Precise 10.3

Precise Insight SDK User Guide

Exported on 03/21/2021

Table of Contents

This help provides the following topics for Precise Insight SDK:

Introducing the Precise add-on AppTier

This section includes the following topics:

About the Precise Add-on AppTier

The Add-on AppTier extends agent-based data collection to applications currently not supported by Precise products. It is an SDK (Software Development Kit) used by a customer's application development team, a third-party vendor, or other professional services team to create an application Collector for new AppTiers in the Precise environment.

AppTier is the Precise abbreviation for an application tier in a Precise environment. Each AppTier contains the instances of one technology (third-party software tool) for functional and business-related views. Precise agents monitor, process, and communicate performance-related information for that tier. An installed Add-on Collector monitors new App-Tier instances and collects application-specific performance data.

The customer's application development team, third-party vendor, or other professional services team plan and develop how the Add-on AppTier collects the application-specific information.

You must perform the following actions to create an Add-on AppTier:

- Develop an Add-on AppTier installation package with files that describe the new application to the Precise environment.
- Install the Add-on AppTier plug-in installation package to a Precise environment and test it.
- Develop the Add-on AppTier Collector. The software designer must either develop the Collector for the application, or use the ARM API standard inside the application.
- Distribute the Add-on AppTier installation package and Collector to Precise users.

Two options are available for planning the Add-on AppTier:

- Design the Add-on AppTier Collector (or external Collector) that generates XML output files. Insight will process and load the data to the Precise product suite.
- Use the ARM (Application Response Measurement) standard API (Application Programming Interface), supplied by Precise Corporation. ARM is an open-source specification and API standard which defines an API library that can be used for integration to a Precise installation. The Insight SDK Collector can quickly extract the minimum information when the system is idle.

Components of the Add-on AppTier

To understand the process of introducing a new AppTier to a Precise environment, you should know how components are used to create an Add-on AppTier.

The following are major Add-on AppTier components:

- If using XML format, a user-defined Add-on AppTier Collector that collects data after installation to the monitored server.
- If using an ARM standard API, an Insight SDK Collector that collects data from the Insight ARM extension dynamic library in the monitored server.
- Insight FocalPoint in the Precise FocalPoint server
- Add-on AppTier plug-in installation package that introduces a new AppTier to the Precise FocalPoint server

About the Add-on AppTier Collector

The Add-on AppTier Collector is user-defined to collect application-specific information in XML format files. As an external Collector, the customer's home-grown application development team, third-party vendor, or other professional services team develops the Collector.

The Add-on AppTier Collector is separate from an Insight Collector. Currently, it can be developed in any programming language or script. When the XML file format is used, the Add-on AppTier Collector passes the data to the Precise Listener by writing files to a folder under the Precise installation directory. The Precise Listener also includes a Harvester. The Harvester is responsible for collecting application-specific data files and transferring the performance files from the monitored server to the Precise FocalPoint server.

About the Insight SDK Collector

The Insight SDK Collector collects application specific information using an ARM standard API. The Insight SDK Collector receives ARM API events from the monitored application in order to extract performance information. An SDK software designer needs to develop the ARM call back functions used inside the monitored application.

For more details regarding the ARM standard, refer to: http://www.opengroup.org/management/arm/. When using ARM SDK format, the Insight SDK Collector passes the data to the Precise Listener by writing files to a folder under the Precise installation directory. The Precise Listener also includes a Harvester. The Harvester is responsible for collecting application-specific data files and transferring the performance files from the monitored server to the Precise FocalPoint server.

About Insight FocalPoint in the Precise FocalPoint server

Insight FocalPoint (usually installed on the Precise FocalPoint server) aggregates all of the performance information that all Insight agents collect. See How the Collectors work for more information.

About the Add-on AppTier plug-in installation package

The Add-on AppTier plug-in installation package includes one or more XML files which contain metadata that describes the new Add-on AppTier to the Precise environment. The software designer defines the Add-on AppTier application entities, counters, and general application-level definitions.

The Add-on AppTier plug-in, a component of the Precise FocalPoint server, reads the plug-in installation package and builds or updates the necessary Precise metadata files. The Add-on AppTier plug-in installation package also invokes the processes needed to implement changes. For example, the installation package invokes the PMDB process to perform schema changes in the database. If needed, consult with the Precise research and development team regarding Add-on AppTier metadata development.

About Add-on AppTier metadata

Each performance record, generated by the user-defined Collector, can include the following information:

- **Entities**. They are identifiers invoked and/or processed by the monitored application, such as: Activity name, Client IP address, user or login name, program name, and others.
- **Server time measurements**. These are time-based counters that indicate the time it took for the monitored application to process the activity till completion.
  Service time counters can represent various functions within the monitored application. These counters were involved in processing the activity, such as: authentication time, email delivery time, database time, and others.
- **Statistical counters**. They count additional properties for the activities, such as: activity message size and I/O Bytes/sec.

- **Executions counts**. The count the number of activities that occurred in one time slice (every 15 minutes).

How the Collectors work

AdminPoint performs the plug-in installation and defines the instances. After plug-in installation, certain screens display the new Add-on AppTier instance in AdminPoint. Both the Add-on AppTier Collector (external) and Insight SDK Collectors can start providing performance data for monitoring.

When the Add-on AppTier Collector is used, it passes the data as XML formatted files to the Precise Listener by writing files to a folder under the Precise installation directory.

The Insight ARM extension is a dynamic library within the application which implements the ARM API and transfers transaction information to the Insight SDK Collector. The Insight SDK Collector correlates the transactions and creates XML performance files.

The Precise Listener (in the monitored server) includes a Harvester that collects application-specific data files from both the Add-on AppTier Collector (external Collector) and the Insight SDK Collector file directory and transfers the performance files from the monitored server to the Insight FocalPoint. Files are loaded from the monitored (Collector) servers to the Insight FocalPoint.

The Insight FocalPoint aggregates all of the performance information that all Precise Collectors collect and stores the collected information in a centralized location. For received application-specific data files, the Insight FocalPoint runs a pre-load process to read the files and converts the data to Precise standards, such as: common time adjustments and location calculations. The pre-load process creates the PMDB load files for the performance records.

The PMDB, on the Precise FocalPoint server, stores performance records as history received from Insight FocalPoint. The data is correlated along AppTiers (allowing Cross AppTier analysis) in the PMDB. Report Manager retrieves the information from the PMDB and generates reports.

The following items display the performance data:

- StartPoint is the dashboard startup screen.
- Here you select the environment, AppTier, and technology to be viewed
- Insight is a GUI screen which associates the problem(s) with their sources
- AdminPoint is the administration tool.
- It lets you add, delete, or change AppTiers and their technologies in a Precise environment
- Report Manager is the reporting tool. It compares the actual performance metrics to an appropriate baseline and identifies significant deviations:
  - **Report Manager** displays the following types of reports:
    - **Top-*n* reports** identify major business activities and resource hogs
    - **Trend reports** establish and analyze long-term system behavior
    - **Exception reports** identify deviations from an established baseline, and from long-term system behavior
    - **Customized reports**

About Insight SDK Collector processing

The Add-on AppTier application data is processed before being harvested by the Precise Listener. The software designer defines the order for processing the application data.

The application data is processed as follows:

- Get the transaction status events from the Insight ARM extension dynamic library, such as transaction start and transaction end.
- Correlate the transaction events using the transaction correlation mechanism.
- Format the transaction data according to the Add-on AppTier XML format.

- Write the transaction to the output file.
- Close the output file and create the signature file at the end of each time slice (every 15 minutes).

About Insight SDK Collector information flow

The Insight SDK Collector receives information from the Insight ARM extension dynamic library by using a dedicated process for each monitored instance, called psi_sdk_poll. A single process (one per monitored instance) called psi_sdk writes SDK files under the Precise Listener directory.

Where to get more information

More information on Precise, its products, technical notes, and so on, can be found in the Release Notes document for this version.

Developing an add-on AppTier

This section includes the following topics:

About developing an Add-on AppTier

To describe the Add-on AppTier development process, we will use an imaginary example of an Add-on AppTier application named Generic Mail Server (GMS). GMS provides email services to organizations and enterprises. It was developed by an independent software vendor here-named GMS-ISV.

GMS users are recognized by the application through their login user name. GMS users can log into the application using various devices and from various locations, such as: from their workstation, home desktop computer, laptop computer, or even their cell phone. The system monitors the user's current connection by logging the IP network address for the client device. GMS users can send and receive mail messages which can include attachments such as: documents, pictures, and various other types of files.

