>prefix</display-mode>
  </user-defined-transaction-name>

If the client-side agent is not installed on your Web server (only server-side is installed), you can not collect the Title DOM element.

user-defined-client-ip

By configuring the element user-defined-client-ip, you can change the way that client IP is collected by the server-side agent. You can instruct Precise for Web to collect the client IP from 3 different sources (instead of the HTTP request itself):

- Cookie
- Request header
- URL parameter

> (i) You may want to change the way the client IP is collected when the Web server is behind some proxy, in which case the collected client IP is always the IP of the proxy.

apache-virtual-hosts

To separate Apache Virtual Hosts in different Precise applications, the Precise administrator needs to perform the following steps:

1. Create an Application in Precise for each Virtual Host. (For more information, see the Working with the *Application Installer wizard* section in the *Precise Installation Guide*.)
2. Place the Apache instance in the multiple instance view in each of the defined Precise applications.
3. Go to the instance registry and set the a registry entry for each of the Virtual Hosts using the following format:
   ```
   <virtual-host-settings>
      <vhost-i>
         <vhost-name>name</vhost-name>
         <precise-application-name>application_name</precise-application-name>
      </vhost-i>
      …
      <!-- map all unmapped Virtual Hosts to this env (if not defined the information will be discarded) -->
      <vhost-i>
         <vhost-name>default</vhost-name>
         <precise-application-name>application_name</precise-application-name>
      </vhost-i>
   <virtual-host-settings>
   ```

   > (i) In a case of an unmapped Virtual Host, transactions can either be displayed in the "default" application ("default" is set), or discarded if "default" is not set.

4. Restart the Web Collector on the monitored server by performing the following steps.
   a. Go to **AdminPoint>Agents**.
   b. Click **Stop** to stop all Web agents on the monitored server.
   c. Click **Start** to start all Web agents on the monitored server.

After performing these steps, you will see the unique information of each Virtual Host in each of the defined Precise applications.

The limitations of this feature are:

- Once the instance is configured as defined, each of the Virtual Hosts are linked to a specific application in Precise and the information cannot be shared in a multiple instance view.
- J2ee instances which are shared between two applications can cause unexpected results, and thus should be defined in two Virtual Host Precise applications.

Server registry

Configuring the server registry affects the Web collector and the instances on the server.

server-side

Table 6 client-side section

Registry elements	Description
use-tomcat-client-collector-as-proxy	If set to true, the instance will sent client-side data to the client collector. Default: true for Apache 1.3 and iPlanet false for the rest.

Table 7 verify section (inside web-collector)

Registry elements	Description
max-user-agent-length	Maximum allowed user agent (browser type) length. Records exceeding this value will be ignored (not loaded). Units: bytes Default: 256
max-user-defined-length	Maximum allowed user defined 1 & 2 length. Records exceeding this value will be ignored (not loaded). Units: bytes Default: 256

Table 8 server-side section (inside verify, inside web-collector)

Registry elements	Description
max-server-period	Maximum allowed server period (not including network period). Records exceeding this value will be ignored (not loaded). Units: milliseconds Default: 3600000 (1 hour)
max-future-time	Maximum allowed difference between current time and future record time. Records that have a later time the allowed will be ignored (not loaded). Units: milliseconds Default: 86400000 (1 day)
empty-http-status-allowed	If true, records with no HTTP status are not ignored. Default: true
fallback-http-status	If a record does not contain HTTP status and empty-http-status-allowed is set to true, sets the value to use for the HTTP status of that record. Default: 0
min-http-status, max-http-status	If fallback-http-status is not set (or set to 0) HTTP statuses outside this range cause the record to be ignored (not loaded). Default: 0, 599
empty-protocol-allowed	If true, records with no HTTP protocol are not ignored. Default: true

Registry elements	Description
max-protocol-length	If the length of the HTTP protocol of a record exceeds this value, the record will be ignored (not loaded). Units: bytes Default: 5
allowed-protocols	If this is set, only HTTP protocols in this list are allowed, other values will cause the record to be ignored (not loaded). Default: none
empty-http-method-allowed	If true, records with no HTTP method are not ignored. Default: true
fallback-http-method	If a record does not contain HTTP method and empty-http-method-allowed is set to true, sets the value to use for the HTTP method of that record. Default: GET
allowed-http-methods	If this is set, only HTTP methods in this list are allowed, other values will cause the record to be ignored (not loaded). Default: none

Table 9 client-side section (inside verify, inside web-collector)

Registry elements	Description
max-text-period	Maximum allowed text period. Records exceeding this value will be ignored (not loaded). Units: milliseconds Default: 900000 (15 minutes)
max-rendering-period	Maximum allowed rendering period. Records exceeding this value will be ignored (not loaded). Units: milliseconds Default: 900000 (15 minutes)

The entries below are similar for 4 sections (with one exception*):

Table 10 summed section (inside server-side, inside output, inside web-collector) raw section (inside server-side, inside output, inside web-collector), summed section (inside client-side, inside output, inside web-collector), raw section (inside client -side, inside output, inside web-collector)

Registry elements	Description
max-files	Maximum files allowed in the output directory. After exceeding this value, no more files will not be written. Default: 1440
max-recs-in-file	Maximum records allowed in 1 output file. After exceeding this value, a new file will be created. Default: 10000
max-time-to-write-file	Maximum time to have 1 file open. Units: milliseconds * Default for summed: 900000 (15 minutes) *Default for raw: 60000 (1 minute)
max-file-size	Set the maximum for file size. Units: bytes Default: 10485760 (10MB)

Table 11 server-side section (inside tac, inside smartlink, inside web-collector), client-side section (inside tac, inside smartlink, inside web-collector)

Registry elements	Description
max-files	Maximum files allowed in the output directory. After exceeding this value, no more files will not be written. Default: 1440
max-recs-in-file	Maximum records allowed in 1 output file. After exceeding this value, a new file will be created. Default: 10000
max-file-size	Set the maximum for file size. Units: bytes Default: 10485760 (10MB)

Table 12 server-side section (inside input, inside web-collector), client-side section (inside input, inside web-collector)

Registry elements	Description
delete-files	When set to true, input raw files are deleted after being processed. Default: true
max-error-files	Maximum number of error files to move to the errors directory when errors occur on files. Default: 10

Table 13 statistics section (inside ivp, inside web-collector)

Registry elements	Description
enabled	When set to false, validation of the statistics component of the web collector are not performed in the verify action (in AdminPoint). Default: true

Table 14 processing section (inside ivp, inside web-collector)

Registry elements	Description
Enabled	When set to false, validation of the core web collector are not performed in the verify action (in AdminPoint). Default: true
server-amount-limit-for-no-client	When client/server correlation is enabled, the web collector warns - in the verify action (in AdminPoint) – that server records were received but no client records did. This number sets the threshold for this warning. Default: 1000
client-amount-limit-for-no-server	When client/server correlation is enabled, the web collector warns - in the verify action (in AdminPoint) – that client records were received but no server records did. This number sets the threshold for this warning. Default: 500
server-amount-limit-for-no-network	When server/network correlation is enabled, the web collector warns - in the verify action (in AdminPoint) – that server records were received but no network records did. This number sets the threshold for this warning. Default: 1000

Registry elements	Description
network-amount-limit-for-no-server	When server/network correlation is enabled, the web collector warns - in the verify action (in AdminPoint) – that network records were received but no server records did. This number sets the threshold for this warning. Default: 1000

Table 15 web-collector section

Registry elements	Description
preferred-data-port	Will be used by the web collector for communication with server and network agents. If this port is busy, any available port will be used. Default: 20999

Table 16 summary section (inside web-collector)

Registry elements	Description
enabled	When set to false, summary processing is not performed by the web collector. This setting is meaningful only on the monitored server (and not on the FP machine where it is ignored). Setting this to false (along with client-server-correlation) sets the work mode to Basic Mode. Default: true

Table 17 server-side section

Registry elements	Description
Enabled	When set to false, server-side collection is disabled. Default: true
client-server-correlation	When set to false, client/server correlation is disabled and in fact the instance will work in Basic Mode (this does not have meaning on the FP machine). Default: true
server-network-correlation	When set to false, server/network correlation is disabled (this does not have meaning on the FP machine). Default: true
filter-do-collector-settings	When set to false, collector settings activity (handling of URL parameters) is done in the web collector instead of in the web filter. Default: true (processing done on filter)

Registry elements	Description
filter-do-data-patterns	When set to false, data patterns processing is done in the web collector instead of in the web filter. Default: true (processing done on filter)
filter-do-locations	When set to false, data locations processing is done in the web collector, instead of the web filter. Default: true (processing done on filter)

Table 18 client-side section

Registry elements	Description
enabled	When set to false client-side collection is disabled Default: true

Table 19 statistics section

Registry elements	Description
enabled	When set to false statistics collection is disabled Default: true
max-files (in output section)	Maximum files allowed in the output directory. After exceeding this value, no more files will not be written. Default: 2000

Table 20 processing section (inside web-collector)

Registry elements	Description
max-data-processor-threads	Maximum number of threads that will be invoked by the web collector to handle data. Default: 50
write-dp-period-ms	The initial wait time of the serialization timer that serializes the current data to files. Units: milliseconds Default: 120000

Registry elements	Description
write-dp-delay-ms	The cycle time of the serialization timer that serializes the current data to files. Units: milliseconds Default: 120000

Table 21 server-side section (inside processing, inside web-collector)

Registry elements	Description
input-queue-max-size	Set the maximum number of records in the input queue. After the queue reaches this size, we start dropping records. May be used to temporarily ease memory consumption. Default: 15000
aging-period	Max time for server records to wait in memory for correlation (with both client & network records). Units: milliseconds Default: 90000 (1.5 minutes)
aging-period-when-using-client-collector	Max time for server records to wait in memory for correlation (with both client & network records) when this instance uses the client collector. Units: milliseconds Default: 30000 (30 seconds)

Table 22 client-side section (inside processing, inside web-collector)

Registry elements	Description
input-queue-max-size	Set the maximum number of records in the input queue. After the queue reaches this size, we start dropping records. May be used to temporarily ease memory consumption. Default: 10000
aging-period	Max time for client records to wait in memory for correlation (with server records). Units: milliseconds Default: 90000 (1.5 minutes)

Registry elements	Description
aging-period-when-using-client-collector	Max time for client records to wait in memory for correlation (with server records) when this instance uses the client collector. Units: milliseconds Default: 180000 (3 minutes)
max-text-period-before-fix	Max time in milliseconds for client records first byte time. If the first byte time exceeds the maximum, it will be fixed by taking the server + network time. Default: 900000 milliseconds

Table 23 network section (inside processing, inside web-collector)

Registry elements	Description
input-queue-max-size	Set the maximum number of records in the input queue. After the queue reaches this size, we start dropping records. May be used to temporarily ease memory consumption. Default: 15000
aging-period	Max time for network records to wait in memory for correlation (with server records). Units: milliseconds Default: 90000 (1.5 minutes)

Table 24 server-side section (inside filtering, inside web-collector)

Registry elements	Description
enabled	When set to false, server-side filtering is disabled Default: true

Registry elements	Description
reset-identifiers	A list of identifiers that will be reset in the filtering-summary process (when needed). The identifiers will be reset, one by one, in the order they appear here until the top N limit is reached. Valid values: • url • domain • userAgent • userDefined1 • userDefined2 • appUserName • clientIP • location • city • state • country • siebelView • urlParams • protocol • httpMethod • sessionID Default: httpMethod, userAgent, protocol

Table 25 client-side section (inside filtering, inside web-collector)

Registry elements	Description
enabled	When set to false, client-side filtering is disabled Default: true

Registry elements	Description
reset-identifiers	A list of identifiers that will be reset in the filtering-summary process (when needed). The identifiers will be reset, one by one, in the order they appear here until the top N limit is reached. Valid values: • url • domain • protocol • userAgent • userDefined1 • userDefined2 • appUserName • clientIP • location • city • state • country • siebelView • title • host • connectionType • privateIP • login Default: userAgent, protocol

FocalPoint registry

The following section describes the FocalPoint registry element.

user-agents

The user-agent element consists of a set of definitions of user agents. A user agent is a device that may access a Web server. A typical user agent is a browser like Microsoft Internet Explorer or Firefox. Other types of user agents are automatic HTTP traffic generators, like Precise Insight Inquire. The Precise for Web Collector agent, automatically collects the user agent of every request that it monitors. To extract a comprehensive user agent name and version from the user agent header of the request, the agent uses the definitions in the user-agents element. By default, the user-agent element contains a definition for most of the industry's known Web servers, like Microsoft Internet Explorer, Firefox, Netscape, Mozilla as well as the integrative Precise Insight Inquire traffic generator. However, you can add any number of user agents to the list, or change the definition of an existing user agent to adjust it to customers private needs or conventions. For example, the definition for the Netscape browser is:

```
<user-agent id="Netscape" display-string="Netscape version">
    <version-search-string>Netscape</version-search-string>
    <search-string value="Netscape"/>
</user-agent>
```

The default user agents that the product supports out-of-the-box are:

• Precise Insight Inquire

- Microsoft Internet Explorer
- Firefox
- Opera
- Netscape
- Mozilla
- Konqueror
- iCab
- Safari

The following table describes the registry elements for the user-agent in the FocalPoint registry.

Table 26 Registry elements

Registry elements	Description
user-agent	A user agent definition that Precise for Web uses to find and describe user agents.
user-agent.id	A string identifier for the user agent. This identifier is not the display name.
user-agent.display-string	The display string that is shown in the user interface for this user agent. You can use the *version* token which is replaced with the detected user agent version.

Adding Web filter parameters

If you use an Apache, iPlanet, Sun ONE, WebSphere, Tomcat, Oracle Application Server, SAP, J2EE, or BEA WebLogic server, you can specify some Web filter settings that the Web server loads on startup. These parameters are the same for all Web servers. To add them, follow the instructions for the specific Web server. For more information, see Addable parameters.

> (i) Modification of IIS Web filter parameters is not yet supported.

To add parameters on an Apache server

1. Open the httpd.conf file.
2. After the LoadModule pssfilter_module line, add the parameter.

For example:

LoadModule pssfilter_module... PssFilterTrace 1

To add parameters on an iPlanet server

1. Open the magnus.conf files.
2. At the end of the Init fn=pss-init line, add the parameter.

For example:

Init fn=pss-init PssFilterTrace="1"

To add parameters on any J2EE server by editing the web.xml file

1. Open the web.xml file.
2. Add the parameters to the <filter> element in the following format:

```
<init-param>
<param-name>xxx</param-name>
<param-value>yyy</param-value>
</init-param>
```

For example:

```
<filter>
<filter-name>pssFilter</filter-name>
  <description></description>
  <filter-class>com.precise.ifweb.Collection.Filters.pssFilter
  .WLFilter</filter-class>
  <init-param>
    <param-name>PssFilterTrace</param-name>
   <param-value>1</param-value>
  </init-param>
</filter>
```

Addable parameters

The following table lists all parameters that you can add to the Web filters. This mechanism should only be used for the instance name and `<i3_root>`. Any other parameter should only be added after consultation with the Precise Enterprise Support Team.

Table 27 Addable Web filter parameters

Parameter	Description
PssInstanceName	Specifies the name of the filter's instance.
PssInstanceID	The instance ID as specified in the infrastructure database.
PssI3Root	Specifies the path to the Precise installation directory.
PssReadConfFile	Specifies whether to read the configuration files on filter initialization. If this parameter is set to FALSE, parameters are taken from the Web server only. Default: TRUE
PssRestartFilesPath	Specifies an alternate location for the pss_restart file. By default this file is located under the directories of the Web server itself (the exact location is web server type dependent). Set this to a different path if the file cannot be created at the default location.
PssInitLog	Specifies whether to write initialization log messages to standard output. In some cases the standard output is written to the Web server log. Applies only to J2EE-based application servers (Sun ONE, WebSphere, Tomcat, Oracle Application Server, SAP J2EE, or BEA WebLogic)

The following table gives a comparison of the parameter and its equivalent registry element.

Table 28 Parameter and Registry element

Parameter	Registry element
PssFilterTrace	filter-trace
PssIgnoreFiles	filter-ignore-files
PssSessionCookieName	filter-session-cookie
PssMaxI4WCookieSize	max-i4w-cookie-size
PssCorrelationMode	web-disable-correlation ⓘ The opposite of this registry element works as its equivalent parameter.
PssExtList	ext-list
PssExtListSeparator	ext-list-separator
pcs_shm_dir	general-pcs-shm-dir
PssURLEncodingName	filter-encoding-name
PssUseURLEncoding	filter-use-encoding

Grouping consecutive URLs by pattern

The Activity tab of Precise for Web displays information on the URLs and pages visited. In some cases, the database tables that contain this information can easily swell and harm the performance of the user interface, for example when URLs are generated automatically.

To reduce the accumulation frequency of URLs, Precise for Web lets you group consecutive URLs by pattern by editing the data-patterns registry element. Each of the data-pattern elements (urls, pages, titles, domains, and user-defined-transaction-name) can be specified with methods. The available methods are prefix, suffix, prefix-by-key, and regex.

You can add as many methods and patterns to any of the sections (<urls>, <pages>, <titles>, <domains>, <user-defined-transaction-name>).

> ⓘ The tags <methodN> and <patternN> should be numbered so that 'N' is replaced with a consecutive number.

> ⓘ In URLs and Pages, pattern definitions only work on the path part. For example, in the URL `/index.jsp?id=&eventid=`, the path part is `/index.jsp` and the parameters part is `id=&eventid=`. To handle collection definitions (aka grouping) on the parameters part, refer to the Collector Settings section. In the above example, any pattern that is defined on the URL, ignores the parameters part (`id=&eventid=`).

The following table describes the available methods.

Table 29 methods

Method	Description
prefix	Take only the defined prefix. For example: <pattern1>abraham</pattern1> - For URL 'abraham123.html' you will see 'abraham%'.
suffix	Take only the defined suffix. For example: <pattern1>html</pattern1> - For URL 'abraham123.html' you will see '%html'.
prefix-by-key	Take text up to the defined key (not including that key). For example: <pattern1>;jsessionid</pattern1> - For URL 'abraham123.html;jsessionid' you will see 'abraham123.html%' (if showReplaceChar="false", you will get 'abraham123.html').
regex	Use the given regular expression to change the given URL. For example: <pattern1 replace="$1">service(.*)</pattern1> - For URL 'service123' you will get '123' (for more on regular expressions, consult your support representative).

