OpenPegasus Tracing User Guide

Update for Pegasus 2.14,
Dec 2014

OpenPegasus provides a tracing facility that helps to investigate the cause of a problem. For example, if requests abort, performance is reduced or unexpected responses appear, the trace messages can indicate where and when the problem occurred.

1 Trace Configuration

The tracing of OpenPegasus can be configured by setting the following properties:

- `traceLevel`
- `traceComponents`
- `traceFacility`
- `traceMemoryBufferKbytes`
- `traceFilePath`

Refer to the cimconfig command man page for more information regarding setting/unsetting of the OpenPegasus configuration properties.

The trace properties `traceFilePath`, `traceLevel`,`traceComponents` and traceFacility can be changed dynamically. Hence, there is no need to re-start OpenPegasus. Only the `traceMemoryBufferKBytes` property must be set before OpenPegasus starts.

1.1 traceLevel

The `traceLevel` property sets the required trace level. The trace level indicates the level of information to be included in the trace output.

The following are the valid trace levels.

Level #	Description
0	Tracing is switched off.
1	Severe trace and log messages (if traceComponents is set to LogMessages)
2	Basic logic flow trace messages, minimal data detail (default)
3	Intra function logic flow and moderate data detail
4	High data detail
5	High data detail + Function entry/exit

Each successive level provides more detailed information and includes information from the levels below. Remember to set the list of components to be traced in the `traceComponents` property.

The default trace level is “0”.

1.1.1 Example:

Command to enable trace level 3 for intra function logic and moderate data detail:

```
cimconfig –s traceLevel=3 -c
```

1.2 traceComponents

The `traceComponents` property allows to enable tracing selectively for a given OpenPegasus component or a list of components. A list is a set of components separated by a comma.

The special component “All” enables tracing for all available components.
If `traceComponents` is set to an empty string, tracing is switched off.

The following table lists the available components:

Authentication	Authorization	CIMExportRequestDispatcher
CIMOMHandle	CMPIProvider	CMPIProviderInterface
CQL	Config	ControlProvider
DiscardedData	Dispatcher	EnumContext (Ver: 2.14)
ExportClient	Http	IndicationGeneration
IndicationHandler	IndicationReceipt	IndicationService
L10N	Listener	LogMessages
MessageQueueService	ObjectResolution	OsAbstraction
ProviderAgent	ProviderManager	Repository
SSL	Server	Shutdown
StatisticalData	Thread	UserManager
WQL	WsmServer	Xml
XmlIO		

1.2.1 Example:

Command to enable tracing of XML requests and responses.

```
cimconfig –s traceComponents=XmlIO –c
```

1.3 traceFacility

The traceFacility property specifies the target facility to which trace messages are written:

traceFacility	Description
File	The trace messages are written to the file specified by traceFilePath.
Log	The trace messages are written to the logging facility using a logging priority TRACE. (The logLevel property must be set to TRACE.)

Memory	The trace messages are written to a memory buffer. It can be found in a memory dump by searching for the eye-catcher "PEGASUSMEMTRACE" The buffer is organized in a wrap around manner. All messages do have a CR/LF. The last message can be identified by a trailing eye-catcher “*EOTRACE*”.

1.3.1 Example:

Command to route the trace messages into the memory buffer:

```
cimconfig –s traceFacility=Memory -c
```

1.4 traceMemoryBufferKbytes

The `traceMemoryBufferKbytes` property specifies the size of the memory trace facility in kBytes (1024 bytes). The minimum is 16kB.The default is 10240kB. This property is a planned configuration property and cannot be changed dynamically. It becomes active after a restart of OpenPegasus.

1.4.1 Example:

Command to set the memory buffer size to 20MB in the planned configuration.

```
cimconfig –s traceMemoryBufferKbytes=20480 –p
```

1.5 traceFilePath

The `traceFilePath` property specifies the output file if the `traceFacility` is set to File. If the file is specified using a relative path, the file is created relative to PEGASUS_HOME.