There are three options available for developing an Add-on AppTier:

- **Third-party vendor**. Develops and/or publishes a software application which enables its customers to monitor and tune performance for a locally installed application through the Precise environment.
  For example, Independent Software Vendor (GMS-ISV) which owns and publishes the Generic Mail Server application, may decide to develop a GMS Add-on AppTier.
  GMS application owners using Precise products can plug in the GMS Add-on AppTier to their local Precise installation to enable performance monitoring of their GMS application.
  The GMS-ISV Research and Development team must develop the GMS Add-on AppTier and make the necessary changes to their GMS application so that performance data can be received and displayed through Precise products, such as: StartPoint, AdminPoint, Insight, and Report Manager.
- **Precise customer**. Develops an Add-on AppTier to monitor local installations or home-grown applications.
  For example, GMS-ISV may decide not to develop an Add-on AppTier for GMS. GMS users that also use Precise products can still develop a customized GMS Add-on AppTier by developing tools and utilities that externally monitor the activity of their local GMS application, which will provide the performance measurements to be displayed in the Precise environment.
  For this situation, the Precise customer's own software development team needs to research how to externally monitor the GMS application. This team will also develop the GMS Add-on AppTier, including the necessary tools and utilities that externally monitor local GMS application activities and provide the performance measurements to be displayed in the Precise environment.
- **Precise professional services**. Precise professional services can tailor solutions for Precise customers by developing Add-on AppTiers for various external applications Precise customers use.

About the development process

Developing an Add-on AppTier is a complex process that requires completing the following software development phases: design, code development, testing, packaging and distribution.

The following is a list of phases you must perform for successful development and delivery of an Add-on AppTier.

- **Design the Add-on AppTier**. The Design phase studies both functional and technical aspects for all components of the Add-on AppTier. Prepare a specifications document with all of the details.
  The Add-on AppTier developer must determine the following regarding the data:
  - What information will be collected from the monitored application
  - How will the information be extracted or sampled
  - How will the information be stored and displayed in the Precise product suite
- **Develop and test the Add-on AppTier installation package**. The Add-on AppTier installation package consists of a set of XML files that contain metadata. The XML files define the Add-on AppTier to the Precise product suite and let Precise products correctly parse and display data they receive from an Add-on AppTier Collector.
  Testing the installation package lets the Add-on AppTier developer examine how Add-on AppTier information will be displayed to the end user through Insight and Report Manager products. After testing, the developer can redesign the Add-on AppTier to tweak and optimize displays.
- **Develop and test the SDK Colector**. The Add-on AppTier installation package and the detailed specifications document (for expected deliverables from the Add-on AppTier Collector) should be ready almost in parallel. Next the Add-on AppTier developer must write code which complies with Precise standards for data format.
  The Add-on AppTier developer has two options for developing code:
  - Write code for an Add-on AppTier Collector, or
  - Write code for an ARM API (through an Insight SDK dynamic extension library)
- **Test the completed Add-on AppTier solution**. The Add-on AppTier or Insight SDK Collector should now be functional and the Add-on AppTier installation package complete. The Add-on AppTier installation package specifies what data is expected from the Add-on AppTier or Insight SDK Collector.
  Perform integration testing of the complete Add-on AppTier solution to verify correct data flow and communication.
  Consider other aspects, such as: usability, performance, and scalability.
- **Package, deploy, and distribute the Add-on AppTier**. After testing is complete for the entire Add-on AppTier solution, plan how the installation package and Add-on AppTier or Insight SDK Collector should be packaged and distributed to end users.
  Consider other aspects, such as: documentation, versioning, licensing, distribution, and support.

About designing an Add-on AppTier

There are two stages in Add-on AppTier design: functional and technical design.

During functional design, the Add-on AppTier developer needs to answer the following questions:

- Which information should be displayed in the Precise environment?
- How will the information be presented to the end user in Insight and Report Manager?

During technical design, the Add-on AppTier developer needs to answer the following questions:

- How will the data be sampled from the monitored application?
- How will the data be stored in the PMDB?

About prerequisites for Add-on AppTier design

To design an Add-on AppTier well, the Add-on AppTier developer must be knowledgeable in the following areas:

- **Monitored application**. Know how the monitored application operates and what is the best method for measuring performance for the application.
  Be able to answer the following questions regarding the monitored application:
  - What are common performance problems?
  - What are common complaints from users?

  - What types of performance measurements can be collected?
  - Does the monitored application support the ARM standard or does it support other standards and APIs?
  - Are there any external tools that can monitor the monitored application's performance?
- **Precise processing and display**. Understand how Insight and Report Manager display performance data from various technologies.
  Be able to answer the following questions:
  - What are the strengths in Insight and Report Manager and how best to utilize them?
  - What are the weaknesses in Insight and Report Manager?

About typical Precise AppTier components

A typical Precise AppTier contains several types of entities and counters.

- **Entities**. Entities identify the data giving it transactional context.
  There are three kinds of entities:
  - In client/server applications, entities can identify the client- or server-side machine, such as: login user name, IP address for client or server machine, and others.
  - Entities also identify information that is passed between the client and server. The passed information can be a message name, email title, transaction name, and others.
  - In self-contained applications (not client-server applications), these entities describe a basic (atomic) recurring operation or function that can be used to categorize application performance, such as: method or function name.
  - These entities are activity definition entities. Multiple entities can together define a single activity. However, there is always one entity that is classified as "the activity."
  - Entities can also identify global settings that are not tied to a specific transaction and effect the monitored application's operation. These entities can help identify problematic settings. For example, the same transaction can operate differently when activated online or in batch mode.
- **Counters**. Counters represent the performance measurement of activities.
  There are four kinds of counters:
  - **Service Time counters**. These counters measure the monitored application's working time to complete an activity. In its most basic form, there is one "work time" counter that measures the entire application's performance from the moment the activity begins until its completion.
    Service Time can sometimes be broken down to measure several separate functions that operate on the activity. For example, if the application is multi-tier, there can be both application and database times. Or perhaps the operation is passed to an external resource, such as: authentication or security checks. Breaking down the Service Time helps you discover which players are concerned with servicing a specific activity, and helps identify which player may be responsible for an existing bottleneck.

    > Service Time defines the working time of the monitored application itself. It does not include network times and fat client times.

  - **Network time**. Network time measures the time it takes for the activity to pass through the network between the client and server. Network time can be calculated by the Add-on AppTier Collector, or by integrating the Add-on AppTier with the Insight Network Collector.
  - **Statistical counters**. These counters are measurements that do not describe the working time of an activity. For example a message can be divided into several packets, or an email can be of a specific length and can contain one or more attachments.
    Statistical counters are often used to characterize poor performing activities. For example, the Precise user may find that email messages with large attachments are causing high network times.
  - **Executions**. An Executions counter specifies the number of invocations for the same activity that occurred in one time slice (every 15 minutes).
    There are different Executions counters for every mutation of entity values. For example: if the same email

message is sent by two different users, there will be two records in the XML file from the Add-on AppTier Collector, each file having a different Executions counter.

About entity classifications

Entities and their attributes identify information that is passed between the client and server. There are two aims for classifying entities:

- **Identify entities for pre-load processing**. Insight performs special processing on specific classified entities, before the data is loaded into the PMDB. The pre-load process includes the following calculations:
  - **Calculate Service Level Agreement (SLA) breaches for activities**. For every activity invoked, Insight calculates the average service time in the time slice as total reported service time divided by the number of executions. Insight compares the average service time to SLA definitions and assigns a color code (green, yellow, or red) to all activity executions in the time slice. For this aim, Insight needs to identify the entity that is classified as the activity.
  - **Calculate the location of the Client IP address**. If an entity with a Client IP classification exists, Insight will look for and assign a location definition to that entity.
- **Cross-AppTier comparison**. Classifying entities enables Insight to display entities from different AppTiers side-by-side for comparison. For example, if an entity is classified as a Client IP, Insight will display this entity side-by-side with the Client IP entity of an Precise for Web product.
  Insight displays a side-by-side comparison of different AppTiers in the Cross-AppTiers workspace.

The following is a complete list of entity classifications as they appear in the Insight Display list:

- Activities
- Super Activities
- Client IPs
- Servers
- Instances
- Users
- Locations

About defining an Activity or Super Activity

An Activity is the atomic (most basic) unit (entity) which defines an action. The following are some examples of an activity:

- SQL statements in a database tier
- URL address in a Web tier
- Method in a Microsoft .NET or J2EE application tier
- Transaction for SAP in an application tier

A Super Activity is an entity which invokes other Activities. For example, a program that activates SQL statements in a database tier is a Super Activity.

About counter classifications

Counters are classified into Service Time, Statistical counters, and Executions, all used for similar purposes:

- **Identify counters for pre-load processing**. To calculate an SLA branch, Insight must identify Service Time counters as well as the Executions counter.
- **Cross-AppTier and Over Time comparison**. Insight and Report Manager use Service Time counters to compare the performance of different AppTiers side-by-side.

About Grouping

Grouping is a feature that enables Precise users to aggregate multiple values of performance measurement for one entity into a single line in a graph, according to a textual pattern. You use this feature to group entity values according to business functions.

After you have defined a grouping for an entity, a new graph and table appear in Insight. For example, Insight displays performance calculations for the Senders grouping. Each displayed line indicates the sum of all performance calculations for Senders that belong to this group.

> ⓘ Report Manager lets you define custom reports for grouped entities. Groups for Client IP entities are called Locations and are defined using a separate settings dialog in AdminPoint.

About Insight and Report Manager standards and restrictions

Insight and Report Manager impose certain standards and restrictions to AppTiers:

- **Flat hierarchy of entities**. Insight and Report Manager do not recognize a hierarchy of entities. There is no tree view and all entities are displayed in identical graphs and tables.
- **All counters are applicable to all entities**. Insight and Report Manager compare all entities according to the same counters; this design is best suited for comparing entities according to performance measurements. This design is less suited for displaying counters that are not applicable to specific entities, such as: instance statistics.
- **Maximum of 10 entities and 10 counters**. Based on the two previous standards for entities, Insight and Report Manager always display all entities compared to all counters. If there are many entities and/or many counters, the screen display or report can become crowded and cumbersome. As a general guideline to follow, create no more than 10 entities and 10 counters.