The following code is an example of the element's content for an URL element when all the grouping methods are activated:

```
<data-patterns>
   <urls>
      <execute>true</execute>
      <methods>
         <method1>
            <name>prefix</name>
            <patterns>
               <pattern1>/redirectlogon.html</pattern1>
            </patterns>
         </method1>
         <method2>
            <name>suffix</name>
            <patterns>
               <pattern2>ads.com</pattern2>
            </patterns>
         </method2>
         <method3>
            <name>prefix-by-key</name>
            <patterns>
               <pattern3>:jsessionid=</pattern3>
            </patterns>
         </method3>
```

```
        <method4>
            <name>regex</name>
            <patterns>
                <pattern4 replace="$1">\d+: (.*)</pattern4>
            <patterns/>
        </method4>
     </methods>
   </urls>
</data-patterns>
```

When URL pattern prefix grouping is activated, a percentage sign (%) is added at the end of a URL. For example, the URL /redirectlogon.html/121231245 is saved in the database as /redirectlogon.html%.

To group consecutive URLs by pattern

1. Open the relevant registry file (instance registry, server registry, or focal-point registry) and find the data-patterns element (If it does not exist, you can add it as appears in the server registry)
2. Edit the data-patterns element in the following way:
   - Set the required <execute> elements to **true**.
   - Choose the method (prefix, suffix, prefix-by-key, or regex) in the required element (urls, pages, titles, domains).
   - Define the URL patterns in the <pattern> elements. You can include as many <pattern> elements as required.
   - If you choose the regex method, use the replacement string. For example:
     <pattern replace="$1">\d+: (.*)</pattern>

3. Save and close the file.
4. Restart the Precise for Web server-side and dynamic instrumentation agents monitoring the relevant environment.

Temporarily stopping the maximum number of table rows test

If the database tables that contain the URLs and pages that are visited exceed the maximum number of rows, the installation verification procedure (IVP) issues an error message. To prevent this message from reappearing until the accumulation frequency of URLs has been lowered, you can temporarily stop the 'maximum number of table rows' test.

To temporarily stop the 'maximum number of table rows' test

1. Open the *<i3_root>*\products\www\ivp\resources\functions_ivp.xml file in an editor:
2. In the file, set the <max-rows-for-alert> element to **0**.
3. Save and close the file.

Dynamic instrumentation

When you install a Web server agent, you are asked whether you would like to dynamically instrument Web pages. Also when you select an existing instance, you can edit it, so that the dynamic instrumentation can be applied. In both cases this results in an additional entry in the AdminPoint window. The new entry shows identical instance information for all but one column: the Agent column. In this column the agent is shown as Web Instrument agent. The Instrument.xml file handles the dynamic instrumentation configuration.

About the instrumentation configuration file (Instrument.xml)

If the instance is part of a cluster, the Instrument.xml file is located in the *<FP server>*\products\i3fp\registry\clusters\www*<name of cluster>* directory.

If the instance is not part of a cluster, the `Instrument.xml` file is located in the `<FP server>\products\i3fp\registry\instances\www\<monitored instance name>\<instance name>\<instance name>` directory. If you make changes in the `Instrument.xml` file, then you will also have to make the changes in the proxy (or proxies). The recommended way to do so it you use the CLI command as describe before. See Instance registry.

Following is an example of the file structure that is explained in the section following the example:

```
<dynamic-inst>
    <enable>true</enable>
    <handle-js>true</handle-js>
    <prolog-string> <![CDATA[<!DOCTYPE html PUBLIC "-//W3C//DTD
    XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/ xhtml1-transitional.dtd">]]>
    </prolog-string>
    <no-recompress>true</no-recompress>
    <accumulate-on-parsing>true</accumulate-on-parsing>
    <max-parsing-bytes>2048</max-parsing-bytes>
    <js-mime-type>text/javascript</js-mime-type>
    <instance-id>1234</instance-id>
    <strict-servlet-api>true</strict-servlet-api>
    <private-log>false</private-log>
    <i3-log>false</i3-log>
<webgarden>
    <enable>false</enable>
    <sampling-interval>2000</sampling-interval>
</webgarden>
<user-agents>
        <user-agent name="debug">
            <i3-log>false</i3-log>
            <private-log>false</private-log>
            <accumulation>default</accumulation>
            <parsing>false</parsing>
            <second-script>false</second-script>
            <enable>true</enable>
        </user-agent>
        <user-agent name="MSIE">
            <i3-log>false</i3-log>
            <private-log>false</private-log>
            <accumulation>default</accumulation>
            <parsing>false</parsing>
            <second-script>false</second-script>
            <enable>true</enable>
        </user-agent>
        <user-agent name="Default">
            <i3-log>false</i3-log>
            <private-log>false</private-log>
            <accumulation>default</accumulation>
            <parsing>false</parsing>
            <second-script>false</second-script>
            <enable>true</enable>
        </user-agent>
</user-agents>
<include-ip>
    <range from="123.23.1.1" to="123.23.1.255" />
```

```
        <range from="125.2.1.1" to="125.2.1.255" />
    </include-ip>
    <url-exclude>
        <suffix>.gif</suffix>
        <suffix>.jpeg</suffix>
    </url-exclude>
    <url-include>
        <prefix>/products</prefix>
        <prefix>/petstore/accounts</prefix>
        <suffix>createData.dll</suffix>
    </url-include>
    <include-char-encoding>
        <char-enc>shift_jis</char-enc>
        <char-enc>jis</char-enc>
    </include-char-encoding>
    <req-header-exclude>
        <header name="content-type" value="application/x-www-form- urlencoded" />
    </req-header-exclude>
</dynamic-inst>
```

About the Instrument.xml tags

The following table contains the tags that can be encountered in the Instrument.xml file. "not applicable" means that the tag has no influence on that specifically chosen instance type.

Table 30 Instrument.xml tags

Tag	Description
dynamic-inst	The first tag of the dynamic instrumentation configuration file. J2EE: mandatory ISAPI: mandatory
enable	Enables or disables the instrumentation filter globally. J2EE: optional ISAPI: optional Values: true, false Default value: true
handle-js*	Enables or disables the JavaScript handling by the filter. J2EE: optional ISAPI: optional Values: true, false Default value: true

Tag	Description
strict-servlet-api (J2EE)	Forces compatibility with the J2EE servlets specifications. J2EE: optional ISAPI: not applicable Values: true, false Default value: true
js-mime-type	Uses js-mime-type protocol. J2EE: optional ISAPI: optional Values: if possible to use dynamically obtained, otherwise use application/x-javascript Default value: if possible use dynamically obtained, otherwise use application/x-javascript
instance-id*	The filter init process fills this tag: • If the tag exists and is empty, it is filled with the right instance-id. • If the tag exists with a value, the value is not changed. • If the tag does not exist, it is not created. • If the filter is able to read the value of the tag, it inserts it to a cookie on the response. ⓘ The tag should only have an empty value on load balanced environments with no server side installed. Default value: empty
prolog-string*	An extra string that is injected during instrumentation (in addition to the regular callout to JavaScript). Usually it is used to add the <! Doctype element (because according to some browsers this tag should come at the beginning of the Web page). J2EE: optional ISAPI: optional
no-recompress	For compressed content. If set to true, insert the instrumentation script without recompression. Apache: applicable for version 2.0 and higher Values: true, false Default value: true

Tag	Description
accumulate-on-parsing	If parsing is on, accumulate response buffers till the <html> or <head> tag is found (see also the note after this table). J2EE: optional ISAPI: optional Apache: applicable for version 2.0 and higher Values: true, false Default value: true
max-parsing-bytes	Looks for the <html> tag or the <head> tag only in the first n bytes specified. J2EE: optional ISAPI: optional Apache: applicable for version 2.0 and higher Values: number of bytes, -1 for full parsing Default value: 2048
private-log	Controls the writing of messages that need to be logged before the user-agent is obtained, to the private log. J2EE: optional ISAPI: optional Values: true, false Default value: false
i3-log	Enables the i3 log globally. J2EE: not applicable ISAPI: not applicable Values: true, false Default value: false
webgarden*	Introduces the Webgarden management section.
enable	Enables or disables "Webgarden" support. Values: true, false Default value: true for IIS and SAP WAS Web servers, false on any other Web server

Tag	Description
sampling-interval	Setting the sampling interval (in milliseconds) in which each process in the "Webgarden" updates its status. Value: any number higher than 500 Default value: 2000
user-agents	Introduces the user-agents section. J2EE: mandatory ISAPI: mandatory
user-agent	Describes the configuration per user-agent. J2EE: mandatory, only for user-agent with name "Default" ISAPI: mandatory, only for user-agent with name "Default"
accumulation	Controls whether to perform accumulation of chunks/parts of the Web content and then preforming instrumentation all together. The default is not to use it and perform instrumentation on-the-fly (and perform accumulation only if necessary). There is an option to force accumulation (true) or to disable instrumentation when accumulation must be done (false). J2EE: mandatory ISAPI: mandatory Values: default, true, false Default value: default
parsing	Controls whether to perform light-parsing to inject the first script after <html> or <head> (what is recognized first). "false" means to inject the script at the beginning of the page, without parsing. J2EE: mandatory ISAPI: mandatory Values: true, false Default value: true
enable	Enables or disables the instrumentation filter per user-agent. J2EE: mandatory ISAPI: mandatory Values: true, false Default value: false

Tag	Description
second-script	Controls whether to inject a second-script at the end of the page. ⓘ This option is reserved for future use. J2EE: mandatory ISAPI: mandatory Values: true, false Default value: false
i3-log	Enables or disables the i3 log per user-agent. J2EE: mandatory ISAPI: mandatory Values: true, false Default value: false
private-log	Enables or disables the private log. J2EE: mandatory ISAPI: mandatory Values: true, false Default value: false
user-agent name	Relates to the browser user-agent that includes this value. The order in the list is important. J2EE: mandatory ISAPI: mandatory Values: Default (mandatory) or free text (such as MSIE or FireFox) that is part of the user-agent string. ⓘ Default should always be the last user in the list and the values are case-sensitive. Default value: For more information, see the following URL: http://www.zytrax.com/tech/web/browser_ids.htm

Tag	Description
include-ip	Introduces the section in which you define the IP addresses that should be included in the dynamic instrumentation process. By default all IPs are included. J2EE: optional ISAPI: optional
range from	Beginning of the IP address range. J2EE: optional ISAPI: optional Value: IP address
range to	End of the IP address range. J2EE: optional ISAPI: optional Value: IP address
url-exclude	Introduces the section in which you define the URLs that should be excluded. By default all URLs are included. J2EE: optional ISAPI: optional
url-include*	Introduces the section in which you define the URLs that should be included. By default all URLs are included. J2EE: optional ISAPI: optional
Prefix	Defines the URL prefix.
suffix*	Defines the URL suffix.
include-char-encoding	Introduces the section for additional character encoding that is supported in the filter (in addition to the built-in character support). J2EE: optional ISAPI: optional

Tag	Description
char-enc	Defines additional character encoding that is supported in the filter (in addition to the built-in character support). Use if your pages are written in none Unicode character encoding. J2EE: optional ISAPI: optional
req-header-exclude	Introduces the section that defines the HTTP request headers for which HTTP requests is excluded. J2EE: optional ISAPI: optional
header name	Defines the HTTP header name.
header value	Defines the specific HTTP header value or "*".

The tags that are marked with an asterisk (*) appear in the Frequently Asked Questions document.

> ⓘ The terms "accumulation", "parsing (without accumulation)", and "accumulate on parsing" can be described as follows:
>
> - **Accumulation**. Keep all response buffers till you have all the document and if parsing is on, only then parse the document and instrument if the <html> or <head> tag is found.
> - **Parsing (without accumulation)**. On the fly check each response buffer and find the <html> or <head> tag. Once found - instrument.
> - **Accumulate on parsing**. Keep response buffers only until the on the fly parsing finds the <html> or <head> tag or till the max-parsing-bytes is reached, and then release the buffers with the right content-length.

About include and exclude tags

The following list describes the order in which the include and exclude tags are handled in the instrument.xml file:

- The optional tag "enable" can disable the whole instrumentation process for all HTTP responses.
- The "include-ip" section is checked first. If there is no "include-ip" section, all IPs can be instrumented.
- The "url-include" section is checked before the "url-exclude" section.

If one "url-include item is specified, this HTTP request must be instrumented. If no "url-include" section is specified, all URLs are included.

If one or more "url-exclude" item(s) is (are) specified, the HTTP request(s) must be skipped.

- If the filter recognizes a "content-type" that does not contain "text/html", the instrumentation process is skipped for that particular HTTP response.

Configuring Precise for Oracle

This section includes the following topic:

- Updating the Oracle version

Updating the Oracle version

When you upgrade your Oracle version, You must uninstall the monitored instance(s) from Precise, and then reinstall the monitored instance(s) using the same instance ID.

1. Make sure the Warehouse Maintenance (Weekly) does not run soon.
2. To uninstall:
   a. Navigate to the AdminPoint installation.
   b. Select **Instances and Clusters**.
   c. Select the instance(s) to be uninstalled, and then click **Delete**.

3. To reinstall:
   a. Navigate to the AdminPoint installation.
   b. Select **Instances and Clusters**.
   c. Click **Add Instance**.
   d. Complete the appropriate fields, and then click **Advanced**.
   e. Click the option to install using the same instance ID (**Associate the instance with the data of the deleted instance**).

> ⓘ If the Warehouse Maintenance (Weekly) Warehouse Process is scheduled to run in between uninstalling/reinstalling, in Parameters, make sure the box to Purge data of deleted instance is not checked.

Configuring Precise for BW

This section includes the following topics:

- About Configuring SLA and Email Alert settings
- About Configuring the CCMS Alerts display
- About Configuring Performance settings

About Configuring SLA and Email Alert settings

SLA and Email Alert settings are configured in the Precise AdminPoint screen.

Setting SLA defaults

To configure your SLA settings follow these steps:

1. On the Precise AdminPoint screen, click **Settings > SLAs**.
2. In the SLA Settings dialog box, select the Service Time tab and in the Technology field, select **SAP**.
3. In the SLA Name column, you can either select an existing SLA definition or create a new one.
    a. ***If you are using an existing SLA definition***:
        i. Select the SLA definition and click the **Edit** button.
        ii. At the bottom of the displayed dialog, enter the new Near-breach and Breach thresholds (in minutes), and then click **OK**.

    b. ***If you are creating a new SLA definition***:
        i. Click the **Add** button.
        ii. In the SLA name field, enter a name for the new SLA definition.
        iii. In the Free text field, enter the name of the SAP BI Process Chain, and then click the right-arrow.
        iv. Enter the new Near-breach and Breach thresholds (in minutes), and then click **OK**.

Setting general alert defaults

To configure your general email alert settings follow these steps:

1. On the Precise AdminPoint screen, click **Settings > Alerts General Settings**. The Alerts General Settings dialog box is displayed.
2. In the Email tab, select the **Integrate Alerts with an email server** check box and enter the email server name or IP address.
3. Set the address you want to appear in the sent by field in email actions.
4. Click **OK**.

For further information refer to the About configuring Alert general settings section on page in the Precise Administration Guide.

Activating the email action alerts

To activate your email action alerts follow these steps:

1. On the Precise AdminPoint screen, click **Settings > Alerts Metric Settings**. The Alerts Metric Settings dialog box is displayed.
2. In the Settings tab, select the required environment and the SAP AppTier.
3. Select Failed Process Chains and click the **Edit** button.
4. In the Metric Properties - Edit dialog, select the Actions tab and select the email option on the Action Type list box.
5. Click the **Add** button. The Add Email Action dialog box is displayed.

6. In the When list box, select the minimum alert severity level that will cause this action to run.
7. If you want to alert higher management only after several sequential triggers, enter a value into the 'for at least <...> times' text box.
8. Set a valid email address.
9. Click **OK**.
10. Select SLA Breaching Process Chains and click the **Edit** button.
11. Repeat steps 4 through 9.

For further information refer to the About configuring Alerts metric settings section on page in the Precise Administration Guide.

About Configuring the CCMS Alerts display

This section describes the CCMS Alerts display configuration.

How to specify additional monitors to be displayed

By default, three monitors are shown in the user interface. You can add more monitors if you like.

To add CCMS Alerts monitors:

1. Specify the monitor set and its name in the following XML file:
   `<i3_root>\products\i3fp\registry\products\indepth-sap\ccmsConfigFile.xml.`
2. Save the file.

How to specify that the Alerts will come from a Central Monitoring instance

The CCMS alerts in your organization may be concentrated in a Central Monitoring instance. You can configure Precise to pull the alerts from there instead of from the BI instance itself.

To specify the Central Monitoring instance the alerts will come from:

1. Add the following tag to the `products\i3fp\registry\products\indepth-sap\ccmsConfigFile.xml` file:

   `<ccmsSourceSystem><SID></ccmsSourceSystem>`
2. Replace `<SID>` by the System ID of the Central Monitoring instance.

   > ⓘ   The Central Monitoring instance must be an instance which is sampled by Precise for SAP.

3. Restart the Precise for SAP FocalPoint.

About Configuring Performance settings

If the `High data-transfer volume` message is shown, it means that the system is processing too many Process Chain executions.

In this case, three options are available:

- Choosing a smaller 'Overall time range' setting will cause less rows to be returned, so it's possible that the message will no longer appear.
- Allow more rows to be returned by raising the perfMaxRows threshold (See How to modify the performance thresholds.). Using this option means that more rows will be processed by Precise for BW, and the user experience might be degraded.

- Filter out short-running Process Chains by raising the perfFilter threshold. Using this option guarantees that less rows will be processed by Precise for BW resulting in a faster user experience, but shorter Process Chains will not be shown.

Specifying the maximum amount of rows

About the perfMaxRows threshold – This value determines the number of rows that the system will return for the selected time frame before the `High data-transfer volume` message is shown.

To determine the number of Process Chain executions in your system, run the following SQL statement in your BW database:

```
select count(distinct LOG_ID) PERFMAXROWS from RSPCPROCESSLOG
```

Specifying the process duration threshold

About the perfFilter threshold – Processes with a run-time (in seconds) lower than this value will not be returned. A value of zero for this value will cause all the Process executions to be returned.

A higher value will cause less runs to be returned and improve the user-experience.

We recommend that you tune this value to find a balance between a smooth user-experience and a rich set of data. The following SQL statement is an example of how to determine the value in a way that will cause 20,000 rows to be returned.

