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.