About functional design of an Add-on AppTier

The functional design of an Add-on AppTier involves setting how the Add-on AppTier will be displayed in Insight and Report Manager; whether to set it as a standalone AppTier, or in multi-AppTier environments.

Consider the following issues when performing a functional design:

- **Familiarize yourself with Insight and Report Manager**. View standard Precise technologies through Insight screens and Report Manager reports.
  See how the products display performance measurements in the various graphs and tables and get a general idea of how you want the monitored application to display.
- **Decide on the entities and counters for the monitored application**. If the monitored application is typically part of multi-tier environments, you will want to classify your entities and counters so that Precise users can benefit from the Insight Cross-AppTier workspace display and the Report Manager Environment Profile report.

About guidelines for the functional design of an Add-on AppTier

The following are guidelines you should follow for entities:

- **Mandatory entities: Instance, Server and Activity**. Instance and Server entities are mandatory because they are used in AdminPoint when you define a new instance for the Add-on AppTier.
  An entity with an Activity classification must also be defined. The Activity entity is the basis for SLA calculations and other processing.
- **Functional attributes**. There are four functional attributes for an entity to consider:

- **Entity name**. The Entity name is used as the title for Insight and Report Manager graphs. You need to specify the entity name in singular and plural variations.
- **Entity classification**. The Entity classification is useful for multi-tier environments to compare this entity against similar entities in different AppTiers.
- **Is it mandatory?** In other words, does the entity contain a value for every activity, or maybe the entity is not applicable in certain cases?
  Instance, Server, and Activity are mandatory entities; every record in the Add-on AppTier Collector file will have a value for these entities.
- **Do you want to allow grouping for the entity?** Grouping is a sensible solution if the entity's collected values can be categorized by Precise users according to a business or technical function. It allows Precise users to group mail Senders according to departmental categories.
  However, the Generic Mail Server Add-on AppTier also has an entity called Message Type that defines regular emails, meeting schedules, and others. The Message Type entity does not support the grouping feature.

The following are guidelines you should follow for counters:

- **Mandatory counters: minimum of one Service Time counter**. There must be at least one counter defined with the Service Time classification.
  The Service Time counter is the basis for all SLA calculations, Cross-AppTier comparisons, and other processing.
- **Optional: Executions**. The Executions counter is not mandatory.
  If not specified, Insight treats every record in the Add-on AppTier Collector file as one execution.
- **Functional attributes**. There are four functional attributes for a counter to consider:
  - **Counter name**. The Counter name is used as an Insight Tool Tip in the Compared by drop-down list and in the report Manager Y-axis legend.
  - **Counter classification**. The Counter classification defines Service Time counters, the Executions counter, and Statistical counters.
  - **Counter color**. Both Insight and Report Manager support a pre-defined color palette.
  - **Unit of measure**. This attribute affects the display format for counter values in Insight and Report Manager. Service Time counters are always time based while the Executions counter is always a number. However, Statistical counters can represent either a number of Bytes. If Bytes, they are summed as KiloBytes, MegaBytes, and so on.

The following are general guidelines:

- **Add-on AppTier name**. The Add-on AppTier name is used during plug-in installation of the Add-on AppTier and in StartPoint, AdminPoint, Insight, and Report Manager screens.
- **SLA applicability**. You can give Precise users the ability to define SLA levels for activities.
  If provided, Precise used can define SLA levels in AdminPoint for the average Service Time for activities. Insight will calculate the SLA breaches for performance data that is received from the Add-on or Insight SDK Collector.

About technical design of an Add-on AppTier

The technical design of an Add-on AppTier involves setting how data will flow from the Add-on AppTier or Insight SDK Collector to Insight, and how the data will be stored in the PMDB.

Consider the following issues when performing a technical design:

- **Use ARM of file-based collection**. You must decide what mode of collection to use. See About guidelines for the technical design of an Add-on AppTier.
- **Design the Collector file**. Whether you use ARM (Application Response Measurement) or file-based collection, the Collector XML file becomes the vehicle by which the Add-on AppTier or Insight SDK Collector provides data for Precise installation.
  Insight must know how to correctly read and parse the Collector file before it is loaded into the PMDB.

About guidelines for the technical design of an Add-on AppTier

The following are guidelines you should follow for the technical design:

- **ARM collection mode**. ARM (Application Response Measurement) is an industry standard that is growing in popularity.
  Implementing the ARM standard in the monitored application will open up the monitored application to all monitoring and tuning tools that support this standard. It also allows the tools to probe the monitored application and display performance calculations and perform application analysis.
  Using the ARM standard API solution involves code instrumentation. This is possible only if the Add-on AppTier developer has access to the source code for the monitored application. For example, the Generic Mail Server vendor can implement the ARM standard API by incrementing the application's code.
  Using the ARM standard API, the Insight SDK Collector creates a Collector XML file (a complex process) and aggregates the information after every time slice (every 15 minutes).
  Currently you can implement the ARM standard API for an Add-on AppTier using C language only.
  For more details regarding the ARM standard, refer to http://www.opengroup.org/management/arm/.
- **File collection mode**. In the file collection mode, the task of developing the Add-on AppTier Collector rests on the shoulders of the Add-on AppTier developer.
  The Add-on AppTier developer can monitor the application and collect performance data any way that suits his needs. He is not limited by restrictions, such as: API standards, language support, platform support, and so on.
  The Add-on AppTier Collector must create Collector XML files using Precise format and file naming conventions.
  See About ETD file application-level entities for more information.
  The Add-on AppTier Collector aggregates performance measurement into 15-minute time slices that are then written into separate Collector files.
- **Entity and counter technical attributes**. There are two technical attributes to consider:
  - **Database column name**. This is the name of the database column in the PMDB that stores entity or counter values.
  - **XML tag name in the Collector file**. This is the name of the XML tag in the Add-on AppTier or Insight SDK Collector file that holds values for the entity or counter.

About developing and testing the Add-on AppTier installation package

The Add-on AppTier plug-in installation package is a series of metadata XML files that contain all functional and technical attributes for Add-on AppTier entities and counters.

Testing the Add-on AppTier installation package gives the Add-on AppTier developer his first glimpse into how the Add-on AppTier will display in Insight and Report Manager. The Add-on AppTier installation package also contains the technical specifications to the Add-on AppTier or Insight SDK Collector file.

About prerequisites for developing an Add-on AppTier installation package

You need the following prerequisites to develop and test the Add-on AppTier installation package:

- Functional and technical specifications for the Add-on AppTier
- Add-on AppTier installation package specifications, as described in Add-on AppTier installation package descriptor file.
- Insight and Report Manager FocalPoints installed

About testing the Add-on AppTier installation package

Perform the following to test the Add-on AppTier installation package:

- **Plug-in the Add-on AppTier package**. Plug-in the Add-on AppTier installation package using AdminPoint. This will test whether the installation package is correct and complete. See About installing the Add-on AppTier plug-in installation package for more information.
- **Manually create a Collector file and load it to the PMDB**. After plugging-in the Add-on AppTier installation package has finished, you need to wait until data is loaded into the PMDB before any information displays through Insight or Report Manager.
  Consequently for testing, you must supply an Add-on AppTier Collector XML file with a sample file. In most cases, a Collector XML file will not exist until you develop the Add-on AppTier Collector. Manually create the Collector XML file according to the XML tag names specified in the Add-on AppTier installation package, and according to file name conventions. See About the XML Collector file for file name conventions.
  After your Collector XML file is successfully loaded into the PMDB, and Insight and Report Manager display the correct information, the sample Collector file can be used as a reference for the following phases in developing the Add-on AppTier or Insight SDK Collector.

About developing and testing the Insight SDK Collector

After testing the Add-on AppTier installation package and an sample Collector file is available as reference, you can write the code for an Insight SDK Collector.

The Insight SDK Collector collects application-specific information using an ARM (Application Response Measurement) standard API (Application Programming Interface) in the application. It correlates the transactions it receives from the Insight ARM extension dynamic library, within the application, and creates performance files.

Initializing the Insight ARM extension dynamic library

Initializing the Insight ARM extension dynamic library begins when the Add-on AppTier application calls to the arm_init function.

Before installing the Insight ARM extension dynamic library, copy the original ARM dynamic library to another directory.

To initialize the Insight ARM extension dynamic library

1. Create or attach to the shared memory segment holding the configuration.
2. Connect to the IPC channel. The IPC channel passes transactions information from the Insight ARM extension dynamic library to the Insight SDK Collector. The IPC channel (UDP port) number is stored in the shared memory.
3. Initialize the application and user-defined properties within the parameters for the arm_init function.
4. Initialize the transaction hash. This hash includes all user-defined transactions defined while the Add-on AppTier application is running.