Oracle:

```
select min(DUR) as PERFILTERVALUE from (select max(ENDTIMESTAMP) –
min(STARTTIMESTAMPT) DUR, row_number() over(order by max(ENDTIMESTAMP) –
min(STARTTIMESTAMPT) – min(STARTTIMESTAMP) desc) mk from RSPCPROCESSLOG group by
LOG_ID) where mk <=20000;
```

MS-SQL:

```
select min(DUR) as PERFILTERVALUE from (select top 20000 max(ENDTIMESTAMP) –
min(STARTTIMESTAMP) DUR from [schema]. RSPCPROCESSLOG group by LOG_ID order by 1
desc)
```

How to modify the performance thresholds

To modify the above performance thresholds, do the following:

1. Open the file:
   <i3_root>\products\sap\focalpoint\bi\resources\SAPAdapterConfig.xml.
2. Modify the respective settings.
3. Save the file.

Configuring Precise for SAP

This section includes the following topics:

- Filtering users
- Enabling the filtering of a specific user
- Disabling the filtering of a specific user
- Importing SAP user information from ASCII files

Filtering users

In certain cases, it is required to exclude specific SAP user actions from being displayed in the Precise for SAP user interface.

For example, you may want to exclude the SAP user used by Precise for SAP and/or Interpoint for SAP to monitor and collect information from the SAP system.

Enabling the filtering of a specific user

To enable filtering a specific user:

1. Open a command prompt on the Precise for SAP FocalPoint server and change directory to the following folder:
   <i3_root>\products\sap\bin\
2. Run the following command:
   I4SAPExcludeEnable.bat <USERNAME>
   where <USERNAME> has to be replaced with the SAP user you want to filter out. The user name must be in UPPER CASE.
3. Restart the Precise for SAP FocalPoint.

Disabling the filtering of a specific user

To disable filtering a specific user:

1. Open a command prompt on the Precise for SAP FocalPoint server and change directory to the following folder:
   <i3_root>\products\sap\bin\
2. Run the following command:
   I4SAPExcludeEnable.bat (without any parameters).
3. Verify that this folder:
   <i3_root>\products\sap\focalpoint\dataretriever\resource
   contains the same files as:
   <i3_root>\products\sap\focalpoint\dataretriever\resource.all
4. Delete the folders:
   <i3_root>\products\sap\focalpoint\dataretriever\resource.all
   and:
   <i3_root>\products\sap\focalpoint\dataretriever\resource.noi3
5. Restart the Precise for SAP FocalPoint.

   > (i) The filter will affect all monitored instances and only 1 user can be filtered out. Therefore, if the filter option is used it is recommended to configure the same monitoring user for all SAP systems.

Importing SAP user information from ASCII files

Precise for SAP automatically imports user information directly from SAP user account tables. User information is imported into Precise for SAP during the installation and is updated each night, automatically adding new users and modifying existing user information.

If your system maintains user lists in locations other than SAP user tables, you can still use them to import information into Precise for SAP. Precise for SAP can import user definitions from ASCII text files, enabling user information to be extracted from practically any source.

Once you have performed a manual import, you should not turn back on the automatic import feature. If you turn back on the automatic import feature, any users that were manually imported and do not appear in the SAP user account tables will be deleted from the PMDB.

> ⓘ   It is recommended that you keep the ASCII text file in another location. When you want to import new users, add them to the ASCII text file and again import the entire list. The assignment of user areas to locales is not deleted during a manual import of the ASCII text file.

Importing from ASCII files

By default, Precise for SAP is set to automatically import data from SAP tables. To manually import user information from ASCII text files, you need to do the following:

- Turn off Automatic Import and do not turn it back on (for more information, see To turn off the automatic import feature in Precise for SAP.)
- Create an ASCII file that contains a list of users.

The ASCII import file must be a comma separated value (CSV) file. This format stores each record as a new line and separates each field with a comma. If the field value contains a comma, enclose the field within double quotes (" "). If you enclose the field within double quotes and the field value also contains a double quote, replace the double quote with two double quotes. For example, a hypothetical field called Corporate "Sales" Force, would be written as Corporate ""Sales"" force.

The following is an example of the contents that can appear in a user text file:

"Production","PRD","000","CHRIS","Ohio - Bldg1","Safety Engineering" Production,PRD,000,PAT,Ohio - Bldg2,"Corporate ""Sales"" Force"

- Convert the ASCII file to internal binary file format.

You must convert the created ASCII file to a Precise internal binary file format using a utility called:

pssp_convert_userfile.exe.

Because other components are dependent on it, you must place this module in the following directory:

   <i3_root>\products\sap\bin.

Otherwise, an error message appears, indicating that required DLL files are missing. Place the binary file in the appropriate Precise for SAP directory in preparation for loading.

To turn off the automatic import feature in Precise for SAP

- Add <user-list>0</user-list> to the following file:

   <i3_root>\products\sap\cfg\IS1\agents-config.xml

   and before </workload-settings>.

To create an ASCII file that contains a list of users

- Create an ASCII file that creates the following fields:

> ⓘ All fields are mandatory and must be separated by a comma without blanks. Each record needs to be created on a separate line.

Field name	Description
system name	The System Name is a free text description of the SAP system. "Production" is an example of a System Name. Use the same System Name used in the Precise for SAP installation program. You must use the same exact name and case, as defined for this system in Precise for SAP.
SID	Indicates the SAP 3 alpha-numeric characters SID (System ID) for this user's SAP system; for example, "PRD"
client	Indicates the SAP client for this user; for example, "000" Define users for each client you want Precise for SAP to monitor.
userid	Indicates the SAP user ID for this user. The user ID must be in upper-case.
user area	Indicates the user area where this user is located. For example, "Ohio - Bldg1" User areas are mapped to individual locales with the Precise for SAP Locale Setting.
organization	Indicates the business organization for this user ID; for example, "sales"

To convert the ASCII file

- Run the following command:

      PSSP_CONVERT_USERFILE ascii-file sid [org-field] [user-area-field]

The `sid` parameter indicates the SAP SID for which users are imported; for example "PRD". All lines in the file should contain a matching SID column.

The `[org-field]` and `[user-area-field]` represent a number that indicates the field used for the values in the file, as follows:

1. Department
2. Building
3. Location
4. Name1
5. Name2
6. Name3
7. Name4
8. Region
9. Cost center
10. Account
11. Class

The default value for `[org-field]` is 1 (Department). The default value for `[user-area-field]` is 2 (Building).

This command creates a new binary file in the same directory as the ASCII file. The name of the file will be `00000000.000.<sid>usrlist`. This file is located on the server where the SAP AppTier Collector agents are installed. It is recommended that you view the file when the conversion process has completed to verify that it contains all the users included in the original file.

To import the binary file

1. Verify that you have turned off the Automatic User feature. If this feature is turned on, it can cause a conflict between a user list that is generated automatically and the user list that you created.
2. To incorporate the binary file in PMDB, copy it to the following directory: `<i3_root>\products\sap\rfg`

Once you have copied the file to this folder, it will automatically be loaded into the PMDB and then deleted.

To turn on the automatic import feature

NOTE   If you turn back on the automatic import feature, any users that were manually imported and do not appear in the SAP user account tables will be deleted from the PMDB.

- Delete the `<user-list>0</user-list>` element from the file:

    <i3_root>\products\sap\cfg\IS1\agents-config.xml

Configuring Precise for Microsoft .NET

This section includes the following topics:

- About configuring Precise for Microsoft .NET
- About the instrumentation file
- About the ActivityCollector.xml file
- Defining the DLLs to be monitored by using the Detection agent
- Invoking the Instrumentation Driver utility

About configuring Precise for Microsoft .NET

When we discuss configuring Precise for Microsoft .NET, we talk about two sections:

- Dynamic instrumentation
- Tracking instrumentation activity

The Microsoft .NET AppTier Collectors agents use the dynamic instrumentation of your Microsoft .NET-based applications to track software activities that are based on Microsoft .NET. Therefore, the instrumentation configuration determines the tracking quality, the amount of overhead involved, and whether the data collected really suits your specific needs.

About the dynamic configuration files

The rules for dynamic instrumentation configuration are defined in the following files:

- **instrumentation.xml**. This file contains explicit instrumentation rules for the tracked instances
- **default-instr-config.xml**. This file contains the default instrumentation rules for the tracked instances.

  > ⓘ Do not modify this file without consulting Precise Technical Support.

- **ActivityCollector.xml**. This file contains the configuration of the Microsoft .NET AppTier Collector agent, which determines other tracking rules, including implicit instrumentation rules.

Limitations of tracking instrumentation activity

Tracking instrumentation activity is subject to the following limitations:

- Only the Microsoft .NET versions 3.0, 3.5, 4.0, 4.5, 4.62, and 4.7 Frameworks are supported.
- The time resolution (granularity) of the tracked activity is approximately 16 milliseconds.
- Certain system DLLs that are part of the Microsoft .NET infrastructure cannot be instrumented. It is strongly recommended not to add an additional system DLL to the DLL list without first contacting Precise Technical Support.

  > ⓘ All system DLLs that are discussed, are safe for instrumentation.

- Generated Microsoft .NET wrappers for COM+ components that are produced by Microsoft .NET Framework SDK utilities (such as TLBImp) cannot be tracked.
- "Pre-Jitted" assemblies cannot be instrumented because they were compiled beforehand. For example, the System.Windows.Forms.dll is pre-jitted in the Microsoft .NET framework.

About the instrumentation file

The Instrumentation.xml file is located in the `<i3_root>\products\dotnet\config` directory. It contains the Instance-specific section.

Following is an example of the file structure:

```
<?xml version="1.0" encoding="utf-8"?>
<instrumentation-config>
<instances>
   <instance name="Instance_A">
      Instance A instrumentation rules
   </instance>
   <instance name="Instance_B">
      Instance B instrumentation rules
   </instance>
</instances>
</instrumentation-config>
```

The file consists of the following subsections:

- **DLL list**. A list of DLLs to be instrumented and specific configuration rules for each DLL. If the underlying process of the instance should be instrumented, it must be mentioned explicitly.
- **Common instrumentation rules ("instrument")** for the whole DLL list.
- **Common exclude rules ("ignore")**. Following is an example of the file structure:

```
<dlls>
   <dll name="dll_A_name" />
   <dll name="dll_B_name" />
   <dll name="dll_C_name" >
      <instrument>
         Specific instrumentation rules for DLL C.
      </instrument>
      <ignore>
         Specific ignore rules for DLL C.
      </ignore>
   </dll>
   <dll name="dll_D_name" >
      <instrument>
         Specific instrumentation rules for DLL D.
      </instrument>
      <ignore>
         Specific ignore rules for DLL D.
      </ignore>
   </dll>
</dlls>
<instrument>
      Common instrumentation rules for all the DLLs in this context (A, B, C, D).
</instrument>
<ignore>
      Common ignore rules for all the DLLs in this context (A, B, C, D).
</ignore>
```

Example of the instrumentation.xml file

In the following example, all DLLs with the prefix "Pet" are instrumented and tracked, regardless of their path. Likewise, for the classes with the prefix "Pet", all methods are instrumented and tracked.

```xml
<?xml version="1.0" encoding="utf-8"?>
<instrumentation-config>
    <instances>
        <instance name="AspNetIIS5">
            <dlls>
                <dll name="Pet*" />
            </dlls>
            <instrument>
                <classes>
                    <class>
                        <name>Pet*</name>
                        <called-method>
                            <methods>
                                <method>
                                    <name>*</name>
                                </method>
                            </methods>
                        </called-method>
                    </class>
                </classes>
            </instrument>
        </instance>
    </instances>
</instrumentation-config>
```

About the instrumentation.xml file tags

The following table describes the tags in the Instrumentation.xml file.

Table 13-1 Instrumentation file tags

Tag name	Description	Comment
instrumentation-config	The top level XML tag.	
instances	The list of instances to be instrumented.	
instance name (attribute)	The logical name of the instance.	
instance tracker (attribute)	The tracker DLL's reference to use.	Optional tag.
dlls	The list of DLLs to be instrumented when loaded by this instance.	Without a list of DLLs, all loaded DLLs should be instrumented.

Tag name	Description	Comment
`dll name` (attribute)	The name of the DLL.	Pattern match (for example `Company.GUI.*`) If the main module of the process must also be instrumented, the process name should be added explicitly to this DLL list.
`dll path` (attribute)	The path of the DLL.	Pattern match. Optional tag. Without this tag, all DLLs with the indicated name are instrumented.
`dll version` (attribute)	The version of the DLL.	Pattern match. Optional tag. Without this tag, all DLLs with the indicated name are instrumented.
instrument	The include list for instrumentation.	
interfaces	List of interfaces to be instrumented.	All classes that implement these interfaces must be instrumented.
interface	A specific interface (item in the interfaces list).	
classes	The list of classes to be instrumented. Also classes that inherit from these classes must be instrumented.	The same syntax as the interfaces section.
class	A specific class (item in the classes list).	
`name` (in interface/class section)	The specific name of the interface/class.	Pattern match. (for example, using * as the name indicates all names in this context)
caller-method	The list of methods to be instrumented on the callee side.	
methods	The beginning of the list.	
method	The description of a specific method (within the classes/interfaces context).	
type	The type of method invocation.	Optional tag (the default is Custom). Specified in the `default-inst-config.xml` file.

Tag name	Description	Comment
name (of method)	The specific name of the method.	Pattern match (for example, using * as the method name indicates all methods in this context).
params	The list of parameter types.	
param	A specific parameter type.	
all-calls-to-method	The list of methods to be instrumented on the caller side.	The same syntax as the caller-method section.
ignore	The exclude list for instrumentation.	The syntax of this section and its subcomponents is identical to the instrument section.

About instrumenting DLLs from the GAC

The Global Assembly Cache (GAC) is a logical directory that stores all assemblies that can be shared among applications. An assembly is placed in the GAC at deployment time, using either an installer that knows about the assembly cache or the Global Assembly Cache Utility (gacutil.exe), found in the MS .NET Framework SDK.

To instrument a DLL from the GAC, do not specify the path attribute in the DLL list. Instead, use only the DLL name. To instrument a DLL with a specific version, use the version attribute.

About instrumenting DLLs of an ASP.NET application

The ASP.NET DLLs are typically loaded from a special temporary directory, such as:

```
C:\WINNT\Microsoft.NET\Framework\v1.1.4322\Temporary ASP.NET
Files\mspetshop\c73f3fa0\1ae4c2f3\assembly\dl2\bee75c45\aa2e9da0_9807c501)
```

As in the case of the GAC, to instrument such DLLS, do not specify the ASP.NET DLL path and only use the DLL name.

About instrumenting standalone applications (Console, WinForm, Windows services)

A standalone application that starts with a main method usually does not finish before the application terminates. This main method typically invokes other methods that actually perform most of the activities in the application. These other methods are started and stopped quite frequently, but their top-level caller (the main method) remains running.

The Microsoft .NET AppTier Collector agent is designed to report only when a complete sequence of method invocations has finished. To deal with this kind of scenario, the ignore tag is used in the default instrumentation file to exclude the instrumentation and the tracking of the main method.

About callee vs. caller-side instrumentation (all-calls-to-method)

Callee-side instrumentation means instrumenting the methods (their prologues and epilogues) to track the method invocations. Caller-side instrumentation means instrumenting the calls to the methods to track the called method invocations. These techniques are two different ways to gather the same data.

Precise for Microsoft .NET generally uses the callee-side instrumentation technique. If callee-side instrumentation is not possible, caller-side instrumentation can be applied instead. To do so, replace the default called-methods tag with the all-calls-to-method tag. This alternative is best used when the tracked method is part of the Microsoft .NET framework system DLLs that cannot be instrumented directly, such as the System.dll.

About tracking service requests (URLs)

Service Requests are tracked by default. The ability to track Service Requests (URLs) relies on the following blocks, which are part of the default instrumentation file:

```
<dll name="System.Web.dll" version="1.*">
    <instrument>
        <classes>
            <class>
                <name>System.Web.HttpRuntime</name>
                <called-method>
                    <methods>
                        <method type="HTTP">
                            <name>ProcessRequest</name>
                        </method>
                    </methods>
                </called-method>
            </class>
        </classes>
    </instrument>
</dll>
<dll name="System.Web.dll" version="2.*">
    <instrument>
        <classes>
            <class>
                <name>System.Web.HttpRuntime</name>
                <called-method>
                    <methods>
                        <method type="HTTP">
                            <name>ProcessRequestNoDemand</name>
                        </method>
                    </methods>
                </called-method>
            </class>
        </classes>
    </instrument>
</dll>
```

The instrumentation configuration partially filters the tracking of Service Requests: Service Requests are only tracked if the respective Web application's DLLs are listed in the instrumentation file.