A trace file is written for the main process of OpenPegasus and for each of the OOP Agents. For the OOP Agents, the file is extended with: <ModuleName>.<UserName>

1.5.1 Example:

Command to set a full qualified trace file.

```
cimconfig –s traceFilePath=/tmp/Pegasus.trc –c
```

The trace file for an OOP agent may look like this:

```
/tmp/cimserver.trc.OperatingSystemModule.root
```

2 How to use the trace

The tracing facility is designed to be used for in-depth problem determination. This may be necessary during development, but may also be needed in production. To meet the different requirements of these cases, the tracing facility can be tailored in

a) the level of detail using the configuration property traceLevel (1 to 5)
b) the focus on special components of OpenPegasus using the `traceComponents` property
c) the target facility of the trace messages using the `traceFacility` property

2.1 Tailoring the amount and quality of the trace

For tailoring the amount and quality of the trace the properties `traceLevel` and `traceComponents` are used. The `traceLevel` property sets the level of detail of the trace, and the `traceComponents` property specifies which component of OpenPegasus should issue trace messages at all.

2.1.1 The traceLevel

When `traceLevel` is set to 1 the trace contains trace messages of severe error conditions and, if the `traceComponents` is set to "ALL" or includes "LogMessages", the log messages are also written to the trace. By the nature of this trace level, the amount and frequency of these trace messages is very low.
The number and level of detail of the trace messages accelerates from `traceLevel` 1 to 5. `traceLevel` 5 adds the method enter/exit messages and is the highest level of tracing. This level can be used to trace the flow of code execution in OpenPegasus.

Refer to the `traceLevel` configuration property description for more information about the levels of detail provided by the various trace levels.

2.1.2 The traceComponents

Most of the trace components specify one or several modules serving special working units of OpenPegasus. Some trace components have special purpose `traceComponents`. These special purpose `traceComponents` are:

Special purpose traceComponents	Description
`All`	All available components are traced.
`DiscardedData`	Issues a trace message when information is discarded or an operation is cancelled to enable OpenPegasus to proceed.
`LogMessages`	All messages written to the Logging Facility are traced.
`StatisticalData`	Prints statistical data to the trace at level 4. This is not a valid trace component when

	OpenPegasus was compiled without statistical data. (PEGASUS_DISABLE_PERFINST)
XmlIO	Prints the complete CIM-XML messages that OpenPegasus exchanges with clients to the trace.

2.2 Routing trace messages

The traceLevel and traceComponents properties are the filter for the traced messages. To specify the location where the trace messages are finally written, use the traceFacility property.

2.2.1 Writing to file

By setting the traceFacility to File, the trace messages are written to a file. The file is specified by the traceFilePath property. The file is continuously written and is growing constantly. The file can be removed while the OpenPegasus server is running, it is recreated automatically.

2.2.2 Writing to memory

By setting the traceFacility to Memory, the trace messages are written to a memory buffer. The traceFacility Memory has the following attributes:

- The buffer is allocated in a continuous memory block.
- The size of the buffer is specified by the traceMemoryBufferKbytes property.
- The messages are written in a wrap-around manner.
- All messages have a trailing CR/LF.
- The last written message has a trailing "*EOTRACE*".
- If a message does not fit into the memory buffer, it is truncated and "*TRUNC*" is appended.

The buffer can be found in memory dumps by searching for the eye-catcher "PEGASUSMEMTRACE". To get the trace messages into the right order, copy the buffer into an editor of your choice, cut the messages from the start of the buffer until "*EOTRACE*" and append them to the end of the buffer.

2.2.3 Writing to log

By setting the traceFacility to Log, the trace messages are written to the Logging facility with the priority TRACE. In addition the logLevel property has to be set to TRACE. Otherwise the trace messages are discarded.

This facility combines the trace message stream with the log message stream. If the traceComponents property is set to "All" or "LogMessages" and the traceFacility is set to "Log", log messages are no longer written to the trace to avoid duplicate entries.

If your OpenPegasus supports the syslog daemon, you can use the capability of the syslog daemon to manage the trace messages. One of the capabilities of the syslog daemon is to route messages to remote systems.

3 Interpreting the trace output file

The following is the standard trace record output format:

```
<Seconds after 1970>s-<micro seconds>us:<Component Name>
[<ProcessID:ThreadID:File name: Line Number>]: <detailed information>
```

Some of the messages may not include the File Name and Line Number information.

The following example shows a sample trace output file:

```
1225804806s-137994us: ProviderManager
[26772:3086764944:DefaultProviderManager.cpp:517]: Initializing Provider
PG_OperatingSystemProvider
1225804806s-138028us: MessageQueueService
[26772:3086764944:MessageQueue.cpp:188]: MessageQueue::lookup failure - name =
CIMOpRequestDispatcher
```