About testing the complete Add-on AppTier solution

Testing the complete Add-on AppTier solution involves completing the following cycle:

- Plugging-in the Add-on AppTier using the Add-on AppTier installation package
- Defining instances and invoking the Insight SDK Collector

Place Collector files in the correct directory, where the Precise harvester automatically puts the files into the appropriate directory on the Precise FocalPoint Server.

Other aspects, such as: performance and scalability for the Add-on AppTier can now be addressed.

About packaging, deployment, and distribution of an Add-on AppTier

These are the final steps in developing the Add-on AppTier solution.

Other aspects, such as: documentation, versioning, licensing, distribution and support should now be addressed.

About Add-on AppTier registration

Before distributing an Add-on AppTier installation package to Precise users, send the newly developed and tested Add-on AppTier installation package to the Precise research and development team for review, validation, and certification.

Your new Add-on AppTier will be allocated a unique tech-code and a user interface titles range to be specified in the ETD file.

> For development and testing, you can use a list of pre-allocated tech-code and a user interface titles range specified in, see About the titles file or files.

The following information must be sent to the research and development team:

- A high-level document describing the monitored application
- A high-level description describing the purpose and functionality of the Add-on AppTier solution
- If using the ARM API option, describe the instrumented transactions and their properties (as passed to the ARM API)
- The Add-on AppTier installation package
- Optional: target customer and use cases
- Contact details for the Add-on AppTier developer

Email the required information to: dl-rnddevelopment-tpmteamleaders@precise.com

Installing the add-on AppTier collector

This section includes the following topics:

About installing and configuring the Add-on AppTier Collector

For each AppTier, specific Collectors collect performance data from instances in your environment. These agents need to be installed for each instance you want to monitor, and activated on the server computer where the instance resides. The Collectors send this data to the relevant product FocalPoint, which processes the information and stores it in the PMDB. The PMDB keeps the data for long-term trend and capacity analyses.

Precise components must be installed in the following order:

- Precise FocalPoint
- PMDB FocalPoint
- Insight FocalPoint
- Report Manager FocalPoint

> Insight, Report Manager FocalPoints, and PMDB FocalPoints must be running during the Add-on AppTier plug-in process.

About installing the Add-on AppTier plug-in installation package

Once you have installed the Precise framework, you can install the Add-on AppTier Collector for the AppTiers that are part of your environment. For each AppTier, specific Collectors collect performance data from the instances in your environment. These agents need to be installed for each instance you want to monitor, and activated on the server where the instance resides. The Collectors send this data to the relevant product FocalPoint, which processes the information and stores it in the PMDB. The PMDB keeps the data for long-term trend and capacity analyses.

The Add-on AppTier installation package is plugged into the Precise environment using the **Setup > Instances and Clusters** option in AdminPoint. The Add-on AppTier dialog box opens with the current Precise installation directory where Precise FocalPoint is installed.

> The AppTier Name is created from the application tag, but can be changed. Be careful that the AppTier Name includes an underscore "_" prefix to the name (for example: _MyApp). To identify an AppTier Name to a specific environment, add an underscore "_" and number suffix to the name (for example: _MyApp_2).

For more information, see Add-on AppTier installation package contents. For more information, see the Precise Administration Guide.

Adding a new Add-on AppTier

All Add-on AppTiers are added to the Precise environment using AdminPoint.

To add a new Add-on AppTier

1. Launch AdminPoint and go to **Setup > Instances & Clusters**.
2. Click **Add Instance**.
3. From the drop-down menu, select **Add a new Technology**.

4. In the new dialog box, choose the Package file you created. The SDK Collector version is automatically filled in.
5. Click **OK**.

About error messages when installing an Add-on AppTier

Certain error messages can appear during a plug-in installation.

Table 1 Typical error messages when installing an Add-on AppTier

Message	Description
ZIP_SECURITY_CHECK_FAIL	The security check for the installation package files has failed. Contact PreciseTechnical Support.
ZIP_FILES_NOT_VALID	The zip files are not valid. For example: XML validation has failed or the application tag does not exist.
APPLICATION_NAME_ALREA DY_EXISTS	The unique AppTier Name (Application Name) already exists. Rename this AppTier Name.

About installing an instance for an Add-on AppTier

After the Add-on AppTier installation has finished, AdminPoint lets you install (add), update, or uninstall the Precise environment instance for the Add-on AppTier.

The installation process for an Add-on AppTier instance, which is monitored by the Insight SDK Collector (an ARM API option), is almost the same as installing the Add-on AppTier Collector (a files API option). The instructions on the instance install screen are similar. Follow the instructions on the screen to copy the shared library.

To install an Add-on AppTier, you must define them by AppTier instance name, monitored server name, and collection server name. Another name for the monitored server is "Instance" server because this server has installed a specific technology (instance) on it. The Instance server can be selected from a list of existing Precise servers. The Add-on AppTier collection server is where the files are stored for harvesting. The Precise Listener should be installed on the collection server.

On the AppTiers page, you can do any of the following actions:

- Add a new instance
- Edit an existing instance
- Uninstall (Delete) an instance

Installing an instance for an Add-on AppTier

Before installing an instance for an Add-on AppTier, define it by AppTier instance name, monitored server name, and collection server name.

To install an instance for an Add-on AppTier

1. Launch AdminPoint and select **Setup > Instances & Clusters**.
2. Click **Add Instance**.
3. From the dropdown menu, select the Add-on technology.

4. Click **OK**.
5. In the SDK Instance Installation - Property Settings dialog box, insert the Instance Server name, the Instance Name, and the Collect Activity from This Server name.

   > The name used in Collect Activity from This Server must be the same as the Instance Server name.

6. Click **Next**.
7. Click **Finished**.

Installing the Insight ARM extension dynamic library

When using the ARM API option, the Insight SDK Collector is installed on the monitored server. At the end of an instance installation, manual action items are displayed and must be performed. Follow the instructions on the screen to complete instance installation.

> The following is an example procedure of action items; make sure to perform the actual action items that are displayed on the screen.

To install the Insight ARM extension dynamic library (example)

1. Stop the Add-on AppTier application.
2. Backup the original ARM dynamic library to another directory specified by the user.
3. Replace the original ARM dynamic library with the Insight ARM extension dynamic library.
4. Insert Insight instance environment variables into the Add-on AppTier application environment variables.
5. Restart the Add-on AppTier application.

After completing installation, use the Precise AdminPoint PMDB, Warehouse Status, and Agents screens to manage Add-on AppTier instances. You can also use Environment Settings. For more information see the Precise Administration Guide.

About the Agents workspace for an Insight SDK AppTier instance

The Insight SDK Collector is installed for each Add-on AppTier instance which uses the ARM API option. Each agent is displayed in a row in the AdminPoint's Agents workspace.

> The Agents workspace does not display any rows Add-on AppTier instances which use the files API option. These Collectors are not managed by Precise.

Removing an Add-on AppTier instance installation package

Use the **Setup > Instances and Clusters** option in AdminPoint to uninstall Add-on AppTier instances and remove the Add-on AppTier installation package completely. Use the following procedures to uninstall Add-on AppTier instances and remove the Add-on AppTier completely.

To remove an Add-on AppTier installation package

1. Launch AdminPoint and select **Setup > Instances & Clusters**.
2. Select the instance you wish to delete.
3. Click **Delete**.

ⓘ If the deleted instance was the last instance in an AppTier, you will have the option of deleting the AppTier. All AppTier instances must be deleted before the AppTier can be deleted.

Deleting an Add-on AppTier instance which works with an ARM API option

When deleting an Add-on AppTier instance, which is monitored by the Insight SDK Collector for the ARM API, Precise removes the Insight Collector for this Insight. At the end of the process, you are instructed to undo the manual action items you had performed during instance installation.

Carefully follow the instructions displayed in AdminPoint.

To delete an Add-on AppTier instance which works with an ARM API option

1. In AdminPoint, launch the Agents workspace and click **Stop** to stop running the Add-on AppTier agent. Select the Insight SDK Agent row which monitors the Add-on AppTier instance on the relevant server.
2. Replace the Insight SDK extension dynamic library with the original ARM dynamic library (previously stored as backup in another directory during installation).
3. Remove the Insight environment variables from the Add-on AppTier application environment variables.
4. In AdminPoint, launch the Agents workspace and click **Start** to restart the agent.

Updating an Existing Add-on AppTier

If the Add-on AppTier definition has changed, you must update the definition for the Add-on AppTier.

To update an existing Add-on AppTier

1. Uninstall Add-on AppTier instances
2. Remove the Add-on AppTier before performing a new plug-in (updated version) of the Add-on AppTier installation package.

About viewing data from an add-on AppTier

This section includes the following topics:

- About viewing an Add-on AppTier in Precise StartPoint
- About viewing an Add-on AppTier in Insight
- About viewing an Add-on AppTier in Report Manager
- About viewing an Add-on AppTier in AdminPoint

About viewing an Add-on AppTier in Precise StartPoint

Within Precise, you can view the data flow diagram of AppTiers on StartPoint and in the Overview tab for Insight. The Add-on AppTier first appears in StartPoint as an unlinked front-end AppTier on the data flow diagram. In StartPoint, you can move the Add-on AppTier to the correct order and add links to the Precise environment. The Add-on AppTier name (for example: _MyApp or _MyApp_2) appears as a workspace on the StartPoint screen. See Installation and Administration.