About tracking SQL statements

SQL statements are tracked by default. The ability to track SQL statements relies on the following blocks, which are part of the default instrumentation file:

```
<dlls>
<dll name="System.Data.Oracleclient.dll">
    <instrument>
      <classes>
        <class>
<name>System.Data.OracleClient.OracleCommand</name>
          <called-method>
            <methods>
              <method type="ADO">
<name>ExecuteOracleNonQuery</name>
              </method>
              <method type="ADO">
                <name>ExecuteOracleScalar</name>
              </method>
              <method type="ADO">
                <name>ExecuteReader</name>
              </method>
              <method type="ADO">
                <name>ExecuteScalar</name>
              </method>
              <method type="ADO">
                <name>ExecuteNonQuery</name>
              </method>
              <method type="ADO">
                <name>Prepare</name>
              </method>
            </methods>
          </called-method>
        </class>
      </classes>
    </instrument>
</dll>
<dll name="System.Data.dll">
    <instrument>
      <classes>
        <class>
          <name>System.Data.SqlClient.SqlCommand</name>
          <called-method>
            <methods>
              <method type="ADO">
                <name>ExecuteReader</name>
              </method>
              <method type="ADO">
                <name>ExecuteXmlReader</name>
              </method>
              <method type="ADO">
                <name>InternalExecuteNonQuery</name>
              </method>
            </methods>
```

```
          </called-method>
        </class>
        <class>
          <name>System.Data.Common.DataAdapter</name>
        <called-method>
            <methods>
              <method type="ADO">
                <name>Fill</name>
              </method>
              <method>
                <name>Update</name>
              </method>
            </methods>
          </called-method>
        </class>
      </classes>
    </instrument>
</dll>
<dll name="Oracle.DataAccess.dll">
    <instrument>
      <classes>
        <class>
          <name>Oracle.DataAccess.Client.OracleCommand</name>
          <called-method>
            <methods>
              <method type="ADO">
                <name>ExecuteReader</name>
              </method>
              <method type="ADO">
                <name>ExecuteScalar</name>
              </method>
              <method type="ADO">
                <name>ExecuteNonQuery</name>
              </method>
              <method type="ADO">
                <name>ExecuteXmlReader</name>
              </method>
            </methods>
          </called-method>
        </class>
      </classes>
    </instrument>
</dll>
</dlls>
<!-- Instrument all the methods of classes which implement the following interfaces. -->
<instrument>
    <interfaces>
    <!-- Uncomment the following section to measure ADO.NET Open Connection response times -->
      <!-- <interface>
        <name>System.Data.IDbConnection</name>
        <called-method>
          <methods>
            <method type="ADO">
```

```
                    <name>Open</name>
                </method>
            </methods>
        </called-method>
    </interface> -->
    <interface>
        <name>System.Data.IDbCommand</name>
        <called-method>
            <methods>
                <method type="ADO">
                    <name>ExecuteReader</name>
                </method>
                <method type="ADO">
                    <name>ExecuteScalar</name>
                </method>
                <method type="ADO">
                    <name>ExecuteNonQuery</name>
                </method>
                <method type="ADO">
                    <name>Prepare</name>
                </method>
            </methods>
        </called-method>
    </interface>
  </interfaces>
</instrument>
```

These blocks define the DLLs and methods that are used for SQL statement extractions. This configuration supports the following ADO.NET providers:

- Microsoft provider for SQL-Server connectivity
- Microsoft provider for Oracle connectivity
- Microsoft ODBC connectivity
- Microsoft OLEDB connectivity.
- Oracle provider for Oracle connectivity

All SQL statements the above ADO.NET providers and methods use are tracked, even if the methods that call these ADO.NET methods are not explicitly selected for tracking in the Instrumentation.xml file.

The default instrumentation file for Precise for Microsoft .NET does already contain these blocks. If your application uses other database providers, consult Precise Technical Support.

About tracking COM+ (Enterprise services)

COM+ components can be developed and used in the following way:

- The COM+ library components are executed and used as a library (DLL) in the context of an executable. The calls to the COM+ component methods look like other intra-process method calls.
- The COM+ server components are hosted in an external COM+ container. A call from the COM+ component client (process) to the hosted COM+ component (external server process) is an inter-process call. In other words, activity exists in two different processes: the client-side process (caller-side) and the server-side process (callee-side).

To have the Microsoft .NET infrastructure reuse COM+ technology, perform the following tasks:

1. Develop Microsoft .NET-based DLLs (Enterprise Services) on top of the COM+ engine.

2. Reuse COM+ components with Microsoft .NET interoperability mechanisms (this case equals other interoperability cases and are not discussed here).

Calls to COM+ library components that were developed in Microsoft .NET equal other Microsoft .NET DLLs (for instrumentation purposes). To track the calls to the methods of these components, you only need to add the appropriate DLLs to the <dlls> list of the instance and specify the classes and methods to be tracked in the <instrumentation> block.

To track COM+ server components that are hosted in the COM+ container, perform the following tasks:

1. To track server-side activity (the activation of the components' methods), add a new instance to the server for the COM+ container using Precise Agent Installer. The underlying process name of the COM+ host is dllhost.exe. In Precise Agent Installer, use this process name, but do not specify the path.
   The <dlls> list of this new instance must include all DLLs of the Microsoft .NET-based COM+ server components to be tracked. The <instrument> block must specify the classes and methods that need to be instrumented.
2. To track client-side activity (the calls to the COM+ components), add the DLLs of the appropriate COM+ component to the client instance (with the appropriate <instrument> block). In addition, insert the following section under the instrumentation section for the COM+ client instance:

```
<dll name="System.EnterpriseServices.dll">
  <instrument>
    <classes>
      <class>
        <name>System.EnterpriseServices.ServicedComponentProxyAttribute</name>
          <called-method>
            <methods>
              <method>
                <name>CreateInstance</name>
              </method>
            </methods>
          </called-method>
      </class>
      <class>
        <name>System.EnterpriseServices. RemoteServicedComponentProxy</name>
          <called-method>
            <methods>
              <method>
                <name>Invoke</name>
              </method>
            </methods>
          </called-method>
      </class>
    </classes>
  </instrument>
</dll>
```

About tracking Web services (calls)

The callee-side Web service infrastructure (Web service server-side) is implemented in ASP.NET. In other words, tracking Web services in the called-side is covered by tracking Service Requests. See About the instrumentation file.

The Microsoft .NET framework contains comprehensive support for Web service clients (caller-side). For example, Visual Studio.NET generates a proxy class code that wraps the calls to the underlying Web service (the generated class extends an appropriate base class, such as System.Web.Services.Protocols.SoapHttpClientProtocol).

Web services are tracked by default. The ability to track Web service calls for all instances relies on the following blocks, which are part of the default instrumentation file:

```
<dll name="System.Web.Services.dll" >
  <instrument>
    <classes>
      <class>
        <name>System.Web.Services.Protocols. SoapHttpClientProtocol</name>
        <called-method>
        <methods>
          <method type="WS">
            <name>Invoke</name>
          </method>
          <method type="WS">
            <name>BeginInvoke</name>
          </method>
          <method type="WS">
            <name>EndInvoke</name>
          </method>
        </methods>
        </called-method>
      </class>
      <class>
        <name>System.Web.Services.Protocols. HttpSimpleClientProtocol</name>
        <called-method>
        <methods>
          <method type="WS">
            <name>Invoke</name>
          </method>
          <method type="WS">
            <name>BeginInvoke</name>
          </method>
          <method type="WS">
            <name>EndInvoke</name>
          </method>
        </methods>
        </called-method>
      </class>
    </classes>
  </instrument>
</dll>
```

About tracking the message queue API

To have Precise for Microsoft .NET track the usage of the Microsoft Message Queue API, you need to instrument the System.Messaging.dll file by manually adding the following blocks to the instrumentation block of the specific instance in the Instrumentation.xml file. Usually, it is sufficient to only instrument and track the methods Send and Receive to ensure that all appropriate Microsoft Message Queue call are tracked.

> ⓘ The default instrumentation file for Precise for Microsoft .NET does not contain these blocks. To track Microsoft Message Queue calls, you must add these blocks manually.

To instrument Message Queue calls, insert the following section under instrumentation rules for the monitored instance in the instrumentation file:

```
<dll name="System.Messaging.dll">
    <instrument>
        <classes>
            <class>
                <name>System.Messaging.MessageQueue</name>
                <called-method>
                    <methods>
                        <method>
                            <name>Send</name>
                        </method>
                        <method>
                            <name>Receive</name>
                        </method>
                        <method>
                            <name>BeginReceive</name>
                        </method>
                        <method>
                            <name>EndReceive</name>
                        </method>
                    </methods>
                </called-method>
            </class>
        </classes>
    </instrument>
</dll>
```

About the ActivityCollector.xml file

The ActivityCollector.xml file is the main configuration file of the Microsoft .NET AppTier Collector agent that gathers activity information. It is composed of the following logical sections:

- **Aggregator settings**. Contains the settings that are grouped under the aggregator tag.
- **Tracker settings**. Contains the settings that are grouped under the tracker tag.

The Aggregator settings sections are read when the Collector agent is loaded. Therefore, if you modify these sections, you must restart the Collector agent through Precise Agent Installer or by restarting the Precise for Microsoft .NET Collector service using the Services window.

The Tracker settings are read when the tracked Microsoft .NET instance (that is, the underlying Microsoft .NET-based process) is loaded. Therefore, if this section is modified, you must restart the Microsoft .NET instance (the underlying process). For the ASP.NET instance, you should run the IISRESET command.

The ActivityCollector.xml file is located in the `<i3_root>\products\dotnet\config` directory. Its settings affect all instances on the monitored server.

If you need to define instance-specific settings, for example for the Tracker threshold, place a copy of this file in the `<i3_root>\products\dotnet\config\instance name`
    directory and modify the settings as required.

The Aggregator and Tracker settings in this file then overrides the settings that are specified in the file that is located in the parent directory.

Example of the ActivityCollector.xml file

The following is an example for the ActivityCollector.xml file:

```
<?xml version="1.0" encoding="UTF-8"?>
<activity-collector-config>
    <!-- psdn_dncol_act.exe configuration, configurable per server(default) & per instance -->
    <aggregator>
        <topnsql>5</topnsql><!-- Top N SQL statement to monitor -->
        <sla>5000</sla><!-- Red SLA breach value for top http invocations in msec. -->
        <nearsla>1000</nearsla><!-- Yellow SLA breach value for top http invocations in msec. -->
        <insane-rt>300</insane-rt><!-- Long running thread timeout in seconds -->
        <!-- The following two items are related to the Insight Smartlink feature and should not be altered by the user manually -->
        <bit-vector>false</bit-vector>
        <last-level-bit-vector>false</last-level-bit-vector>
    </aggregator>
    <!-- Tracker.dll settings, configurable per server (default) & per instance -->
    <tracker>
        <threshold>50</threshold><!-- Threshold in msec for filtering out events before forwarding it to the collector. -->
    </tracker>
</activity-collector-config>
```

About the ActivityCollector.xml file tags

The following table describes the important tags in the ActivityCollector.xml file. You may modify these tags only. It is not recommended to modify any other tags.

Table 13-2 Collector configuration file tags

Tag name	Description
activity-collector-config	The top-level XML tag.
Aggregator	The top-level definitions for the Collector agent's aggregator.
Topnsql	The top number of SQL statements to monitor.
Sla	The SLA (red) value (in milliseconds) for ASP.NET instance URLs.
nearsla	The Near SLA (yellow) value (in milliseconds) for ASP.NET instance URLs.
insane-rt	The timeout value for long running threads/URLs. A method or URL that is longer than this threshold is not collected.
tracker	Specific definitions for the tracker.
threshold	The threshold (in milliseconds) for filtering events before they are forwarded to the Collector agent.

Defining the DLLs to be monitored by using the Detection agent

A Microsoft .NET instance consists of the DLLs that make up your Microsoft .NET application. For Precise for Microsoft .NET to monitor a Microsoft .NET instance, you must first define the DLLs that you want to monitor.

You can use the Detection agent to detect all relevant DLLs that are currently running in the background. Using the Detection agent involves manually modifying the instrumentation configuration file.

To detect the DLLs currently running by using the Detection agent

1. Verify that your Microsoft .NET application is running.
2. Open the command prompt window.
3. Change directory to the `<i3_root>` directory.
4. Run the following command:
   - Microsoft .NET Framework version 1.1.
     `products\dotnet\install\psdn_detect_dlls.exe /im` *image name* `/f` *filter file name*

   - Microsoft .NET Framework version 2.0 and 3.0.
     `products\dotnet\install\FW2.0\psdn_detect_dlls.exe /im` *image name* `/f` *filter file name*

   where the parameter values should be set as follows:
   - **image name**

        If this is an ASP.NET instance , the name of this executable depends on the Internet Information Server (IIS) type.
        - IIS 6: `w3wp.exe`
        - IIS 7:

        If this is a regular .NET instance , the image name is the name of the .NET executable (without the path).
   - **filter file name**

        The path and name of the file that lists the modules to be filtered out, as follows:
        `products\dotnet\install\dlls_filter.xml`
   As alternative to this step, you can also use the Process Explorer. See http://technet.microsoft.com/en-us/sysinternals/bb896653.aspx.
5. Save the XML output of the command to a temporary file.

To define the DLLs in the instrumentation file

1. Open the following file in an editor:

     `<i3_root>\products\dotnet\config\instrumentation.xml`
2. Within the XML output that you saved in step 5 above, locate the <module name> tag for a DLL file you want to instrument and copy the file name.
   For example, if you want to instrument the file petshop.web.dll, copy its name from the line `<module name="petshop.web.dll">`.
3. In the instrumentation file, paste the name of the DLL file into the entry for the instance that you want to monitor.
   For example:
   <instances>
     <instance name="AspNetIIS6" >
       <dlls>
       <dll name="petshop.web.dll"/>
       ...
       </dlls>

```
        </instance>
      </instances>
```
4. Repeat step 2 and step 3 for each DLL file that you want to instrument.
5. Restart your Microsoft .NET application for the changes to take effect.

Invoking the Instrumentation Driver utility

The instrumentation process is the execution of a Collector agent, which reads the code (the DLLs) of your application and stores an instrumented version in the `<i3_root>\products\dotnet\cache\instr` cache directory:

The contents of the cache can then be used for future invocations of your Microsoft .NET application, avoiding the time to re-instrument. The instrumentation process can last anywhere between a few minutes and an hour, depending on the size of the DLLs.

The instrumentation process consumes memory. The required memory for instrumenting a DLL is approximately 20 times the size of that DLL file. As a result, the startup of your Microsoft .NET application can be considerably slow.

The instrumentation is triggered when the Microsoft .NET application starts and one of the following criteria is applicable:.

- The .NET instance is installed for the first time.
- A change is made to the instrumentation.xml file (the instrumentation configuration file).
- A new code version is deployed on the monitored server.

To minimize the time it takes the Microsoft .NET application to start up, you can invoke a Precise for Microsoft .NET utility called Instrumentation Driver. The Instrumentation Driver populates the cache before it restarts the Microsoft .NET application.

> ⓘ   If the instance uses DLLs from several directories or several single DLLs, invoke the Instrumentation Driver utility several times using the appropriate directory or DLL names.

It is recommended to stop your Microsoft .NET application before you invoke the Instrumentation Driver utility. If you invoke the utility while the application is running, the instrumentation results are placed in the cache and used only during the next startup of the Microsoft .NET application.

If your application changes, for example because a new version is deployed, or if you modify the instrumentation.xml file, you can activate the Instrumentation Driver utility to recalculate the cache contents.

> ⓘ   The Instrumentation Driver utility may not calculate the cache contents for all DLLs that the application uses. These DLLs are only instrumented during the next startup of the Microsoft .NET application.

To invoke the Instrumentation Driver utility

1. After the Microsoft .NET instance is installed, configure the DLLs, classes, and methods to be instrumented in the instrumentation.xml file.
2. Open a command prompt window and run the following command:
   cd `<i3_root>`
   products\dotnet\bin\psdn_instr_validate.exe [-k *instance-name*]
   -f *directory-name* | -dll *name* [-gac]
   where:
   instance-name is the Microsoft .NET instance name, such as AspNetIIS5. If this parameter is omitted, the Instrumentation Driver works for all the Microsoft .NET instances that are installed on the monitored server.
   directory-name is the name of the DLL directory of the Microsoft .NET instance to be instrumented. This is the location where you deployed your Microsoft .NET code. For example, a DLL for an ASP.NET application

may be stored in:

```
C:\Inetpub\wwwroot\MyWebApp\bin
```

name is the name of a single DLL (without the path). If this DLL is located in the GAC, add `-gac` when running the command.

Configuring Precise for Tuxedo

This section includes the following topics:

- The challenge
- Solution overview
- Working with Precise for Tuxedo

The challenge

Providing a solution that enables you to see end-to-end information involving Web, multiple J2EEs, Tuxedo, and Oracle.

Solution overview

SmartLink correlates end-to-end activities and provides information on your environment's performance. This information includes total response time broken down into AppTiers, and a complete cross-AppTier invocation tree for each transaction. This enables pinpointing the problematic activity in a specific AppTier that is causing poor performance in the transaction, and drilling down in context of that activity to the relevant Precise product, for further analysis and resolution.

The solution has been specifically designed to:

- Provide end-to-end correlation visibility for transactions whose flow includes Web, J2EE, Tuxedo, and Oracle.
- Recognize Tuxedo calls from J2EE in two protocols (WTC-CORBA and WTC-JATMI).
- Support both cross-JVM correlations and the J2EE>Tuxedo>Oracle correlation. The supported AppTiers for the solution are: Web, J2EE, Tuxedo, and Oracle.

The SmartLink user interface has two tabs: Activity and Flow. The Activity tab is the default tab that displays the response time breakdown of transactions in an application. The Flow tab displays the complete invocation tree of a specific transaction. Additionally, SmartLink data is available in the Insight user interface when switching to **SmartLink is On** mode.

> ⓘ For additional information on SmartLink, its Flow and Activity tabs, and Insight with SmartLink ON, refer to the *Precise Insight User's Guide*, Chapter 7 – Running SmartLink for PeopleSoft environments.

Working with Precise for Tuxedo

This section includes the following topics:

- Usage scenario
- About the SmartLink user interface within Insight
- Drilling down to other Precise products from the SmartLink Flow tab

Usage scenario

To work with Precise for Tuxedo, perform the following steps:

1. On the SmartLink user interface: Analyze transaction breakdown across Web > J2EE1 > J2EE2 >…> J2EEn > Tuxedo.
2. On the Insight user interface with "SmartLink On": Analyze transaction breakdown across J2EEn > Tuxedo > Oracle (J2EEn = last J2EE in the call chain which invokes Tuxedo).

3. From the StartPoint screen, select **Environment 1** (Web and J2EE).
4. On the upper right-hand side of the screen, select **SmartLink**. This will open SmartLink in the Activity tab.
5. Click the Flow tab.
6. The table on the left side of the screen displays your transactions and the Tree view on the right side of the screen displays the breakdown of the transaction selected on the table.
7. On the Tree view, select the service you want to investigate.
   A fish eye view of the service is displayed providing you with additional information on the service.
8. To further analyze the service, click the **Analyze** link.

> ⓘ If you select a Tuxedo service and click Analyze, the Insight screen is displayed with the Tuxedo tab selected and the **SmartLink is On** mode enabled.

About the SmartLink user interface within Insight

When SmartLink is activated with "**SmartLink is On**," a chart showing how the time of each AppTier relates to the total time, is displayed.

The Main area displays a table showing the performance data divided by entities.

> ⓘ This Insight screen is suitable for investigating the transaction from the last J2EE tier, through the Tuxedo, to the database tier. The Cross-AppTiers and Web tabs are not relevant for this solution.

About the left pane

The left pane displays an Overview pie graph and an Over Time bar graph below it.

The Overview graph displays the response time breakdown to AppTiers net times for the transactions displayed in the table on the right pane.

The Over Time graph displays the response time summed over time (shown as a separate graph for each AppTier).

About the right pane

In the Insight user interface, the default view for each AppTier (tab) is the graphs view. If you select a specific entity in the Display drop down menu, then the entity data is displayed in a table view. The table view details information regarding the transactions associated with the Overview graph, displayed on the left pane.

If you perform a drill down operation on the right pane, all AppTiers in Insight are affected. For example, if you perform a drill down (in Insight) on a Tuxedo service in the Tuxedo tab, and then switch to the Oracle tab, Insight will display a subset of the information (resulting from the drill-down operation).

The drill down operation also affects the graphs on the left pane. For example, if you drill down a Tuxedo service on the right pane, the graphs on the left pane will display a subset of the information (resulting from the drill-down operation).

Drilling down to other Precise products from the SmartLink Flow tab

From the SmartLink Flow tab you can drill down directly to Precise for Web, Precise for J2EE, Precise for Oracle, and Insight (for Tuxedo nodes) in context of a specific element.

The context is set according to the selected tree node and selected row in the Instances table.

> ⓘ Precise for Oracle data is available if activated from the J2EE tier, not the Tuxedo tier.

Administrating Precise communication

This section includes the following topics:

- Changing a Precise Listener port
- About direct communication

Changing a Precise Listener port

You may need to change the port that a Precise Listener listens on after you complete the installation of Precise. For example when another application requires usage of this port, or when the firewall policy changes.

To change a Precise Listener port, use the CLI command: Listener Port Change command. For more details, refer to the *CLI User's Guide*.

About direct communication

The communication transmission between two different product FocalPoints or agents that are based on Java, such as the Precise for J2EE Collector agent, Alerts InformPoint, and Precise for Web agents is a direct communication and it relies on that the Precise default communication port is open between the servers.

Be aware of the following facts:

- Communication between two agents on the same server is direct by default.
- Communication between two agents on different servers is through the Precise Listener port.

Altering between direct and indirect communication

You can change the direct and indirect communication parameter in the Precise registry for sites for which direct communication cannot be enabled due to Firewall limitations. Changes in this direct and indirect communication parameter apply the new behavior to all FocalPoints and agents that are based on Java. You can also create an indication file on a specific server that runs FocalPoints or agents that are based on Java to change the behavior only for this server.

To change the direct and indirect communication for all java based agents

1. Stop the Java based agents on all the servers.
2. Log on to the server with the Precise FocalPoint and edit the Precise registry file:
   <i3_root>\products\i3fp\registry\products\infrastructure\communication\settings.xml
3. Change the direct-communication parameter from 'true' to 'false' as shown in the following excerpt:
   <direct-communication>false</direct-communication>
4. Restart the agents that you have stopped.

To change the direct and indirect communication for a specific server

1. Stop the Java based agents on the server.
2. Create an empty indication file in the following folder:
   <i3_root>\Infra\listener\etc\disable_direct_routing
3. Restart the agents that you have stopped.

Configuring Alerts settings

This section contains the following topics:

- About role management in Alerts
- About configuring Alerts general settings
- About configuring Alerts metric settings
- About editing metric properties
- About setting Alerts SNMP connectivity
- About Alerts MOM connectivity
- About creating customized metrics

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.

> (i) 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.

> (i) 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.

> (i) 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.

Dynamic parameter	Definition
$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.

Dynamic parameter	Definition
$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.	Get
		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.	

No.	Field	Description	Applies to
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

No.	Field	Description	Applies to
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
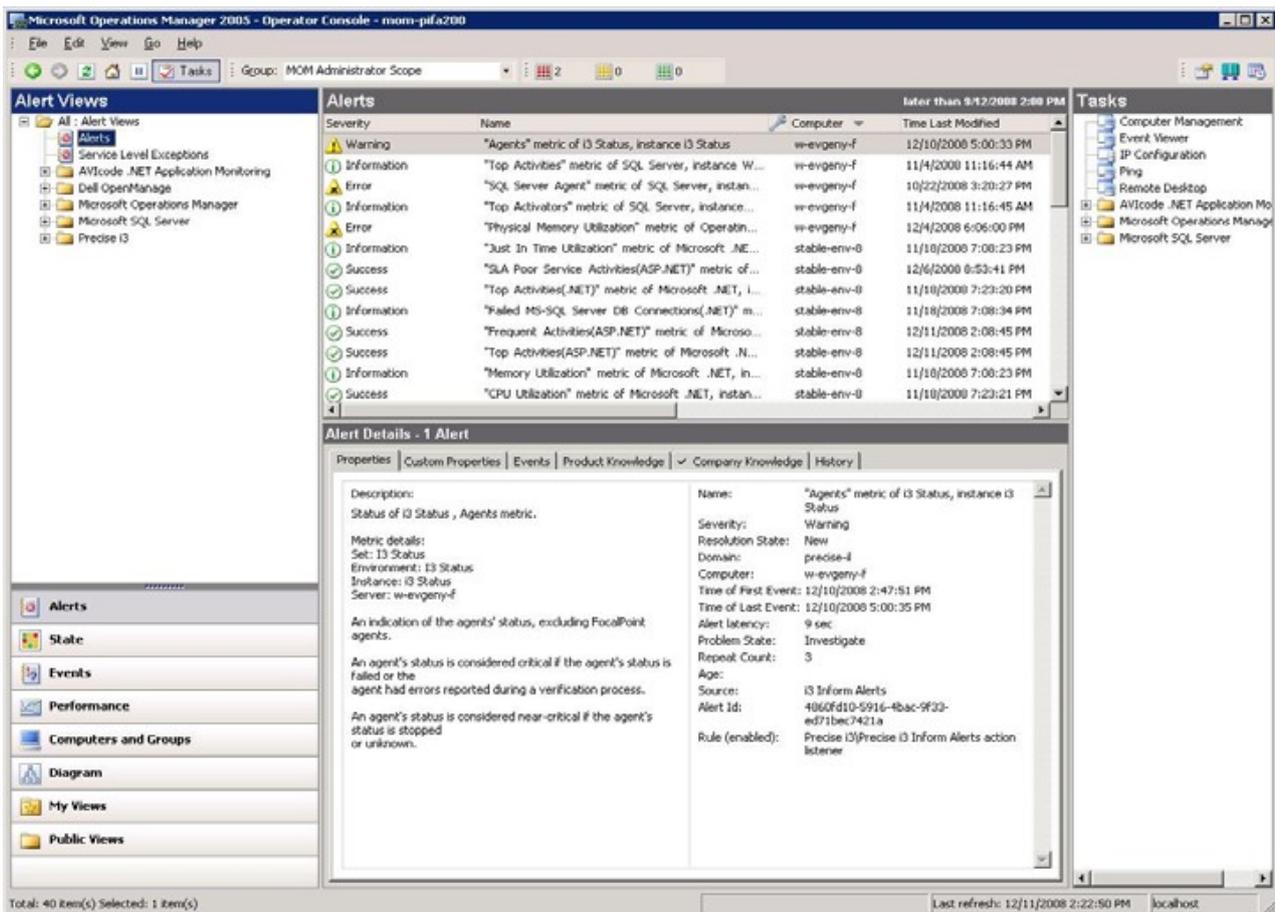
If you have successfully installed the Precise Management Pack (see *Precise Installation Guide*), you will see a Precise folder in the Alert Views tree. This folder displays on the Alerts View Results pane, all actions generated as MOM alerts.

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

Alerts alert	MOM severity
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:

Windows `.\products\pulse\userprograms\MYPROG.bat`.

UNIX or Linux `./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.txt then cat test.txt | wc -c else echo 0 fi	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.

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.

Dynamic parameter	Definition
@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.

Advanced settings

This section contains the following topics:

- Changing the location of installation packages
- About over-instrumentation protection

Changing the location of installation packages

The Precise FocalPoint agent stores installation packages files in the following folder:

`<i3_root>/products/i3fp/distribution_source`

The distribution source folder contains a great quantity of files, and consumes a lot of disk space. If you have difficulties in allocating disk space on the Precise FocalPoint file system, you can specify an alternative folder as the distribution source folder. This alternative folder can be on the same machine or on a network share.

To define an alternative distribution source folder

1. Stop the Precise FocalPoint.
2. Create a folder in another location and copy all the files from the folder path below, to the new folder.
   `<i3_root>/products/i3fp/distribution_source`
3. Edit the file path:
   `<i3_root>/products/i3fp/registry/products/infrastructure/setup/settings.xml`
4. Change:
   <distribution-source-location>products/i3fp/distribution_source</distribution-source-location> to the new location of the distribution source folder (the one you've copied the files to)

> ⓘ Use the forward slash (/) in this entry for UNIX and Windows systems.

If you have placed the folder on a network share, you should:

- Provide the Precise user with Read and Write permissions for this folder.
- On Windows, change the Precise FocalPoint service authentication from Local System account to an administrator that has permissions to this folder.

For more details about running Precise with a different user, see appendix D in the *Installation Guide*.

> ⓘ Do not share the same distribution_source folder amongst several Precise installations.

> ⓘ In case of a federation installation, only the main Precise FocalPoint holds the distribution source folder, and all operations mentioned in this section are relevant only to the main Precise FocalPoint server.

About over-instrumentation protection

> ⓘ Over-instrumentation protection should be implemented when in non-production mode only.

Over-instrumentation protection monitors the overhead that instrumentation introduces and turns off instrumentation when the contribution to the overhead from instrumentation exceeds the thresholds that were user-defined or set by adaptive instrumentation relative to the current overhead budget. Over-instrumentation protection estimates the average percentage that instrumentation contributes to the response time of a method or

method invocation. If that time exceeds the threshold, instrumentation on that method or method invocation is disabled and messages are written to the application system error stream and to the over-instrumentation protection log file.

Configuring over-instrumentation protection

Over-instrumentation protection is enabled by default because the OverInstrumentationProtection.xml file is included in the InstrumenterConfigList.xml file, which is located in the `<i3_root>/products/j2ee/config/ JVMID` directory.

Over-instrumentation protection uses the following tunable parameters that are contained in the system.properties file located in each JVM's configuration directory:

- **Sample Rate**. The sample rate specifies how frequently the overhead of instrumentation is assessed for a method or method invocation. Default: 1000 invocations.
- **Threshold**. The threshold specifies the maximum percentage of overhead that instrumentation may contribute to the average response time of a method or method invocation. Adaptive instrumentation sets this value. When adaptive instrumentation modifies the threshold, the previous value is commented in place and the new threshold is appended. Default: 50%.

> ⓘ This default does not mean that the application runs 50% slower, nor does it directly correlate with the actual performance of the application.

- **Minimum Invocation Count**. The minimum number of invocations that must have completed before the overhead imposed by instrumentation is assessed. The minimum invocation count is used to ensure that a sufficient sampling of callee-side or caller-side response time has been accumulated before engaging over-instrumentation protection. Default: 1000 invocations
- **Absolute Threshold**. The maximum time, expressed in microseconds, that logger-based instrumentation may take to execute at any given call site. The threshold applies individually to all call sites that have logger-based instrumentation injected and enabled. Adaptive instrumentation sets this value. When adaptive instrumentation modifies the threshold, the previous value is commented in place and the new threshold is appended. Default: 100 microseconds.
- **Absolute Minimum Invocation Count**. The minimum sample of response times that is required to assume a valid sample for the absolute threshold. Default: The current setting for Minimum Invocation Count.
- **Absolute Minimum Data Collection Count**. The minimum sample of logger-based instrumentation response times that are required to assume a valid sample for the absolute threshold. Default: The current setting for Minimum Data Collection Count.

To disable over-instrumentation protection

- Remove the reference to the OverInstrumentationProtection.xml file in the InstrumenterConfigList.xml file.

To change the parameters of over-instrumentation protection

- Edit or create a system.properties file in the JVM-specific configuration directory `<i3_root>/products/ j2ee/config/JVMID` and add the following lines:
indepth.j2ee.runtime.overInstrumentationProtection.sampleRate=1000
indepth.j2ee.runtime.overInstrumentationProtection.threshold=50
indepth.j2ee.runtime.overInstrumentationProtection.
minimumInvocationCount=1000
indepth.j2ee.runtime.overInstrumentationProtection.
minimumDataCollectionCount=1000
indepth.j2ee.runtime.overInstrumentationProtection.
absolute.threshold=<microseconds>
indepth.j2ee.runtime.overInstrumentationProtection.
absolute.minimumInvocationCount=<count>

indepth.j2ee.runtime.overInstrumentationProtection.
absolute.minimumDataCollectionCount=<count>
The system.properties file is checked periodically for changes, so these parameters can be changed without restarting the monitored JVM. Any property that is set above the adaptive instrumentation comment that states "#The next lines were added by adaptive instrumentation" is retained intact regardless of the parameter settings. This handling of allows user customization to be retained and not overwritten. Absolute parameters that are set below the comment are reset by adaptive instrumentation.

About identifying over-instrumentation protection activity

When over-instrumentation protection disables instrumentation, it issues the following log messages to the application system error stream and to the Over-instrumentation protection log files:

```
<i3_root>/logs/j2ee.JVMID.oip.log
```

```
<i3_root>/logs/j2ee.JVMID.oip.trc
```

When over-instrumentation protection disables instrumentation on a method, an error message containing text such as the following is created:

```
disabled_logging:
[classname, methodname, signature, invocation-count, response-time, ]
data-collection-count, data-collection-time, threshold]
   <OverInstrumentationProtectionHandler>
```

When over-instrumentation protection disables instrumentation on a method invocation, an error message containing text such as the following is created:

```
disabled_logging:
[classname, methodname, signature, classname-called,
methodname-called, signature-called, invocation-count,
response-time, data-collection-count, data-collection-time, threshold]
<OverInstrumentationProtectionHandler>
```

where

- **classname** is the qualified class name of the method or caller of a method (callee-side).
- **methodname** is the name of the method (callee-side).
- **signature** is the signature of the method (callee-side).
- **classname-called** is the qualified class name of the method (caller-side).
- **methodname-called** is the name of the method (caller-side).
- **signature-called** signature of the method (caller-side).
- **invocation-count** is the cumulative number of invocations.
- **response-time** is the cumulative response time of invocations.
- **data-collection-count** is the cumulative number of logger invocations.
- **data-collection-time** is the cumulative response time of logger invocations.
- **threshold** is the threshold.

Using custom J2EE instrumentation

To target your application with fine granularity, you can create custom instrumentation configurations. Custom instrumentation lets you manually instrument classes that are not included in the default Precise for J2EE monitored classes. It also lets you exclude classes that are monitored by default. To use custom instrumentation, you must edit the appropriate instrumentation configuration file and verify that the edited file is uncommented in the InstrumenterConfigList.xml file, the configuration file that Precise for J2EE uses to keep track of your instrumentation configurations.

> ⓘ There are some minor variations to code format between using Monitoring Configuration or performing custom instrumentation manually. Nevertheless these minor variations are acceptable.

> ⓘ Using custom instrumentation requires knowledge of Java language constructs. Refer to an introductory Java language or programming primer for an explanation of Java language concepts.

> ⓘ Custom instrumentation is only recommended for very specific instrumentation needs. In other cases, it is recommended to use the Monitoring Configuration or Adaptive Instrumentation options, available in the Monitor Settings dialog box (**Settings>Monitor Settings**). For more information regarding these instrumentation options, the "About Monitoring Configuration" section in the *Precise for J2EE User's Guide*.

About modifying instrumenter configuration files

Instrumentation instructions are contained in a series of XML files that can be found in the Precise for J2EE agent installation on the host where the monitored JVM runs. The InstrumenterConfigList.xml file, which also exists in each monitored JVM's configuration directory, references the instrumentation configuration files. See About instrumenter configuration file reference .

How you modify your instrumentation configuration depends on what you try to do:

- To enable a single existing instrumentation configuration that was not enabled by default, you can enable this configuration by uncommenting the reference to the file in the InstrumenterConfigList.xml file.
- To change the instrumentation instructions within a specific instrumentation configuration file, for example to instrument a specific method, you would modify the Custom.xml file (or any other instrumenter configuration XML file). In addition, you must verify that this instrumentation configuration file is enabled in the InstrumenterConfigList.xml file.

> ⓘ By default, the custom instrumentation configuration files are located in the instrumenter directory:
> `<i3_root>/products/i3fp/registry/products/j2ee/config/instrumenter`.
> All relative paths of referenced configuration files are relative to this directory.

You may also copy these files to JVM ID-specific directories to customize them on a per JVM basis. You can use the following special variables to specify the absolute path of these per-JVM configuration files:

- `${indepth.j2ee.home}` expands to `<i3_root>/products/j2ee`
- `${indepth.j2ee.server_id}` expands to the raw JVM ID
- `${indepth.j2ee.jvm_id}` expands to the JVM ID (with serial number)

For example, if you have copied the Custom.xml file into the `<i3_root>/products/j2ee/config/petstore` directory, you can reference it with:

```
<config-file>

    ${indepth.j2ee.home}/config/${indepth.j2ee.server_id}/Custom.xml

  </config-file>
```

Enabling an additional configuration file

The following procedure describes how you can enable an additional configuration file.

To enable an additional configuration file

1. Open the InstrumenterConfigList.xml file in the JVM ID-specific configuration directory for your application server:
   `<i3_root>/products/i3fp/registry/products/j2ee/config/JVMID/`
2. In the InstrumenterConfigList.xml file, locate the XML block that defines the file you want to enable. For example, to enable the JNDI custom instrumentation file, locate the following section:
   <!--
      JNDI
      Uncomment to instrument.
   -->
   <!--
      <config-file>
      JNDI.xml
      </config-file>
   -->
3. Remove the XML comments surrounding the file name. For example:
   <!--
      JNDI
      Uncomment to instrument.
   -->
   <config-file>
      JNDI.xml
   </config-file>

Adding your own custom instrumentation configuration file

The following procedure describes how you can add your own custom instrumentation configuration file.

To add your own custom instrumentation configuration file

1. Create a new user-defined XML configuration file, for example:
   UserDefined.xml, in the following way:
   <?xml version="1.0" encoding="UTF-8"?>
   <!--
      User Defined Instrumenter Configuration File
   -->
   <instrumenter-config>
      <custom-config>
         <java-classes>
            <java-class>
               <class-name>
               </class-name>
               <methods>
                  <method>

```
            <name>
            </name>
        </method>
      </methods>
    </java-class>
  </java-classes>
</custom-config>
</instrumenter-config>
```
2. To add the file to the InstrumenterConfigList.xml file, use the following syntax:
```
<!--
    User defined instrumenter config file
-->
  <config-file>
      UserDefined.xml
  </config-file>
```
3. Modify the new XML configuration file to include the interfaces, classes, and methods you want to instrument. See the next section for more information.