About viewing an Add-on AppTier in Insight

The Add-on AppTier is displayed in the Overview tab for Insight, either in the Insight Cross-AppTiers workspace or in the workspace for the Add-on AppTier. Grouped entities (by default) are not displayed in the Insight user interface. See the Precise Insight User Guide for more details.

About viewing an Add-on AppTier in Report Manager

Report Manager compares the actual performance metrics to an appropriate baseline and identifies significant deviations. When you select a report, Report Manager displays its assigned reports in the Reports Table. Report Manager displays the following types of reports:

- Top-*n* reports to identify major business activities and resource hogs
- Trend reports to establish and analyze long-term system behavior
- Exception reports to identify deviations (from an established baseline) also from long-term system behavior

Report Manager reports are grouped into the following six different sets:

- Profile
- Exceptions
- Availability
- Load Balancing
- Capacity Planning
- Customized

See Precise Report Manager User Guide for more information.

About viewing an Add-on AppTier in AdminPoint

AdminPoint operations include a number of workspaces used to configure or update Add-on AppTier instances, after completing installation.

Add-on AppTier installation package definitions

This section includes the following topics:

Add-on AppTier installation package descriptor file

The Add-on AppTier installation package zip file should be accompanied by a descriptor XML file. AdminPoint receives the descriptor XML file when the Add-on AppTier is plugged-in.

The Add-on AppTier installation descriptor file carries the same name as the Add-on AppTier installation package zip file. The descriptor file extension is xml.

Add-on AppTier installation package descriptor file example

The following is an example of an Add-on AppTier installation package descriptor file.

```
<sdk-package>
     <apptier-name> Generic Mail Server </apptier-name>
     <package-files>
          <package-file package-to-set="true"> GenericMailServer.zip
          </package-file>
     </package-files>
</sdk-package>
```

The following is the basic hierarchy of an XML-based descriptor file:

The root element <sdk-package>; has no attributes

- **AppTier name element; has no attributes**. Its text value is the Add-on AppTier name and it is used in AdminPoint plug-in dialog screen.
- **Package files element has two Package file child elements**. Their text values are the names of the Add-on AppTier installation package zip file and the descriptor XML file.
- The Package file element for the Add-on AppTier installation package zip file should have one attribute named package-to-set with a value of "true."

Add-on AppTier installation package contents

The Add-on AppTier plug-in installation package defines to the Precise environment how the information is structured. It defines the general characteristics of the application, entities, and counters.

There are several files inside the installation package. These XML files are prepared before Add-on AppTier installation on a server. All of the files are placed together in a zip file.

The following files are found in the installation package:

- External Technology Designer (ETD) file
- GUI resource file or files

The software designer assigns a version number to the Add-on AppTier installation package in order to differentiate between future versions of the information structure that may not be compatible.

The name of the installation package zip file should be as follows: I3SDK_*application-name_version*.zip

Table 1 Installation package definitions

Element	Description
I3SDK	This the root element and identifies the installation package for the Add-on AppTier. Value: always I3SDK
application-name	Defines the application name, maximum 10 characters. Value: String
version	Identifies the current version for this installation package. Value: Numeric

ETD file

The ETD file defines entities and counters for each performance record.

ETD file example

The following is an example of the ETD file in the Add-on AppTier installation package. Notice the hierarchy of all elements, attributes, and sub-elements.

The following is the basic hierarchy of an XML-based ETD file:

- The root element is <SDK>
- General characteristics of the application and their attributes
- Entities and their attributes
- Counters and their attributes

```
<etd version="1.0">
     <version major="0" minor="1"/>
     <!-- application level definitions -->
     <application id="MS" tid="1" internal_name="MAILSERVER" range="20000">
          <integration network="no" availability="no" instance-is-server="no" agent-
type="file" /> <!-- "file" or "arm" collection mode -->
          <pw threshold="0"/>
          <sla-entity eid="A" />
          <collection dir="products/sdk/MS" remote="no" min-activity-report="3" max-
activity-report="15" availability-report="15" />
          <service-time>
               <counter tid="100" />
          </service-time>
          <zoom-displays />
          <performance />
     </application>
     <entities>
          <!-- mail msg, activity -->
          <entity eid="A" tid="501" plural-tid="502" grouping-tid="503" plural-
grouping-tid="504" length="256">
```

```
               <display mandatory="yes" order="1" />
               <data class="activity" type="varchar" dbcolumn="MAIL_MSG"
mandatory="yes" last-summary="all" collection-id="msg" />
               <load allow-drop="no" />
               <integration alerts="no" foresight="yes" />
               <service-time />
          </entity>
          <!-- sender, user classification -->
          <entity eid="U" tid="511" plural-tid="512" grouping-tid="513" plural-
grouping-tid="514" length="256"
               <display mandatory="no" order="6" />
               <data class="user_name" type="varchar" dbcolumn="SENDER"
mandatory="yes" last-summary="all" collection-id="sender" />
               <load allow-drop="no" />
               <integration alerts="no" foresight="yes" />
               <service-time />
          </entity>
          <!-- sender ip, "client ip" classification -->
          <entity eid="C" tid="521" plural-tid="522" grouping-tid="523" plural-
grouping-tid="524" length="256">
               <display mandatory="no" order="7" />
               <data class="client_ip" type="varchar" dbcolumn="SENDER_IP"
mandatory="yes" last-summary="all" collection-id="C_IP" />
               <load allow-drop="no" />
               <integration alerts="no" foresight="yes" />
               <service-time />
          </entity>
          <!-- msg type, "user defined" classification, no grouping -->
          <entity eid="T" tid="531" plural-tid="532" length="256">
               <display mandatory="no" order="9" />
               <data class="user-defined" type="varchar" dbcolumn="MSG_TYPE"
mandatory="yes" last-summary="all" collection-id="type" />
               <load allow-drop="no" />
               <integration alerts="no" foresight="yes" />
               <service-time />
          </entity>
          <!-- instance - name of mail server -->
          <entity eid="I" tid="541" plural-tid="542" grouping-tid="543" plural-
grouping-tid="544" length="256">
               <display mandatory="yes" order="2" />
               <data class="instance" type="varchar" mandatory="yes" last-
summary="all" />
               <load allow-drop="no" />
               <integration alerts="no" foresight="yes" />
               <service-time />
          </entity>
          <!-- server machine -->
          <entity eid="S" tid="551" plural-tid="552" grouping-tid="553" plural-
grouping-tid="554" length="256">
               <display mandatory="yes" order="3" />
               <data class="server" type="varchar" mandatory="yes" last-
summary="all" />
               <load allow-drop="no" />
```

```
                <integration alerts="no" foresight="yes" />
                <service=time />
            </entity>
        </entities>
        <counters>
            <!-- delivery time - service time -->
            <counter tid="100">
                <display color="34" colorRGB="123" sortable="yes" sum-
format="duration-HH:MI:SS" avg-format="avgDuration-HH:MI:SS.TTT" />
                <data class="service" type="float" dbcolumn="DELIVERY_TIME"
mandatory="yes" func="SUM" collection-id="delivery" />
            </counter>
            <!-- msg size - statistical counter -->
            <counter tid="110"
                <display color="30" colorRGB="125" sortable="yes" sum-
format="byteFloat" avg-format="byteFloat" />
                <data class="statistic" type="float" dbcolumn="MSG_SIZE"
mandatory="yes" func="SUM" collection-id="size" />
            </counter>
            <!-- recipients - execution counter -->
            <counter tid="120">
                <display color="33" colorRGB="126" sortable="yes" sum-format="number"
avg-format="" />
                <data class="execution" type="float" dbcolumn="REQUESTS"
mandatory="yes" func="SUM" collection-id="recipients" />
            </counter>
        </counters>
        <misc-fields>
            <!-- timestamp field -->
            <field tid="999">
                <data class="timestamp" type="timestamp" dbcolumn="TIMESTAMP"
mandatory="yes"
                <collection-id="D" />
            </field>
        </misc-fields>
        <agent-installer-params />
</etd>
```

About ETD file definitions

Each entity and each counter has a set of properties that describe how they are received, stored, and displayed in Precise products.

The following table specifies version types under the EDT file SDK root element.

Table 2 Types of versions under the ETD file SDK root element

Version Types	Description
major	For documentation of the Add-on AppTier installation package version.

Version Types	Description
minor	For documentation of the Add-on AppTier installation package version.

About ETD file application-level entities

Application-level definitions are a set of properties that describe the scope and display of data from the Add-on AppTier application.

The following table specifies the various identifiers for the add-on technology. The <application> element is a mandatory child element of the root element.

Table 3 Application element positioned under the Precise root directory

Application Attribute Name	Description
tid	The title ID for the technology name. Value: Numeric. Always exists in the file or files. Mandatory: Yes
internal_name	Internally identifies a Precise product. Value: max length of 18. Always uppercase letters or digits only (no blanks or underscore used). Usually the value is derived from the title of the application. Mandatory: Yes
id	Identifies the add-on technology. Value: always 2 characters; provided as part of the Precise registration procedure. Mandatory: Yes
range	Used in conjunction with tid attributes to make the values unique across technologies. Value: Numeric; provided as part of the Precise registration procedure. Mandatory: Yes

The following table specifies the collection mode and possible integration with Precise components. The <integration> element is a mandatory child element of <application>.