About custom instrumenter configuration

Precise for J2EE allows you great flexibility in instrumenting Java applications. You can instrument specifically named packages, interfaces, classes, and methods. In addition, you can instrument all classes in a package or all methods in a class using the wildcard character. Sub-elements of <instrumenter-config> also allow you to target the behavior of methods so that you can instrument all calls to a method as well as all calls the method makes, or calls from a specific method to another specific method. Precise for J2EE instrumentation also takes advantage of various Java language properties, allowing you, for example, to instrument all classes that extend a common subclass or interface, or to instrument classes and methods marked with a runtime-visible annotation.

> (i) Using custom instrumentation requires knowledge of Java language structures. Refer to an introductory Java language or programming primer for an explanation of Java language concepts.

The Workflow Framework example is used to illustrate how instrumentation is applied to Java language constructs. The Workflow Framework example is based on the following classes and interfaces:

```
package xmp.wfm.task;

@Retention(RetentionPolicy.RUNTIME)
public @interface TransactionProvider {}

@Retention(RetentionPolicy.RUNTIME)
public @interface Transaction {}

@Retention(RetentionPolicy.RUNTIME)
public @interface RecoverableTransactionProvider {}

@Retention(RetentionPolicy.RUNTIME)
public @interface Resource {}

@TransactionProvider
public interface Task {

    @Transaction
    public void start();
    public void stop();
    public void stop(boolean force);
```

```
    }

    @RecoverableTransactionProvider
    public interface RecoverableTask extends Task {
        public void start(RecoverableTaskContext context);
    }

    public abstract class AbstractTask implements Task {

        @Transaction
        public void start() { }
        protected void start(TaskContext taskContext) { }
        public void stop() { }
        public void stop(boolean force) { }

    }

    package xmp.wfm.server;

    @Resource
    public class NonRecoverableTaskAdapter extends AbstractTask {
    }

    @Resource
    public class RecoverableTaskAdapter extends AbstractTask implements RecoverableTask {
        public void start() { }
        public void start(RecoverableTaskContext context) { }
        protected void recover(RecoverableTaskContext context) { }
        public void stop() { }
        public void stop(boolean force) { }
    }
```

About instrumenting interfaces

Instrumentation can be applied to methods in abstract and concrete classes that implement specific interfaces.

About instrumenting methods in implementations of an interface

The <custom-config> element can be used to cause instrumentation to be applied to methods of abstract and concrete classes that implement an interface.

This instrumenter configuration file causes instrumentation to be applied to the start and stop methods of abstract and concrete classes that implement the Task interface.

```xml
<?xml version='1.0'?>
<instrumenter-config>
    <custom-config>
        <java-classes>
            <java-class>
                <class-name> xmp.wfm.task.Task </class-name>
            <methods>
                <method>
                    <name> start </name>
                </method>
                <method>
                    <name> stop </name>
                </method>
```

```
        </methods>
      </java-class>
    </java-classes>
  </custom-config>
</instrumenter-config>
```

Instrumentation is applied to methods of abstract and concrete classes that satisfy the following criteria:

- The matched class implements, directly or indirectly, the interface that is specified in the <class-name> element.
- The name of the method matches the name that is specified in one of the <name> elements.
- The method, including signature, was declared in the matching interface.

Based on these rules, custom-type instrumentation are applied to the following methods:

- `AbstractTask.start()`
- `AbstractTask.stop()`
- `AbstractTask.stop(boolean)`
- `RecoverableTaskAdapter.start()`
- `RecoverableTaskAdapter.stop()`
- `RecoverableTaskAdapter.stop(boolean)`

However, instrumentation is not applied to the following methods:

- `AbstractTask.start(TaskContext)` because the method was not declared in the Task interface.
- `RecoverableTaskAdapter.start(TaskContext)` because the method was not declared in the Task interface.
- `RecoverableTaskAdapter.recover(RecoverableTaskContext)` because the method was not declared in the Task interface.

About restricting instrumentation to methods that match a signature

The <params> element can be included within a <method> element to restrict instrumentation to a method that is based on a signature. See About method signature matching.

This instrumenter configuration file causes instrumentation to be applied to the start and stop(boolean) methods of abstract and concrete classes that implement the Task interface.

```
<?xml version='1.0'?>
<instrumenter-config>
  <custom-config>
    <java-classes>
      <java-class>
        <class-name> xmp.wfm.task.Task </class-name>
        <methods>
          <method>
            <name> start </name>
          </method>
          <method>
            <name> stop </name>
            <params>
              <param> boolean </param>
            </params>
          </method>
        </methods>
      </java-class>
    </java-classes>
```

```
    </custom-config>
  </instrumenter-config>
```

Instrumentation is applied to methods of abstract and concrete classes that satisfy the following criteria:

- The matched class implements, directly or indirectly, the interface that is specified in the <class-name> element.
- The name of the method matches the name that is specified in the <name> element.
- The method was declared in the matching interface.

Based on these rules, custom-type instrumentation is applied to the following methods:

- `AbstractTask.start()`
- `AbstractTask.stop(boolean)`
- `RecoverableTaskAdapter.start()`
- `RecoverableTaskAdapter.stop(boolean)`

However, instrumentation is not applied to the following methods:

- `AbstractTask.start(TaskContext)` because the method was not declared in the Task interface.
- `AbstractTask.stop()` because the method does not match the specified signature of (boolean).
- `RecoverableTaskAdapter.start(TaskContext)` because the method was not declared in the Task interface.
- `RecoverableTaskAdapter.recover(RecoverableTaskContext)` because the method was not declared in the Task interface.
- `RecoverableTaskAdapter.stop()` because the method does not match the specified signature of (boolean). See About method signature matching.

About extending instrumentation to interfaces matching a wildcard

Inheritance is not considered when wildcards are used with <class-name> element. It is not possible to use wildcards to cause instrumentation to be applied to all abstract or concrete class's that implement interface's that match a wildcard pattern.

However, wildcards can be used to apply instrumentation to all abstract or concrete classes whose names match the wildcard pattern that is specified in the <class-name> element.

See About using the wildcard character *.

About extending instrumentation to methods that match a wildcard

Wildcards can be used in the <name> element. This instrumenter configuration file causes instrumentation to be applied to the start() and stop() methods of abstract and concrete classes that implement the Task interface.

```
<?xml version='1.0'?>
<instrumenter-config>
  <custom-config>
    <java-classes>
      <java-class>
        <class-name> xmp.wfm.task.Task </class-name>
        <methods>
          <method>
            <name> s* </name>
            <params> <params/>
          </method>
        </methods>
      </java-class>
    </java-classes>
```

```
    </custom-config>
  </instrumenter-config>
```

When wildcards are used in the <name> element and the matching construct is an interface, instrumentation is applied to methods of abstract and concrete classes that satisfy the following criteria:

- The name of the method matches the wildcard pattern that is specified in the <name> element.
- The signature of the method matches the signature that is specified in the <params> element (if present). Specifying empty <params> </params> elements indicates that methods with zero arguments are matched, as in a signature of (). If <params> </params> is omitted, all signatures are matched.
- The method is declared in the matched interface.

Based on these rules, custom-type instrumentation is applied to the following methods:

- `AbstractTask.start()`
- `AbstractTask.stop()`
- `RecoverableTaskAdapter.start()`
- `RecoverableTaskAdapter.stop()`

However, instrumentation is not applied to the following methods:

- `AbstractTask.start(TaskContext)` because the method was not declared in the Task interface.
- `AbstractTask.stop(boolean)` because the method does not match the specified signature of ().
- `RecoverableTaskAdapter.start(TaskContext)` because the method was not declared in the Task interface.
- `RecoverableTaskAdapter.recover(RecoverableTaskContext)` because the method name does not match the wildcard pattern.
- `RecoverableTaskAdapter.stop(boolean)` because the method does not match the signature of ().

About extending instrumentation to methods that are declared in extending interfaces

By default, instrumentation is restricted to methods that are declared in the matching interface. The <apply-to-subtypes> element can be used to extend instrumentation to apply to methods that are declared in extending interfaces.

This instrumenter configuration file causes instrumentation to be applied to the start methods of abstract and concrete classes that implement the Task interface or an interface that extends the Task interface.

```
    <?xml version='1.0'?>
    <instrumenter-config>
      <custom-config>
        <java-classes>
          <java-class>
            <class-name> xmp.wfm.task.Task </class-name>
            <methods>
              <method>
                <name> sta* </name>
                <apply-to-subtypes>true</apply-to-subtypes>
              </method>
            </methods>
          </java-class>
        </java-classes>
      </custom-config>
    </instrumenter-config>
```

Instrumentation is applied to methods of abstract and concrete classes that satisfy the following criteria:

- The name of the method matches the wildcard pattern that is specified in the <name> element.

- The signature of the method matches the signature that is specified in the <params> element (if present).
- The method is declared in the matched interface.
- The matched class has one or more interfaces are the same as or extend the interface that is specified in the <class-name> element.

Based on these rules, custom-type instrumentation is applied to the following methods:

- `AbstractTask.start()`
- `RecoverableTaskAdapter.start()`
- `RecoverableTaskAdapter.start(RecoverableTaskContent)`

However, instrumentation is not applied to the following methods:

- `AbstractTask.start(TaskContext)` because the method was not declared in the Task interface or in an interface that extends the Task interface.
- `AbstractTask.stop()` because the method does not match the specified wildcard pattern of sta*.
- `AbstractTask.stop(boolean)` because the method does not match the specified wildcard pattern of sta*.
- `RecoverableTaskAdapter.recover(RecoverableTaskContext)` because the method name does not match the specified wildcard pattern of sta*.
- `RecoverableTaskAdapter.stop()` because the method does not match the specified wildcard pattern of sta*.
- `RecoverableTaskAdapter.stop(boolean)` because the method does not match the specified wildcard pattern of sta*.

About extending wildcards to apply to methods that are declared in implementations

By default, instrumentation is restricted to methods that are declared in the matching interface. The <apply-to-implementations> element can be used to extend instrumentation to apply to methods that are declared in abstract and concrete classes that implement the matching interface.

This instrumenter configuration file causes instrumentation to be applied to the start methods of abstract and concrete classes that implement the Task interface.

```
<?xml version='1.0'?>
<instrumenter-config>
  <custom-config>
    <java-classes>
      <java-class>
        <class-name> xmp.wfm.task.Task </class-name>
        <methods>
          <method>
            <name> sta* </name>
            <apply-to-implementations>true
            </apply-to-implementations>
          </method>
        </methods>
      </java-class>
    </java-classes>
  </custom-config>
</instrumenter-config>
```

Instrumentation is applied to methods of abstract and concrete classes that satisfy the following criteria:

- The name of the method matches the wildcard pattern that is specified in the <name> element.
- The signature of the method matches the signature that is specified in the <params> element (if present).

- The method is declared in a class that extends the abstract class or implements the interface that is identified in the <class-name> element.

Based on these rules, custom-type instrumentation is applied to the following methods:

- `AbstractTask.start()`
- `AbstractTask.start(TaskContent)`
- `RecoverableTaskAdapter.start()`
- `RecoverableTaskAdapter.start(RecoverableTaskContent)`

However, instrumentation is not applied to the following methods:

- `AbstractTask.stop()` because the method does not match the specified wildcard pattern of sta*.
- `AbstractTask.stop(boolean)` because the method does not match the specified wildcard pattern of sta*.
- `RecoverableTaskAdapter.recover(RecoverableTaskContext)` because the method name does not match the specified wildcard pattern of sta*.
- `RecoverableTaskAdapter.stop()` because the method does not match the specified wildcard pattern of sta*.
- `RecoverableTaskAdapter.stop(boolean)` because the method does not match the specified wildcard pattern of sta*.

About instrumenting classes

Instrumentation can be applied to methods in abstract and concrete classes.

About instrumenting methods in abstract and concrete classes

The <custom-config> element can be used to cause instrumentation to be applied to methods of abstract and concrete classes.

This instrumenter configuration file causes instrumentation to be applied to the start and stop methods of abstract and concrete classes that extend the AbstractTask class.

```xml
<?xml version='1.0'?>
<instrumenter-config>
  <custom-config>
    <java-classes>
      <java-class>
        <class-name> xmp.wfm.task.AbstractTask </class-name>
        <methods>
          <method>
            <name> start </name>
          </method>
          <method>
            <name> stop </name>
          </method>
        </methods>
      </java-class>
    </java-classes>
  </custom-config>
</instrumenter-config>
```

Instrumentation is applied to methods of abstract and concrete classes that satisfy the following criteria:

- The matched class extends the class specified in the <class-name> element.
- The name of the method matches the name that is specified in one of the <name> elements.

- The method was declared in the matching class.

Based on these rules, custom-type instrumentation is applied to the following methods:

- `AbstractTask.start()`
- `AbstractTask.start(TaskContent)`
- `AbstractTask.stop()`
- `AbstractTask.stop(boolean)`
- `RecoverableTaskAdapter.start()`
- `RecoverableTaskAdapter.start(TaskContent)`
- `RecoverableTaskAdapter.stop()`
- `RecoverableTaskAdapter.stop(boolean)`

However, instrumentation is not applied to the following methods:

- `RecoverableTaskAdapter.recover(RecoverableTaskContent)` because the method was not declared in the AbstractTask class

About restricting instrumentation to methods that match a signature

The <params> element can be included within a <method> element to restrict instrumentation to a method that is based on signature.

This instrumenter configuration file causes instrumentation to be applied to the start and stop(boolean) methods of abstract and concrete classes that extend the AbstractTask class.

```
<?xml version='1.0'?>
<instrumenter-config>
  <custom-config>
    <java-classes>
      <java-class>
        <class-name> xmp.wfm.task.AbstractTask </class-name>
        <methods>
          <method>
            <name> start </name>
          </method>
          <method>
            <name> stop </name>
            <params>
              <param> boolean </param>
            </params>
          </method>
        </methods>
      </java-class>
    </java-classes>
  </custom-config>
</instrumenter-config>
```

Instrumentation is applied to methods of abstract and concrete classes that satisfy the following criteria:

- The matched class extends the interface that is specified in the <class-name> element.
- The name of the method matches the name that is specified in the <name> element.
- The method signature matches the types that are specified using the <param> elements.
- The method was declared in the matching class.

Based on these rules, custom-type instrumentation is applied to the following methods:

- `AbstractTask.start()`

- `AbstractTask.start(TaskContent)`
- `AbstractTask.stop(boolean)`
- `RecoverableTaskAdapter.start()`
- `RecoverableTaskAdapter.start(TaskContent)`
- `RecoverableTaskAdapter.stop(boolean)`

However, instrumentation is not applied to the following methods:

- `AbstractTask.stop()` because the method does not match the specified signature of (boolean).
- `RecoverableTaskAdapter.recover(RecoverableTaskContext)` because the method was not declared in the AbstractTask class.
- `RecoverableTaskAdapter.stop()` because the method does not match the specified signature of (boolean) See About method signature matching.

About extending instrumentation to classes matching a wildcard

Inheritance is not considered when wildcards are used with a <class-name> element. It is not possible to use wildcards to cause instrumentation to be applied to all abstract or concrete classes that extend classes that match a wildcard pattern.

However, wildcards can be used to apply instrumentation to all abstract or concrete classes whose names match the wildcard pattern that is specified in the <class-name> element.

About extending instrumentation to methods that match a wildcard

Wildcards can be used in the <name> element.

This instrumenter configuration file causes instrumentation to be applied to the start() and stop() methods of abstract and concrete classes that extend the AbstractTask class. See About using the wildcard character *.

```
<?xml version='1.0'?>
<instrumenter-config>
    <custom-config>
        <java-classes>
            <java-class>
                <class-name> xmp.wfm.task.AbstractTask </class-name>
                <methods>
                    <method>
                        <name> s* </name>
                        <params> <params/>
                    </method>
                </methods>
            </java-class>
        </java-classes>
    </custom-config>
</instrumenter-config>
```

When wildcards are used in the <name> element and the matching construct is a class, instrumentation is applied to methods of abstract and concrete classes that satisfy the following criteria:

- The name of the method matches the wildcard pattern that is specified in the <name> element.
- If the <params> element is present, the signature of the method matches the signature that is specified in the <params> element. Specifying empty <params> </params> indicates that methods with zero arguments are matched, as in a signature of ( ). If <params> </params> is omitted, all signatures are matched.
- The method is declared in the matched class.

Based on these rules, custom-type instrumentation is applied to the following methods:

- `AbstractTask.start()`
- `AbstractTask.start(TaskContent)`
- `AbstractTask.stop()`
- `RecoverableTaskAdapter.start()`
- `RecoverableTaskAdapter.start(TaskContent)`
- `RecoverableTaskAdapter.stop()`

However, instrumentation is not applied to the following methods:

- `AbstractTask.stop(boolean)` because the method does not match the specified signature of ().
- `RecoverableTaskAdapter.recover(RecoverableTaskContext)` because the method name does not match the wildcard pattern.
- `RecoverableTaskAdapter.stop(boolean)` because the method does not match the specified signature of ().

About extending instrumentation to methods that are declared in extending classes

By default, instrumentation is restricted to methods that are declared in the matching class. The <apply-to-subtypes> element can be used to extend instrumentation to apply to methods that are declared in extending classes.

This instrumenter configuration file causes instrumentation to be applied to the start methods of abstract and concrete classes that extend the AbstractTask class or a class that extends the AbstractTask class.

```xml
<?xml version='1.0'?>
<instrumenter-config>
    <custom-config>
        <java-classes>
            <java-class>
                <class-name> xmp.wfm.task.AbstractTask </class-name>
                <methods>
                    <method>
                    <name> sta* </name>
                    <apply-to-subtypes>true</apply-to-subtypes>
                    </method>
                </methods>
            </java-class>
        </java-classes>
    </custom-config>
</instrumenter-config>
```

Instrumentation is applied to methods of abstract and concrete classes that satisfy the following criteria:

- The name of the method matches the wildcard pattern that is specified in the <name> element.
- The signature of the method matches the signature that is specified in the <params> element (if present).
- The method is declared in the matched class.
- The matched class is the same as or extends the class specified in the <class-name> element.

Based on these rules, custom-type instrumentation is applied to the following methods:

- `AbstractTask.start()`
- `AbstractTask.start(TaskContent)`
- `RecoverableTaskAdapter.start()`
- `RecoverableTaskAdapter.start(RecoverableTaskContent)`

However, instrumentation is not applied to the following methods:

- `AbstractTask.stop()` because the method does not match the specified wildcard pattern of sta*.

- `AbstractTask.stop(boolean)` because the method does not match the specified wildcard pattern of sta*.
- `RecoverableTaskAdapter.recover(RecoverableTaskContext)` because the method does not match the specified wildcard pattern of sta*.
- `RecoverableTaskAdapter.stop()` because the method does not match the specified wildcard pattern of sta*.
- `RecoverableTaskAdapter.stop(boolean)` because the method does not match the specified wildcard pattern of sta*.

About instrumenting using annotations

Instrumentation can be applied to classes and interfaces that are annotated with a specific annotation

About instrumenting annotated classes and interfaces

The <annotation-name> element can be added to the <java-class> element to cause instrumentation to be applied to classes that are annotated with a specific annotation.

This instrumenter configuration file causes instrumentation to be applied to the start and stop methods of abstract and concrete classes that extend a class or implement an interface that is annotated with the TransactionProvider annotation.

```
<?xml version='1.0'?>
<instrumenter-config>
   <custom-config>
      <java-classes>
         <java-class>
            <class-name> * </class-name>
            <annotation-name> xmp.wfm.task.TransactionProvider </annotation-name>
            <methods>
               <method>
                  <name> start </name>
               </method>
               <method>
                  <name> stop </name>
               </method>
            </methods>
         </java-class>
      </java-classes>
   </custom-config>
</instrumenter-config>
```

Instrumentation is applied to methods of abstract and concrete classes that fill the following criteria:

- The matched class extends a class that is annotated with the annotation specified in the <annotation-name> element.
- The name of the method matches the name that is specified in the <name> element.

Based on these rules, custom-type instrumentation is applied to the following methods:

- `AbstractTask.start`
- `AbstractTask.stop`
- `RecoverableTaskAdapter.start`
- `RecoverableTaskAdapter.stop`

About instrumenting annotated methods

The <annotation-name> element can be added to the <method> element to cause instrumentation to be applied to methods that are annotated with a specific annotation.

This instrumenter configuration file causes instrumentation to be applied to all methods in the application that are annotated with @Transaction:

```
<?xml version='1.0'?>
<instrumenter-config>
   <custom-config>
      <java-classes>
         <java-class>
            <class-name> * </class-name>
            <methods>
               <method>
                  <name> * </name>
                  <annotation-name> xmp.wfm.task.Transaction</annotation-name>
               </method>
            </methods>
         </java-class>
      </java-classes>
   </custom-config>
</instrumenter-config>
```

Instrumentation is applied to methods that fill the following criteria:

- The method is annotated with `@Transaction`

Based on these rules, custom-type instrumentation is applied to the following methods:

- `AbstractTask.start`

About instrumenting annotated methods in implementing or inheriting classes

The <apply-to-subtypes> element can be added to the <method> element along with the <annotation-name> element, to cause classes that override or implement an annotated method to be instrumented.

This instrumenter configuration file causes instrumentation to be applied to all methods in the application that are annotated with @Transaction, and to implementing and overriding methods:

```
<?xml version='1.0'?>
<instrumenter-config>
   <custom-config>
      <java-classes>
         <java-class>
            <class-name> * </class-name>
            <methods>
               <method>
                  <name> * </name>
                  <annotation-name> xmp.wfm.task.Transaction</annotation-name>
                  <apply-to-subtypes>true</apply-to-subtypes>
               </method>
            </methods>
         </java-class>
      </java-classes>
```

```
    </custom-config>
  </instrumenter-config>
```

Instrumentation is applied to methods that fill the following criteria:

- The method is annotated with `@Transaction`
- Or - the method implements an interface method that is annotated with `@Transaction`
- Or - the method overrides a method that is annotated with `@Transaction`

Based on these rules, custom-type instrumentation is applied to the following methods:

- `AbstractTask.start`
- `RecoverableTaskAdapter.start`

About instrumenting only classes that are not assignable to a class or interface

The < not-assignable-to> element can be added to the <java-class> element to prevent the rule from being applied to classes that are assignable to specific types.

This instrumenter configuration file causes instrumentation to be applied to all the classes in the xmp.wfm.task package, except for the ones that are assignable to RecoverableTask:

```
<?xml version='1.0'?>
<instrumenter-config>
  <custom-config>
    <java-classes>
      <java-class>
        <class-name> xmp.wfm.task.* </class-name>
        <not-assignable-to> xmp.wfm.task.RecoverableTask </not-assignable-to>
        <methods>
          <method>
            <name> start </name>
          </method>
        </methods>
      </java-class>
    </java-classes>
  </custom-config>
</instrumenter-config>
```

Instrumentation is applied to methods that satisfy the following criteria:

- The class is in the package `xmp.wfm.task`
- The class is not assignable to `xmp.wfm.task.RecoverableTask`

Based on these rules, custom-type instrumentation is applied to the following methods:

- `AbstractTask.start`

About instrumenting all calls from a method

Instrumentation can be applied to all calls from methods that match criteria specified in a <java-classes> element. This instrumenter configuration file causes instrumentation to be applied to all calls from the methods that match the <java-classes> element.

```
<?xml version='1.0'?>
<instrumenter-config>
  <all-calls-from-method>
    <java-classes> ]
```

```
            ...
        </java-classes>
      </all-calls-from-method>
    </instrumenter-config>
```

The rules that apply to the <java-classes> element that is documented in preceding sections are applied when it is used inside the <all-calls-from-method> element.

About instrumenting calls to methods

The <all-calls-to-method> element can be used to apply instrumentation to all calls to specific methods.

This instrumenter configuration file causes instrumentation to be applied to all calls made to printIn methods in the java.io. package or to run () methods.

```
    <?xml version='1.0'?>
    <instrumenter-config>
      <all-calls-to-method>
        <methods>
          <method>
            <name> java.io.*.println </name>
          </method>
          <method>
            <name> *.run </name>
            <params> <params/>
          </method>
        </methods>
      </all-calls-to-method>
    </instrumenter-config>
```

The <name> element has a different interpretation from its use in all other cases. When the <name> element is used inside an <all-calls-to-method> element, it must represent a fully-qualified method name. The portion of the <name> after the last dot (".") is considered to be the method name, and the portion of the <name> before the last dot is considered to be the class name. When you use wildcards, it is important to use a wildcard pattern that fits the scheme described. For example, the wildcard pattern *.set* matches all methods whose name starts with set of all classes, and xmp.task.*.* matches all methods in all classes whose name starts with xmp.task.

The <invocation-relationship> element can be used instead of the <all-calls-to-method> element to restrict instrumentation to calls to specific methods.

This instrumenter configuration file causes instrumentation to be applied to all calls made to println methods in the java.io. package or to run() methods from methods in abstract or concrete classes that implement the AbstractTask interface.

```
    <?xml version='1.0'?>
    <instrumenter-config>
      <invocation-relationship>
        <java-class>
          <class-name> xmp.task.AbstractTask </class-name>
        </java-class>
        <invoked-method> java.io.*.println </invoked-method>
        <invoked-method> *.run() </invoked-method>
      </invocation-relationship>
    </instrumenter-config>
```

The rules that apply to the <java-class> element that is documented in preceding sections are applied when it is used inside the <invocation-relationship> element.

The <invoked-method> represents a fully-qualified method name. The portion of the <invoked-method> after the last dot (".") is considered to be the method name, and the portion of the <invoked-method> before the last dot is considered to be the class name. When you use wildcards, it is important to use a wildcard pattern that fits the scheme described. For example, the wildcard pattern *.set* matches all methods whose name starts with set of all classes, and xmp.task.*.* matches all methods in all classes whose name starts with xmp.task.

The <invoked-method> may also include a method signature. The method signature should follow these rules:

- The method signature must be enclosed in parentheses ("(" and ")").
- The method signature must not contain spaces.
- A comma (",") must separate parameter types in the method signature.
- The method signature may include wildcards.

About preventing instrumentation for classes, methods, and calls to methods

Instrumentation can be prevented for specific packages and sub-packages, classes, methods, or calls to methods. The <ignore-config> element in instrumenter configuration files is used to illustrate how to prevent instrumentation from being applied to specific packages and sub-packages, classes, method, or calls to methods. See   About instrumenter configuration file reference .

As with the <custom-config>, <all-calls-from-method>, <all-calls-to-method>, and <calls-from-method-to-method> elements, the <ignore-config> element can be placed in its own instrumenter configuration file. The examples should be considered to imply that the <ignore-config> element must be included in an instrumenter configuration file that contains other instrumentation directives.

About preventing instrumentation for all methods of classes in a package and sub-packages

The <java-classes> element can be used within an <ignore-config> element to prevent instrumentation from being applied to all methods of classes in a package and sub-packages using wildcards in the <class-name>.

This instrumenter configuration file prevents instrumentation from being applied to the AbstractTask class:

```
<?xml version='1.0'?>
<instrumenter-config>
    <custom-config>
        <java-classes>
            <java-class>
                <class-name> xmp.wfm.* </class-name>
                <methods>
                    <method>
                        <name> * </name>
                    </method>
                </methods>
            </java-class>
        </java-classes>
    </custom-config> ]
    <ignore-config>
        <java-classes>
            <java-class>
                <class-name> xmp.wfm.task.* </class-name>
            </java-class>
        </java-classes>
    </ignore-config>
</instrumenter-config>
```

Based on these rules, instrumentation is not applied to the following methods:

- `AbstractTask.start()`
- `AbstractTask.start(TaskContext)`
-
      `AbstractTask.stop()`
-
      `AbstractTask.stop(boolean)`

However, instrumentation is still applied to the following methods:

- `RecoverableTaskAdapter.start()`
- `RecoverableTaskAdapter.stop()`
- `RecoverableTaskAdapter.stop(boolean)`

About preventing instrumentation for methods of a class

The <java-classes> element can be used within an <ignore-config> element to prevent instrumentation from being applied to some or all methods of abstract or concrete classes, or to all methods of a class in a specific package and its sub-packages.

This instrumenter configuration file prevents instrumentation from being applied to the AbstractTask class:

```
<?xml version'1.0'?>
<instrumenter-config>
   <custom-config>
      <java-classes>
         <java-class>
            <class-name> xmp.wfm.task.AbstractTask </class-name>
            <methods>
               <method>
                  <name> * </name>
               </method>
            </methods>
         </java-class>
      </java-classes>
   </custom-config>
   <ignore-config>
      <java-classes>
         <java-class>
            <class-name> xmp.wfm.task.AbstractTask </class-name>
         </java-class>
      </java-classes>
   </ignore-config>
</instrumenter-config>
```

Based on this configuration, instrumentation is not applied to the following methods:

- `AbstractTask.start()`
- `AbstractTask.start(TaskContext)`
-
      `AbstractTask.stop()`
-
      `AbstractTask.stop(boolean)`

However, instrumentation is applied to the following methods:

- `RecoverableTaskAdapter.start()`
- `RecoverableTaskAdapter.stop()`

- `RecoverableTaskAdapter.stop(boolean)`

The <methods> element can be supplied to prevent instrumentation from being applied to some methods of a class, but not others.

This instrumenter configuration file prevents instrumentation from being applied to all stop() methods:

```
<?xml version'1.0'?>
<instrumenter-config>
    <custom-config>
        <java-classes>
            <java-class>
                <class-name> xmp.wfm.task.Task </class-name>
                <methods>
                    <method>
                        <name> * </name>
                    </method>
                </methods>
            </java-class>
        </java-classes>
    </custom-config>
    <ignore-config>
        <java-classes>
            <java-class>
                <class-name> * </class-name>
                <methods>
                    <method>
                        <name> stop </name>
                        <params> <params/>
                    </method>
                </methods>
            </java-class>
        </java-classes>
    </ignore-config>
</instrumenter-config>
```

Based on this configuration, instrumentation is not applied to the following methods:

-
        `AbstractTask.stop()`
-
        `RecoverableTaskAdapter.stop()`

However, instrumentation is applied to the following methods:

- `AbstractTask.start()`
-
        `AbstractTask.stop(boolean)`
-
        `RecoverableTaskAdapter.start()`
-
        `RecoverableTaskAdapter.stop(boolean)`

About preventing instrumentation for calls from a method

The <all-calls-to-method> element can be used within an <ignore-config> element to prevent instrumentation from being applied to all calls to specific methods.

This instrumenter configuration file prevents instrumentation from being applied to all calls to Java runtime classes (java.*) methods.

```
<?xml version'1.0'?>
<instrumenter-config>
    <ignore-config>
        <java-classes>
            <all-calls-to-method>
            <methods>
            <method>
            <name> java.* </name>
            </method>
            </methods>
            </all-calls-to-method>
        </java-classes>
    </ignore-config>
</instrumenter-config>
```

The <all-calls-to-method> element is used within an <ignore-config> element, the <name> element of <method> elements is expected to be a qualified method name. The last dot in the <name> separates the class name from the method name.

This is also true if wildcards are used. This instrumenter configuration file prevents instrumentation from being applied to all calls to methods in the java class.

```
<?xml version'1.0'?>
<instrumenter-config>
    <ignore-config>
        <java-classes/>
            <all-calls-to-method>
                <methods>
                    <method>
                        <name> java.* </name>
                    </method>
                </methods>
            </all-calls-to-method>
        </java-classes>
    </ignore-config>
</instrumenter-config>
```

About preventing instrumentation for calls to a method

The <invocation-relationship> element can be used within an <ignore-config> element to prevent instrumentation from being applied to calls to specific methods from specific calling methods.

This instrumenter configuration file prevents instrumentation from being applied to all calls to Java runtime classes (java.*) methods that take an int[] parameter from the stop() methods in the xmp.wfm package and sub-packages.

```
<?xml version='1.0'?>
<instrumenter-config>
    <ignore-config>
        <java-classes/>
            <invocation-relationship>
                <java-class>
                    <class-name> xmp.wfm.* </class-name>
                    <methods>
```

```
                <method>
                    <name> stop </name>
                    <params> <params/>
                </method>
            </methods>
        </java-class>
        <invoked-method> java.*.*(int[]) </invoked-method>
    </invocation-relationship>
</java-classes>
</ignore-config>
</instrumenter-config>
```

About instrumenting calls to EJB business method implementations

The J2EE specification does not require that an EJB implementation implement the remote interface directly. Application servers typically generate a skeleton that implements the remote interface directly and delegates calls to the EJB implementation.

Because there is no direct relationship between an EJB implementation and the remote interface, it is difficult to instrument the business methods of an EJB implementation without first identifying the remote interfaces and then manually instrumenting the methods.

About the Calls to EJB instrumentation feature

The Calls to EJB implementations instrumentation feature adds a configurable extension to the instrumenter to allow calls to EJB implementations to be instrumented.

When the feature is enabled, the Calls to EJB implementations instrumentation feature searches for classes that match certain marker classes or interfaces or name patterns. If a class matches, caller-side instrumentation is applied to any calls from a method that makes a call to another method of the same name and signature.

The marker classes and interfaces are expected to identify the EJB skeletons. As an example, consider these application classes:

```
public interface Account extends EJBObject {
    public void deposit(float amount);
}
public class AccountEJB implements SessionBean {
    public void deposit(float amount)
        updateAccount(getCurrentBalance() + amount);
    }
}
```

Under WebLogic 7.0, EJB skeletons extend the weblogic.rmi.internal.Skeleton class. The "Calls to EJB implementations" instrumentation feature can be configured to find the skeleton and instrument the call to deposit. As an example, the WebLogic skeleton might look as follows:

```
public class AccountEJB_bfqop_WKSkel
extends Skeleton
implements Account {
    public void init() { /* ... */ }
    public void invoke() { /* ... */ }
    public void deposit(float amount) {
        impl.deposit(amount);
```

```
        }
    }
```

In this case, the call from AccountEJB_bfqop_WKSkel.deposit to AccountEJB.deposit is instrumented because the call is to a method with the same name and signature as the caller.

> ⓘ   init() and invoke() are probably not instrumented unless they call another method with the same name and signature.

Instrumentation that is applied by the "Calls to EJB implementations" instrumentation feature show up in the instrumenter log as FIXED-CALLER-SIDE instrumentation.

The EJBImpl.xml file looks like the following example:

```
<instrumenter-config>
  <planner-config>
    <class-name>com.precise.javaperf.instrument.planner. CallsToEJBImplementationPlanner
    </class-name>
    <property>
      <name> skeletonMarkerClass </name>
      <value> weblogic.ejb20.internal.BaseEJBObject </value>
    </property>
    <property>
      <name> logBeginMethodName </name>
      <value> logBeginEjbServer </value>
    </property>
    <property>
      <name> logEndMethodName </name>
      <value> logEndEjbServer </value>
    </property>
    <property>
      <name> invocationType </name>
      <value> EJBImpl </value>
    </property>
  </planner-config>
</instrumenter-class>
```

The <class-name> element identifies the name of the new "pluggable instrumentation planner." The following table lists the recognized properties.

Table 18-1 Recognized properties

Property name	Multiplicity	Description
logBeginMethodName	Optional	Identifies the logger method to call to report calls to EJB business methods.
logEndMethodName	Optional	Identifies the logger method to call to report calls to EJB business methods.
invocationType	Optional	Identifies how the logger events show up in Precise for J2EE.

Property name	Multiplicity	Description
skeletonMarkerClass	Any Number	Identifies the classes that an EJB skeleton must extend. Only classes that extend a skeleton class or implement a skeleton interface are searched for calls to methods with matching name and signature.
skeletonMarkerInterface	Any Number	Identifies the classes that an EJB skeleton must implement. Only classes that extend a skeleton class or implement a skeleton interface are searched for calls to methods with matching name and signature.
implementationMarkerClass	Any Number	Identifies the classes that must be extended by calls with a matching name and signature.
implementationMarkerInterface	Any Number	Identifies the classes that must be implemented by calls with a matching name and signature.

Applying instrumentation using the "Calls to EJB" instrumentation feature

The following procedure describes how to apply instrumentation using the "Calls to EJB" instrumentation feature.

To apply instrumentation using the "Calls to EJB" instrumentation feature

1. Add the EJBImpl.xml file to the InstrumenterConfigList.xml file:
   ```
   <instrumenter-config-list>
      <config-file>
         EJBImpl.xml
      </config-file>
   </instrumenter-config-list>
   ```
2. Restart the application server.

About instrumenter configuration file reference

The purpose and schema of the various files that configure instrumentation for a monitored JVM is described in About the master configuration file, About the instrumenter configuration files, About the structure of instrumenter configuration files, and About common instrumenter configuration matching techniques.