Table 4 Integration element positioned under the application element

Integration Attribute Name	Description
network	Specifies whether integration network monitoring is currently available. Must always be either Yes or No. Value: Yes, if integration network monitoring is currently supported. Value: No, if integration network monitoring is currently not supported. "No" is the default value. Mandatory: No
availability	Specifies whether monitoring for availability is currently available. Must always be either Yes or No. Value: Yes, if monitoring for availability is currently supported. Value: No, if monitoring for availability is currently not supported. "No" is the default value. Mandatory: No (for future use)
instance-is-server	Specifies whether the same entity is used as both instance and server. Must always be either Yes or No. Value: Yes, if instance is installed on a server. Value: No, if instance is not installed on a server. "No" is the default value. Mandatory: No (for future use)
agent-type	Identifies the File or ARM collection. Value: either "file" or "arm". Mandatory: Yes

The following table specifies possible parameters for the PMDB load process. The <pw> element is an optional child element of <application>.

Table 5 PMDB element positioned under the application element

PW Attribute Name	Description
threshold	The load threshold. Rows with a service time smaller than specified are collapsed into one row. Value: Numeric, if specified. Mandatory: No (for future use)

The following table specifies the entity that is used for SLA calculations. The <sla-entity> element is an optional child element of <application>.

Table 6 sla-entity element positioned under the application element

sla-entity Attribute Name	Description
eid	If an entity with a Precise classification activity was defined, put its entity as value. Otherwise, use an empty string.

The following table specifies the collection properties. The <collection> element is a mandatory child element of <application>.

Table 7 Collection element positioned under the application element

Collection Attribute Name	Attributes
dir	The relative path under the Precise Listener where Collector files are harvested. A valid path name always includes a forward slash "/" as a file separator. For example: products/sdk/MS. Mandatory: Yes
remote	Specifies whether remote collection is available. The server where the Collector runs is different than the server where the monitored application runs. Value: Yes, No. "No" is the default value. Mandatory: No
min-activity-report	Specifies the minimum frequency for reporting activity data (largest time slice). Value: Numeric, if specified. Mandatory: No (for future use)
max-activity-report	Specifies the maximum frequency for reporting activity data (smallest time slice). Value: Numeric, if specified. Mandatory: No (for future use)
availability-report	Specifies the frequency for reporting availability data (size of time slice). Value: Numeric, if specified. Mandatory: No (for future use)

The following table specifies the counters that make up the application's service time. The <service-time> element is a mandatory child element of <application>.

Table 8 Service-time element positioned under the application element

Service-time	Description
service-time	Specifies the display order for the service time breakdown in Insight and Report Manager. Must have at least one child element <counter>; multiple children are allowed. Each counter child must have a "tid" attribute with a value equal to the counter's "tid". (The child for the <counters> element is equal to the child of the root.) This element has no attributes.

The following table specifies a zoom display. The <zoom-displays> element is an optional element of <application>.

Table 9 Zoom-displays element positioned under the application element

Zoom-displays Element Name	Description
zoom-displays	This element has no attributes (for future use).

The following table specifies performance. The <performance> element is an optional element of <application>.

Table 10 Performance element positioned under the application element

Performance Element Name	Description
performance	This element has no attributes (for future use).

About ETD file entities

This section contains functional and technical specifications for Add-on AppTier entities.

The following table specifies the entities for the Add-on AppTier. The <entities> element is a mandatory child element of the root.

Table 11 Entities element positioned under the root element

Entities Element Name	Description
entities	Must have at least one child element <entity>. Multiple children are allowed. The Instance and Server entities are mandatory. There must be one entity with a class activity. This element has no attributes. Only one entity per class attribute: client_ip, activity, program, application, user_name, instance or server, can be defined. Multiple entities with user-defined class are allowed.

The following table defines one entity. The <entity> element is a mandatory child element of <entities>. Multiple instances are allowed.

Table 12 Entity element positioned under the entities element

Entity Attribute Name	Description
eid	Uniquely identities an entity. Value: One character, uppercase letter only. Mandatory: Yes
tid	The Title ID for an entity, as it appears in the drop-down menu list. Value: Numeric, always exists in the file or files. Mandatory: Yes
plural-tid	The Title ID for an entity in plural form, as it appears in the graph title. Value: Numeric, always exists in the file or files. Mandatory: Yes
grouping-tid	If specified, this entity has a grouping function. It specifies the Title ID for a grouped entity, as it appears in the drop-down menu list. Value: Numeric, always exists in the file or files. Mandatory: No
plural-grouping-tid	If specified, this entity has a grouping function. It specifies the Title ID for a grouped entity in plural form, as it appears in the graph title. Value: Numeric, always exists in the file or files. Mandatory: No
length	The maximum length value in numbers. Value: Numeric Mandatory: Yes

These entities have the following length limitations:

- Entity Name (singular), maximum 30 characters
- Entity Name (plural), maximum 30 characters
- DB Column Name, 2 - 20 characters

The following table defines the display properties of the entity. The <display> element is a mandatory child element of <entity>.

Table 13 Display element positioned under the entity element

Display Attribute Name	Description
mandatory	Defines whether the entity graph can be hidden in Insight display settings. Value: Yes, No Mandatory: Yes
order	Defines the position of the entity's graph in the Insight "all" display. Value: Numeric Mandatory: Yes

The following table defines the database related properties. The <data> element is a mandatory child element of <entity>.

Table 14 Data element positioned under the entity element

Data Attribute Name	Description
class	Defines the classification used to map the entity to the Insight Cross-AppTier display. Valid values: "activity", "program", "application", "user_name", "instance", "server", "client_IP", or "user_defined". Mandatory: Yes
type	Defines the data type of a database column. Value: "varchar" Mandatory: Yes
dbcolumn	The name of the database column. Value: Yes, No Mandatory: Yes Instance and Server entities have fixed database column names that are automatically set by Precise during plug-in installation. The attribute cannot be specified for these entities in the ETD file.
mandatory	Defines whether the database column is nullable. Values: Yes or No Mandatory: yes

Data Attribute Name	Description
last-summary	Specifies whether an entity exists in all PW summary levels. Valid values: "all", "T", "H", "G", "D", or "W". Mandatory: No (for future use)
collection-id	The name of a field in a Collector file. Value: a valid XML tag. Mandatory: Yes NOTE: Instance and Server entities have no collection-id. The values for these entities in the

Collector file are specified in the rowset level; not in the row level.

The following table specifies loading parameters. The <load> element is an optional child element of <entity>.

Table 15 Load element positioned under the entity element

Load Element	Description
allow-drop	Specifies whether this entity can be dropped during loading. (This option appears in PW load process parameters.) Value: Yes, No Mandatory: No (for future use)

The following table defines whether an entity can be filtered. The <filter> element is an optional child element of <entity>. If specified, it must have one child element <value> with a text value.

Table 16 Filter element positioned under the entity element

Filter Element	Description
filter	This element has no attributes (for future use).

The following table specifies whether this entity is applicable for other Precise products besides Insight. The <integration> element is an optional child element of <entity>.

Table 17 Integration element positioned under the entity element

Integration Attribute Name	Description
alerts	Specifies whether this entity is applicable for Alerts. Values: Yes, No Mandatory: No (for future use)

Integration Attribute Name	Description
report manager	Specifies whether this entity is applicable for Report Manager. Values: Yes, No Mandatory: Yes

The following table specifies whether this entity has a special service time breakdown. The <service-time> element is an optional element of <entity>.

Table 18 Service-time element positioned under the entity element

Service-time Element	Description
service-time	This element has no attributes (for future use).

About etd file counters

This section contains functional and technical specifications for Add-on AppTier counters.

The following table specifies the counters for the Add-on AppTier. The <counters> element is a mandatory child element of the root.

Table 19 Counters element positioned under the root

Counters Element	Description
counters	Specifies the counters for the Add-on AppTier. The <counters> element must have at least one child element <counter>. There must be at least one counter with a class service. It must be the first one in the counters section (in the etd file). This element has no attributes. Execution counter must be defined in the etd.xml file (only one can be defined). Although the counter is defined, the agent does not need to report this counter in the data file.

The following table defines the attribute for one counter only. The <counter> element is a mandatory child element of <counters>.

Table 20 Counter tid element positioned under the counters element

Counter Attribute Name	Description
tid	The Title ID for a counter. Value: Numeric, always exists in file or files. For different counters, the values of tid must be specified in intervals of 10. Mandatory: Yes

The following table defines the display properties of the counter. The <display> element is a mandatory child element of <counter>.