Figure 18-1 Instrumenter Configuration
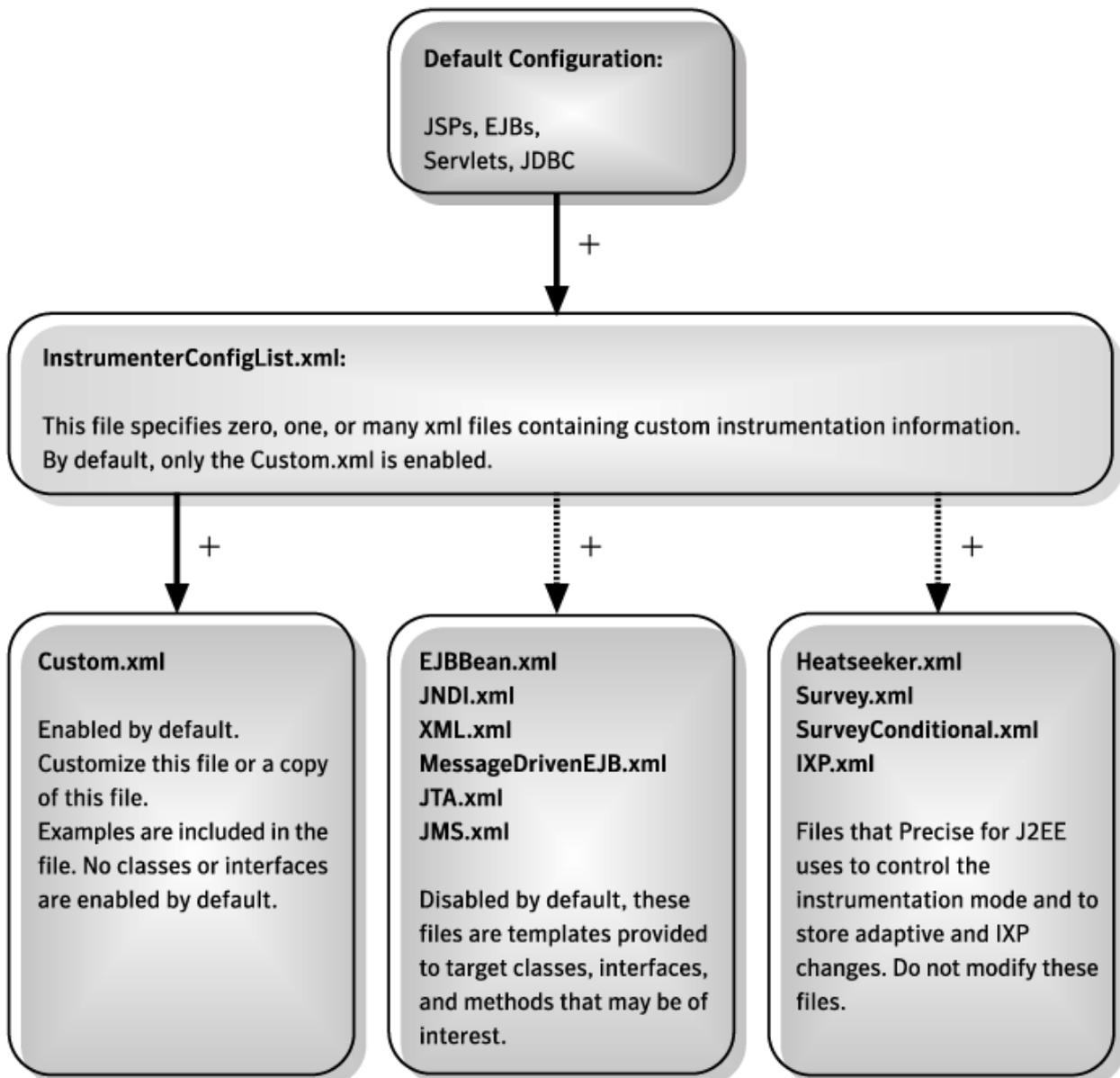
About the master configuration file

Instrumentation for each monitored JVM is configured from a master configuration file. The master configuration file, InstrumenterConfigList.xml, contains references to instrumenter configuration files that contain specific rules that are used to determine where and how to apply instrumentation to a monitored JVM.

The master configuration file can be found under the JVM-specific config directory for a monitored JVM. The master configuration file has the following structure:

```
<?xml version='1.0'?>
<instrumenter-config-list> ]
    <config-file> <!-- occurs 0 or more times -->
        instrumenter-config-file-path
```

```
    </config-file>
  </instrumenter-config-list>
```

The instrumenter config file path specifies the absolute or relative path to an instrumenter configuration file. Paths are relative to

`<i3_root>/products/i3fp/registry/products/j2ee/config/instrumenter` .

The following special variables can be used in the instrumenter config file path:

- `${indepth.j2ee.home} expands to  <i3_root>/products/j2ee`
- `${indepth.j2ee.server_id} expands to the JVM ID (with no sequence number)`
- `${indepth.j2ee.jvm_id} expands to the JVM ID (with a  sequence number)`

About the instrumenter configuration files

Instrumenter configuration files contain specific rules that are used to determine where and how to apply instrumentation. There are several default instrumenter configuration files located in the

`<i3_root>/products/i3fp/registry/products/j2ee/config/instrumenter` and
`<i3_root>/products/i3fp/registry/products/j2ee/config/instrumenter/sample directories.`

The following table lists and describes these files:

Table 18-2 Instrumentation configuration files

File name	Description
Logger.xml	Product configuration.
Planners.xml	Controls the order in which instrumentation is applied.
Heatseeker.xml	Adaptive instrumentation analysis results that are loaded at startup. This file is regenerated when adaptive instrumentation policies are run.
Ixp.xml	Instrumentation Explorer applied changes that are loaded at startup. This file is regenerated when you click the Instrumentation Explorer Apply Changes button.
Survey.xml	Adaptive survey instrumentation configuration.
SurveyConditional.xml	Adaptive conditional instrumentation configuration.
SurveySynchronization.xml	Adaptive synchronization instrumentation.
Servlet.xml	Default Java Servlet instrumentation configuration
GenericPortal.xml	Default Generic Portal-specific instrumentation configuration. Detects the portal and portlet configurations that implement javax.portlet.GenericPortlet.
GenericPortlet.xml	Default Generic Portlet-specific instrumentation configuration. Instruments the portlet lifecycle and action methods.
JSP.xml	Default JSP instrumentation configuration.

File name	Description
WebLogicJSP.xml	Default (BEA WebLogic™-specific) JSP instrumentation configuration.
WebSphereJSP.xml	Default (IBM® WebSphere®-specific) JSP instrumentation configuration.
WebLogicPortal.xml	Default BEA WebLogic Portal-specific instrumentation configuration. Detects the portal and portlet configuration. This file is populated when an application server portal version is selected in Precise Framework Installer.
WebLogicPortlet.xml	Default BEA WebLogic Portlet-specific instrumentation configuration. Instruments the detected portlets. This file is populated when an application server portal version is selected in Precise Framework Installer.
EJB.xml	Default EJB instrumentation configuration. Handles instrumentation of EJB stubs.
Ignore.xml	Default "ignored" instrumentation configuration.
CallsToJDBC.xml	Default "caller-side" JDBC instrumentation configuration.
OverInstrumentationProtection.xml	Over-instrumentation protection instrumentation configuration.
IndepthWeb.xml	Default instrumentation configuration that Precise for Web uses. This file is populated when Precise for Web is installed.
TACPeopleSoft.xml	Insight SmartLink for PeopleSoft instrumentation configuration.
TACWebApps.xml	Insight SmartLink for Web applications instrumentation configuration
Custom.xml	Contains an example instrumentation only. You should use this file as an example and edit this file.
CallsFromMethodToMethod.xml	Contains an example instrumentation only. You must edit this file.
LeakSeeker.xml	Configuration for Leak Seeker instrumentation.
WebLogicEJB.xml	Handles BEA WebLogic-specific EJB "lifecycle operation" instrumentation.
WebSphereEJB.xml	Handles IBM WebSphere-specific EJB "lifecycle operation" instrumentation.
OracleEJB.xml	Handles Oracle-specific EJB "lifecycle operation" instrumentation.
JNDI.xml	Default JNDI instrumentation configuration.

File name	Description
DataSource.xml	Default JDBC DataSource instrumentation configuration.
EJBBean.xml	Default EJB implementation instrumentation.
JTA.xml	Default Java Transaction instrumentation.
MessageDrivenEJB.xml	Default Message-Driven EJB instrumentation.
JMS.xml	Default Java Messaging Service instrumentation.
XML.xml	Default XML and XSL instrumentation.
Calls.xml	Template for configuring all calls to and all calls from methods.
EJBImpl.xml	Sample EJB implementation instrumentation.
Jolt.xml	Sample Jolt instrumentation.
MBeanImpl.xml	Sample MBean implementation instrumentation.
PeopleSoft.xml	Old PeopleSoft instrumentation.
SAP61.xml	SAP 6.1 instrumentation.
SmartuneInstrumentation.xml	Optional SmarTune instrumentation for servlet include and session analysis.

About the structure of instrumenter configuration files

The instrumenter configuration files have the following general structure:

```
<?xml version='1.0'?>
<instrumenter-config>
    <custom-config> </custom-config>
    <all-calls-to-method> </all-calls-to-method>
    <all-calls-from-method> </all-calls-from-method>
    <calls-from-method-to-method> </calls-from-method-to-method>
    <ignore-config> </ignore-config>
</instrumenter-config>
```

See About custom instrumentation configuration, About all calls to method instrumentation configuration, About all calls from method instrumentation configuration, About calls from method to method instrumentation configuration, and About ignore instrumentation configuration.

About custom instrumentation configuration

The <custom-config> element has the following structure:

```
<custom-config>
    <java-classes>
```

```
        <java-class> <!-- occurs 0 or more times -->
           <class-name> class-or-interface-name </class-name>
              <methods>
                 <method> <!-- occurs 0 or more times -->
                 <name> method-name </name>
                 <params> <!-- optional -->
                    <param> <!-- occurs 0 or more times -->parameter-type </param>
                 </params>
                 <capture-param-index> <!-- optional -->capture-parameter </capture-param-index>
                 </method>
              </methods>
        </java-class>
     </java-classes>
  </custom-config>
```

About all calls to method instrumentation configuration

The <all-calls-to-method> element has the following structure:

```
<all-calls-to-method>
   <methods>
      <method> <!-- occurs 0 or more times -->
         <name> method-name </name>
         <params> <!-- optional -->
            <param> <!-- occurs 0 or more times -->
            parameter-type </param>
         </params>
      </method>
   </methods>
</all-calls-to-method>
```

About all calls from method instrumentation configuration

The <all-calls-from-method> element has this structure:

```
<all-calls-from-method>
   <java-classes>
      <java-class> <!-- occurs 0 or more times -->
         <class-name> class-or-interface-name </class-name>
         <methods>
            <method> <!-- occurs 0 or more times -->
            <name> method-name </name>
            <params> <!-- optional -->
               <param> <!-- occurs 0 or more times -->
               parameter-type </param>
            </params>
            </method>
         </methods>
      </java-class>
   </java-classes>
</all-calls-from-method>
```

About calls from method to method instrumentation configuration

The <calls-from-method-to-method> element has this structure:

```
<calls-from-method-to-method>
    <invocation-relationship> <!-- occurs 0 or more times -->
        <java-class>
            <class-name> class-or-interface-name </class-name>
                <methods>
                    <method> <!-- occurs 0 or more times -->
                        <name> method-name </name>
                        <params> <!-- optional -->
                            <param> <!-- occurs 0 or more times -->
                            parameter-type </param>
                        </params>
                    </method>
                </methods>
        </java-class>
        <invoked-method> <!-- occurs 0 or more times -->
        invoked-method-name </invoked-method>
    </invocation-relationship>
</calls-from-method-to-method>
```

See About method signature matching.

About ignore instrumentation configuration

Use the <ignore-config> element to configure rules that are used to determine when to prevent instrumentation from being applied to all methods in specifically matched classes or packages, to specifically matched methods, or to specifically matched calls to methods.

The <ignore-config> element has this structure:

```
<ignore-config>
    <java-classes> <!-- optional -->
        <!-- Purpose and structure described below. -->
    </java-classes>
    <invocation-relationship> <!-- occurs 0 or more times -->
        <!-- Purpose and structure described below. -->
    </invocation-relationship>
    <all-calls-to-method> <!-- optional -->
        <!-- Purpose and structure described below. -->
    </all-calls-to-method>
</instrumenter-config>
```

Use the <java-classes> element to prevent instrumentation from being applied to all methods in specifically matched classes or packages or to specifically matched methods. The <java-classes> element has this structure:

```
<java-classes>
    <java-class> <!-- occurs 0 or more times -->
        <class-name> class-or-interface-name </class-name>
        <methods> <!-- optional -->
            <method> <!-- occurs 0 or more times -->
            <name> method-name </name>
            <params> <!-- optional --> </params>
            </method>
```

```
        </methods>
      </java-class>
    </java-classes>
```

See About method signature matching.

For each method to be instrumented, the following rules are used to determine if instrumentation should not be applied to the method:

- If the method is declared in a class whose name matches the specified class or interface name of a <java-class> element, no instrumentation should be applied to the method.
- If the matched <java-class> includes a <methods> element, the method must also match the specified method name of a <method> element so that no instrumentation is applied to the method.
- If the matched <method> element includes a <params> element, the method must also match the specified signature so that no instrumentation is applied to the method.

Wildcards are permitted in the class or interface name and method name. See About using the wildcard character *.

Use the <invocation-relationship> element to prevent instrumentation from being applied to specifically matched calls to methods from a specifically matched calling method. The <invocation-relationship> element has the following structure:

```
<invocation-relationship> <!-- occurs 0 or more times -->
  <java-class>
    <class-name> class-or-interface-name </class-name>
    <methods>
      <method> <!-- occurs 0 or more times -->
      <name> method-name </name>
      <params> <!-- optional -->
      </params>
      </method>
    </methods>
  </java-class>
  <invoked-method> <!-- occurs 0 or more times -->
  qualified-method-name </invoked-method>
</invocation-relationship>
```

See About method signature matching.

For each call to a method to be instrumented, the following rules are used to determine if instrumentation should not be applied to the method:

- If the called method is declared in a class whose name matches the specified class or interface name of a <java-class> element, no instrumentation should be applied to the method.
- If the matched, <java-class> includes a <methods> element, the method must also match the specified method name of a <method> element so that no instrumentation is applied to the method. If the matched <method> element includes a <params> element, the method must also match the specified signature so that no instrumentation is applied to the method.

Use the <all-calls-to-method> element to prevent instrumentation from being applied to specifically matched calls to methods from any calling method. The <all-calls-to-method> element has the following structure:

```
<all-calls-to-method>
  <methods>
    <method> <!-- occurs 0 or more times -->
      <name> method-name </name>
      <params> <!-- optional -->
      </params>
    </method>
```

```
      </methods>
   </invocation-relationship>
```

See About method signature matching.

The <name> element must represent a fully qualified method name. The portion of the <name> element after the last dot (".") is considered to be the method name, and the portion of the <name> element before the last dot is considered to be the class name. When you use wildcards, it is important to use a wildcard pattern that fits the scheme described. For example, the wildcards pattern *.* matches all methods of all classes.

About common instrumenter configuration matching techniques

Common instrumenter configuration matching techniques are method signature matching and using the wildcard character *.

About method signature matching

The <params> element configures rules that are used to match method signatures for processing by the instrumenter. The <params> element has the following structure:

```
<params> <!-- optional -->
   <param> <!-- occurs 0 or more times --> parameter-type </param>
</params>
```

The parameter type is the same as the abstract type declarator for the parameter type in Java. For example, the parameter-type for a parameter of type java.lang.String is java.lang.String, and the parameter type for a parameter of type int[][] is int[][].

The following primitive parameter types names are recognized:

- `boolean`
- `byte`
- `char`
- `double`
- `float`
- `int`
- `long`
- `short`
- `void`

The <invoked-method> elements in the <calls-from-method-to-method> element are used to match specific method signatures for specific calls from one method to another. All methods that match the signature are instrumented. Wildcard expressions are not supported and return types are not matched. The <invoked-method> format must be expressed using the same primitive types that were discussed for the <param> element.

For example:

<invoked-method> java.lang.Thread.sleep(long) </invoked-method>

or

<invoked-method> com.acme.shared.comm.Connector.<init> (java.lang.String,int[][]) </invoked-method>

About using the wildcard character *

In addition to using specific names to reference items such as classes, interfaces, and methods, for instrumentation, you can also use the wildcard character * to instruct Precise for J2EE to instrument all classes, methods, or packages within the scope of the instruction.

The following table illustrates how the wildcard character can be used.

Table 18-3 *Usage of wildcard character ***

Wildcard	Instrumented elements	Non-instrumented elements
*	xmp.server.Main xmp.task.AbstractTask xmp.task.AbstractTask$1 xmp.task.util.TaskUtilities	
xmp.task.*	xmp.task.AbstractTask xmp.task.AbstractTask$1 xmp.task.util.TaskUtilities	xmp.server.Main
$	xmp.task.AbstractTask$1	xmp.server.Main xmp.task.AbstractTask xmp.task.util.TaskUtilities

> ⓘ Use wildcard characters only when discovering the methods to instrument. Otherwise, it may result in instrumentation that does not yield meaningful performance metrics but introduces unwanted overhead. Do not implement wildcarded instrumentation in production environments.

Including application server classes in Leak Seeker instrumentation

To obtain information on collections and arrays with the most elements, Leak Seeker instruments all user application classes but, by default, excludes application server classes. In special circumstances, you may want Leak Seeker to collect information on application server classes as well.

> ❗ Instrumenting application server classes may increase excessively the startup time as well as the running time of the application.

To include Leak Seeker instrumentation for application server classes

1. In the *JVMID*/LeakSeeker.xml file, find the application server class prefix for the application server classes that you want to instrument. For example, to identify WebLogic application server classes to instrument, you would search for lines beginning with com.bea and weblogic.
2. Comment out the lines for the application server whose classes you want to instrument. For example, the following lines cause Leak Seeker to collect information for WebLogic application server classes.

        <!--leakseeker-if-package>com.bea</leakseeker-if-package>
        <!--leakseeker-if-package>weblogic</leakseeker-if-package>