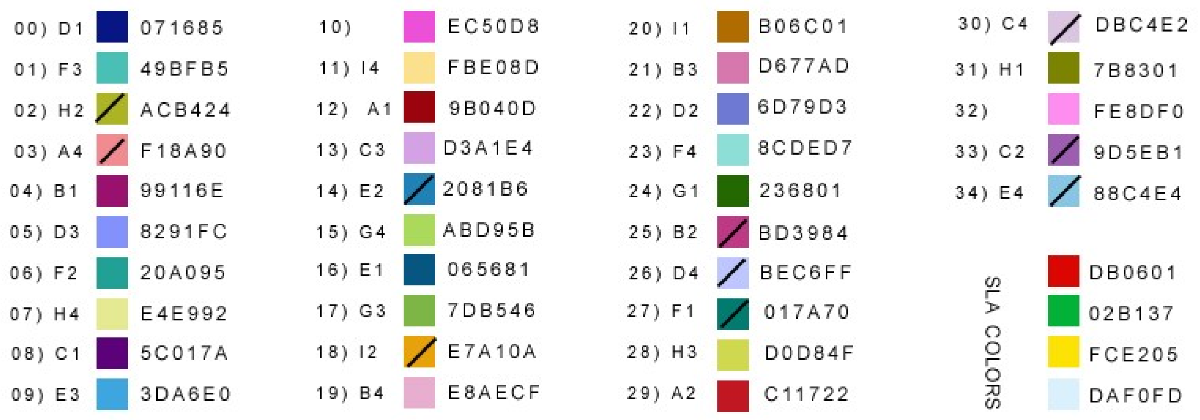
Table 21 Display element positioned under the counters element

Display Attribute Name	Description
color	Specifies the color code for Insight bar graphs. Value: Figure 1 The value is the sequential number before the bracket. Mandatory: Yes
colorRGB	Specifies the color code for Report Manager reports. Value: Figure 1 The value is the hexadecimal value on the right of the color that then needs to be converted to its equivalent decimal value. For example color with the code 0 and hexdecimal value 071685 becomes 464516 decimal. Mandatory: Yes
sortable	Specifies whether a counter appears in an Insight sort drop down list. Value: Yes, No Mandatory: Yes
sum-format	Displays the format summed value. Value: see counter display formats table for Display sub-elements. Mandatory: Yes
avg-format	Displays the format average value. Value: see counter display formats table for Display sub-elements. Mandatory: Yes

The following figure displays the color codes that can be used for the color and color RGB attributes of the display element.

Ignore the diagonal lines in some of the color boxes.

Figure 1 Color code table



00) D1	071685	10)	EC50D8	20) I1	B06C01	30) C4	DBC4E2			
01) F3	49BFB5	11) I4	FBE08D	21) B3	D677AD	31) H1	7B8301			
02) H2	ACB424	12) A1	9B040D	22) D2	6D79D3	32)	FE8DF0			
03) A4	F18A90	13) C3	D3A1E4	23) F4	8CDED7	33) C2	9D5EB1			
04) B1	99116E	14) E2	2081B6	24) G1	236801	34) E4	88C4E4			
05) D3	8291FC	15) G4	ABD95B	25) B2	BD3984					
06) F2	20A095	16) E1	065681	26) D4	BEC6FF	SLA COLORS	DB0601			
07) H4	E4E992	17) G3	7DB546	27) F1	017A70		02B137			
08) C1	5C017A	18) I2	E7A10A	28) H3	D0D84F		FCE205			
09) E3	3DA6E0	19) B4	E8AECF	29) A2	C11722		DAF0FD			

The following table specifies counter display formats for Display sub-elements positioned under the counter sub-element.

Table 22 Counter display formats for the Display sub-elements positioned under the counter sub-element.

Format	GUI Definition	Description
Duration	sum-format: HH:MI:SS avg-format: HH:MI:SS.TTT	Time counter (format for Service-time classification)
int Count	number	Integer counter, big numbers, display as K/M/G.
float Count	numberFloat	Execution classified counter must have sum-format="number" and avg-format="". Statistical classified counters have number, numberFloat, or byteFloat.
Bytes	byteFloat	Bytes; may be displayed as bytes, KB, MB, GB.

The following table defines database-related properties. The <data> element is a mandatory child element of <entity> and <counter>.

Table 23 Data element positioned under the counter element

Data Attribute Name	Description
class	Defines the classification used to build a service time breakdown and statistical graphs. Value: "service", "execution", or "statistic". Mandatory: Yes

Data Attribute Name	Description
type	Defines the data type for a database column. Value: float Mandatory: Yes
dbcolumn	The name of a database column. Value: valid database column. Mandatory: Yes Execution classified counter has a fixed dbcolumn name: "REQUESTS".
mandatory	Defines whether the database column is nullable. Value: Yes, No Mandatory: Yes
func	An aggregation function for the PW summary process. Value: sum Mandatory: No
collection-id	The field name in a Collector file. Value: a valid XML tag. Mandatory: Yes

The following table specifies additional fields that are neither entities or counters. The <misc-fields> element is a mandatory child element of the root.

Table 24 Misc-fields element positioned under the root element

Misc-fields Element	Description
misc-fields	Must have at least one child element <field>. Multiple children are allowed. Currently, one field child is mandatory: timestamp field. This element has no attributes.

The following table defines the field for a counter. The <field> element is a mandatory child element of <misc-fields>.

Table 25 Field element positioned under misc-fields element

Field Attribute Name	Description
tid	A dummy Value. Value: "999" Mandatory: Yes

The following table defines the database related properties. The <data> element is a mandatory child element of <field>.

Table 26 Data element positioned under field element

Data Attribute Name	Description
Class	Defines the classification. Value: "timestamp" Mandatory: Yes
dbcolumn	The name of a database column is fixed. Value: "TIMESTAMP" Mandatory: Yes
mandatory	Defines whether the database column is nullable. Value: Yes Mandatory: Yes
Type	Defines the data type of a database column. Value: timestamp Mandatory: Yes
collection-id	The field name in a Collector file. Value: a valid XML tag. Mandatory: Yes

The following table specifies the name of the timestamp field in the Collector file. The timestamp field is a mandatory field child element of <misc-fields>.

Table 27 Timestamp field positioned under the misc-fields element

Timestamp Field	Description
timestamp	The collection-id attribute.

The following table specifies needed parameters for installation. The <agent-installer-params> element is an optional child element of the root.

Table 28 Agent-installer-params element positioned under the root element

Agent-installer-params Element	Description
agent-installer-params	This element has no attributes (for future use).

The following are counter guidelines:

- Service Time, at least one is required
- Executions, always fixed
- Counter Name, maximum 15 characters
- DB Column Name, between 2 - 20 characters

About the titles file or files

The titles file or files specify captions and headings that are displayed in Precise products. These are the application names as well as entity and counter names. A different titles file exists for each required language. At the least, one English-language titles file named titles-en.xml has to be created.

> Precise displays captions in English and Japanese only for Precise supported technologies.

The following is an example of a titles file in English.

```
<titles-en>
  <tid="1"> Generic Mail Server </t>
  <displays>
  </displays>
  <entity-single>
    <tid="501"> Mail Msg </t>
    <tid="511"> Sender </t>
    <tid="521"> Sender IP </t>
    <tid="531"> Msg Type </t>
    <tid="541"> Instance </t>
    <tid="551"> Server </t>
  </entity-single>
  <entity>
    <tid="502"> Mail Msgs </t>
    <tid="512"> Senders </t>
    <tid="522"> Sender IPs </t>
    <tid="532"> Msg Types </t>
    <tid="542"> Instances </t>
    <tid="552"> Servers </t>
  </entity>
  <entity-grouping-single>
    <tid="503"> Mail Msgs (Grouped) </t>
    <tid="513"> Senders (Grouped) </t>
    <tid="523"> Senders Location </t>
    <tid="543"> Instances (Grouped) </t>
    <tid="553"> Servers (Grouped) </t>
  </entity-grouping-single>
  <entity-grouping>
    <tid="503"> Mail Msgs (Grouped) </t>
    <tid="513"> Senders (Grouped) </t>
    <tid="523"> Senders Location </t>
    <tid="543"> Instances (Grouped) </t>
```

```
        <tid="553"> Servers (Grouped) </t>
    </entity-grouping>
    <fields>
        <tid="100" class="service" > Delivery </t>
        <tid="110" class="statistic" > Msg Size </t>
        <tid="120" class="execution"" > Recipients </t>
    </fields>
</titles-en>
```

The titles file naming convention (for languages other than English) is as follows: `titles_`*`language-code`*`.xml`

The following is the basic hierarchy of an XML-based titles file:

- The root element is named with the language code; has no attributes
- Add-on AppTier name
- Entity names in a single variation
- Entity names in a plural variation
- Entity group names in a single variation
- Entity group names in a plural variation
- Counter names

The following table specifies the properties for the Add-on AppTier name element. The <t> element is a mandatory child element of the root element.

Table 29 Add-on AppTier name element properties

t Atribute Name	Description
id	The value is the same as the value for the attribute tid element <application> in the ETD file.

The text value for the <t> element, under the root element, is the Add-on AppTier name.

All elements under the root element have the same basic hierarchy; multiple <t> sub-elements, each specifying the name of one entity or counter. All <t> sub-elements have one attribute named "id" with a value that connects the ETD file entity or counter to its name.

<t> sub-element properties vary under the following elements:

- <entity> element
- <entity-single> element
- <entity-grouping> element
- <entity-grouping-single> element

The following table specifies <t> sub-element properties for various elements.

Table 30 Sub-element properties for <t> under various elements

t Attribute Name	Description
id	Under the <entity> element, the value is the same as the <entity> element's attribute plural-tid value for the relevant entity from the ETD file. Under the <entity-single> element, the value is the same as the relevant <entity> element's attribute tid from the ETD file. Under the <entity-grouping> element, the value is the same as the relevant <entity> element's attribute plural-grouping-tid from the ETD file. Under the <entity-grouping-single> element, the value is the same as the relevant <entity> element's attribute grouping-tid from the ETD file.

The text value for the <t> element is the entity name as it should be displayed by Precise products. The following table specifies the properties for <t> sub-elements under the <fields> element.

Table 31 Sub-element properties for <t> under the <fields> element

t Attribute Name	Description
id	The value is the same as the <counter> element's attribute tid value for the relevant counter from the ETD file.
Class	The value is the same as the <data> sub-element's attribute class for the relevant counter element <counter> from the ETD file.

The text value for the <t> element is the counter name as it should be displayed by Precise products.

Insight ARM extension library calls

This section includes the following topic:

- About the Insight ARM extension dynamic library

About the Insight ARM extension dynamic library

The Insight ARM extension dynamic library uses several configuration parameters which are stored in a shared memory segment. To find the shared memory, some environment variables must be defined for the Add-on AppTier application instance. This method passes some parameters to the Insight ARM extension dynamic library which runs inside the Add-on AppTier application. Additional configuration details, for example: the communication port (between the Insight ARM extension dynamic library and the Insight SDK Collector), the log file and log level are found in the shared memory.

The following table describes the three groups of internal ARM calls.

Table B-1 Groups of internal ARM calls

Group Type	Description
Initialization	An example of calls are: arm_init or arm_getid. The Initialization group includes the following: • **Main initialization**. This initializes all of the information regarding the running application. • **Transaction initialization**. This defines the transaction name and transaction details. All transaction information is saved by the Insight SDK extension dynamic library until the application ends.
Transaction status	An example of calls are: arm_start, arm_stop, and arm_update. The Transaction status group includes the following: • All application information • All transaction definitions • Current transaction details
Termination	For example: arm_end This signals the end of the application. In this call, the Insight ARM extension dynamic library releases the allocated shared memory and closes the IPC channel. If the user forgets to call a stop function, all resource processes are automatically released when the process ends.

Each transaction status event is sent to the Insight SDK Collector.

The following are transaction function calls:

- **Transaction start (arm_start call)**. It is called by the application at the start of each transaction.
- **Transaction stop (arm_stop call)**. It is called by the application at the end of each transaction.
- **Transaction update (arm_update call)**. The application can call any time after transaction start and before transaction stop for any transaction.

The Insight ARM extension dynamic library passes transaction status to the Insight SDK Collector (psi_sdk_poll process).

The following are transaction status information properties:

- ARM application name
- ARM user name
- ARM transaction name
- ARM transaction ID
- ARM transaction status (sent during stop status)
- ARM correlation ID
- Timestamp
- User metrics
- User additional buffer

All of the properties are sent in the following format:

```
property_name1=value1, property_name2=value2
```

XML collector file conventions and formats

This section includes the following topics:

- About the XML Collector file
- About XML Collector file rows

About the XML Collector file

The Collector file is an XML file with one or more rows that are generated from user-defined definitions. These data files, collected by the Add-on AppTier Collector, are put in a Collector directory in the <*i3_root*> for local client harvesting.

The Collector file location is `[i3]/products/SDK/tech_code`.

The location file is specified in the ETD file which is part of the Add-on AppTier installation package, described in Appendix A. A Collector file has the following properties:

- File name and location
- File format and contents (file header and file body)
- File closure and stamp
- File restrictions

A Collector file has the following naming convention:

> *tech-code_instance-name*
> *_monitored-server_unique-id.[prf|avl]*

The following table shows the elements of a Collector file name.

Table 1 Elements of a Collector file name

Element	Description
Tech-code	2-letter technology code (U1, U2, and so on). The technology code is assigned to your Add-on AppTier application during registration and is specified in the ETD file.
Monitored-server	Name of monitored (instance) server (for example: _MyServer). The monitored server is installed through AdminPoint.
Instance-name	Name of instance (added previously through AdminPoint).
Unique-id	A sequence number that is generated by the Insight SDK Collector and ensures the file name is unique. There is no standard or restriction to the sequence number range and format.
File extension	Prf is used for performance data; avl is used for availability data (not currently supported).

The body of the Collector file contains multiple row elements, each of them representing a summary of a distinct activity, collected after a default 15-minute timeslice. The entities and counters are defined as child elements with text values.

The following is one example of a GMS Collector file.

```
Filename: U1_MyServer_1104643728.xml
<rowset savvy="MS"instance="MailServer1" server="toi-vm-1">
    <row>
        <D> 2007-07-1108:15:00.0 </D>
        <msg> change request </msg>
        <sender> Eliza Doll </sender>
        <C_IP> 10.1.1.4 </C_IP>
        <type> mail </type>
    </row>
</rowset>
```

The following is the basic hierarchy for an XML-based Collector file:

- The root element <rowset> has three attributes
- Multiple <row> elements under the root; each row represents all invocations of a unique activity during a 15-minute time slice.

The following table specifies the attributes for the root <rowset> element.

Table 2 Root <rowset> element attributes

Rowset Attribute Name	**Description**
Savvy	A 2-letter technology code (U1, U2, and so on). The technology code is assigned to your Add-on AppTier application during registration, and is specified in the ETD file.
Instance	Instance name (added earlier through AdminPoint).
Server	Monitored (instance) server name. For example: _MyServer. The monitored server is installed through AdminPoint.

About XML Collector file rows

The body of an XML Collector file is a series of <row> elements. Each row specifies the activation of a unique activity in the Add-on AppTier during a 15-minute time slice.

The row element has sub-elements that specify values for entities and counters. The sub-element name is the same as the data sub-element's attribute collection-id value, for entities, counters, and the timestamp field in the ETD file.

The following example shows how attributes in the ETD file relate to XML tags in the SDK Collector file.

Three consecutive dots (...) represent additional standard code not given.

```
<etd version="1.0">
    ...
    <entities>
        <!-- mail msg, activity -->
        <entity eid="A" tid="501" ...>
            <data class="activity" dbcolumn="MAIL_MSG" collection-id="msg" />
            ...
    </entities>
    ...
    <misc-fields>
        <field tid="999">
            <data class="timestamp" dbcolumn="TIMESTAMP" collection-id="D" />
        </field>
    </misc-fields>
</etd>
<rowset savvy="MS" instance="MailServer1" server="toi-vm-1">
    <row rownum="1">
        <D> 2007-07-11 08:15:00.0 </D>
        <msg> change request </msg>
        ...
    </row>
</rowset>
```

All row sub-elements must be specified as a collection-id for one of the following: entity, counter, or timestamp field in the ETD file.

Observe the following standards for Collector file rows sub-element text values.

- **Timestamp field**. Values should be specified by GMT time zone. Mandatory format: YYYY-MM-DD-HH24:MM:SS
- **Entities**. Values should be enclosed with the following: <! [CDATA [….]] >
- **Counters**. Service time counters must be reported in the following format: sss.mmm where seconds and milliseconds with a decimal point have a character dot.

About file closure and stamp

The SDK Collector file cannot be harvested to the Precise FocalPoint server until the file is closed and stamped by the SDK Collector. The SDK Collector closes and stamps the file to signal file readiness for subsequent processing and loading to the PMDB.

In an ARM collection mode, closing and stamping is handled by the Insight SDK Collector. In a file collection mode, the third-party SDK Collector needs to handle these tasks.

To stamp the Collector file, the SDK Collector must create an empty file (file with size zero [0] bytes) with the same name as the XML Collector file but with a different extension.

The following are XML Collector file extensions:

- .pok extension for .prf
- .aok file for .avl

The following are file name examples that are produced by the SDK Collector for the GMS Add-on AppTier that is installed on server MailServer-1 and monitors instance MailInstance-1.

MS_MailInstance-1_MailServer-1_1189403.prf

where file size is 135 KB.

MS_MailInstance-1_MailServer-1_1189403.pok

where file size is 0 KB.

About time slices and activity summaries

Collector files are harvested at time periods having fixed intervals. These time intervals are called time slices and their length is fixed at 15 minutes. The Precise harvester waits one minute, after a time slice has passed, and searches for new files to move to the Precise FocalPoint server.

For example: at noon, harvesting starts at the following times: 12:01, 12:16, 12:31, and 12:46.

The Precise harvester searches for and processes only stamped Collector files (such as a couple of identically named .prf and .pok files) and only in the location specified in the ETD file.

The harvesting schedule just mentioned is similar to the one that takes place for an Insight Collector and for component Collectors. The Collectors usually produce one file that includes a summary of all recorded activity. "Summary" is used here to mean the aggregation of performance calculations for all invocations for the same activity during the time slice.

For example: all invocations for a Windows calculator program on the same computer, by the same login user will be aggregated into one <row> element in an OS Collector file. The timestamp in the aggregated row will point to the start of the time slice. This standard is used to optimize the Collector's footprint on the Operating System, to reduce the Collector file's size and handle the issue of scalability.

The Insight SDK Collector follows this standard when in ARM collection mode. However, this standard is not mandatory for SDK Collector file mode. The Collector can create as many files in the same time slice and use whatever aggregation mechanism needed. The SDK Collector is responsible to manage such issues as footprint, file size, and scalability.