Atomic Labs

Enterprise Edition

Reactor Configuration Guide

Last update: **June 2010**

Table of Contents

Pion Reactors

This following sections of this document contain notes and other useful information about Pion reactors.

Reactor Type	**Reactor Name**	**See Page**
Collection	Log File Input Reactor	8
	Sniffer Reactor	13
Processing/Transform	Aggregate Reactor	8
	Clickstream Reactor	34
	Content Hash Reactor	45
	Filter Reactor	49
	Fission Reactor	53
	Python Reactor	57
	Script Reactor	63
	Session Filter Reactor	63
	SQL Reactor	66
	Transform Reactor	75
Storage	Database Output Reactor	82
	Google Analytics Reactor	91
	Log Output Reactor	94
	Multi Database Reactor	97
	Omniture Genesis Reactor	102
	Unica OnDemand Reactor	109
	Webtrends Reactor	112

Important Notes About Reactors

This section contains important information common to Pion and/or reactors, including naming conventions and other usage-related conventions, limits, and practices.

Database, Table, and Column Naming Conventions

Naming rules in Pion are a bit more restrictive than some databases, for example SQLite or Oracle. This is because Pion tries to support all databases. Because Pion's naming rules result in names that are the 'lowest common denominator,' any name that is valid in Pion should work in any database.

In Pion, database, table, and column names must conform to SQL92 compliant database rules, meaning that these names must:

1. Start with a letter (a-z or A-Z)
2. Contain only letters (a-z or A-Z), numbers (0-9) and the underscore character (_)
3. Contain less than 32 characters

Note: While Pion Term names may contain dashes (-), periods(.), or the pound/hash sign (#) those characters are not allowed in database column names.

Comparisons

Comparisons are used in the following reactors: Aggregate Reactor, Clickstream Reactor, Database Output Reactor, Filter Reactor, Fission Reactor, Session Filter Reactor, and SQL Reactor.

Comparisons are used as filters, and a series of comparisons is called a filtering rule chain. The following applies to everywhere a series of comparisons is used. Ultimately, the goal of comparisons is to take an event coming into the reactor and get a Boolean (TRUE or FALSE) result.

What the result of any given comparison means is different for each reactor and use case, but the configuration (and the code it uses on the back end) is common:

Each row in the table can be referred to as a "Comparison." The order of comparisons **is** significant because comparisons are processed sequentially.

Each Comparison has: **Term**, **Comparison** (type), **Value**, and matching criteria (**Match All** and **Match All Comparisons**).

- **Term**, defines the value of the term to be used for the comparison logic.
- **Comparison** (type), specifies the type of comparison to perform (greater than, less than, matches, regular expression, etc.).

  Note that the selections available for comparison change depending on the data type of the selected term. If you select a string for example, you cannot choose "less than" and if you select an integer, you cannot execute a regular expression match.

- **Value**, specifies a value to be compared against the value of the selected term.

  For example, if you want only events whose term value is greater than 10 (numeric term type), you would select the Term (in the first column), the value "greater than" (in the second column), and a value of "10" (in the third column).

Matching criteria specify which of the defined comparison conditions must actually match in order to pass the filter.

- **Match All**, when filled, and if the term has multiple values (a very rare occurrence), then all values of the term must match for the Comparison to match. If empty (not filled), then only one value of the term must match for the Comparison to match.
- **Match All Comparisons** (check box), when filled, **all** Comparisons much match for the result of the rule chain to be TRUE. If empty (not filled), then **any** Comparison must match for the result to be TRUE. In all cases, if no Comparisons match (and at least one Comparison has been defined), then the result will be FALSE.

Reactor Input and Output Connections

Using Pion’s graphical interface (the workspace), you can easily connect reactors using the "Connect Reactors" tool.

To delete a connection, you can either:

- Move your mouse over a connection between two reactors on the Reactors workspace. The connection will be highlighted in yellow. If you then left-click on the connector, a dialog appears, asking if you want to delete the connection.
- Access the reactor’s configuration dialog (double-click on the reactor), and delete the connection from the reactor’s "Connections" section. When two reactors are connected, the connection can be deleted from either end of the connection.

Every reactor has a Connections section which describes the connections to and from the reactor. Depending on the reactor:

- Input connections describe the connection by which streams of events **flow into** this reactor (how events are captured).
- Output connections describe the connection by which streams of events **flow from** this reactor after being captured or processed.

There may be zero or more (any number) of each of type of connection. In the case of a Filter Reactor for example, events flow into the reactor from Input Connections. After processing, events that have met the Comparison rules are delivered to the other reactors, as described in the Output Connections, while events that do not meet the rules are dropped.

Typically, collection reactors do not have input connections, unless they are feeds, and storage reactors do not typically have output connections, unless they are feeds to other reactors which allow events to simply "pass through" the reactor.

Field	**Description**
From	The name of the reactor from which this reactor gets data (events).
Connection ID	The identifier of the connection between this reactor and the reactor receiving the data.
To	The name of the reactor to which this reactor passes data (events).
Connection ID	The identifier of the connection between this reactor and the reactor which receives the data.

To delete a connection, click on the "x" at the end of the row describing that connection.

Collection Reactors

Collection reactors are responsible for getting data into Pion and therefore **every Pion workflow must start with at least one collection reactor**. Collection reactors acquire the data from its native format and translate it into normalized Pion events that can be understood and processed by any other reactor.

The two main Collection reactors are:

Log Input Reactor, which reads log files.

Sniffer Reactor, which captures/monitors network traffic.

These reactors are described in the following sections.

Log File Input Reactor

The Log File Input Reactor can capture data from any structured text file so long as that file’s format is defined in a “Codec.” A Pion Codec defines an input or output format for things like logs. A Codec can also define notation such as JSON or XML. Codecs are used to 'break' data into chunks that can be recognized or manipulated.

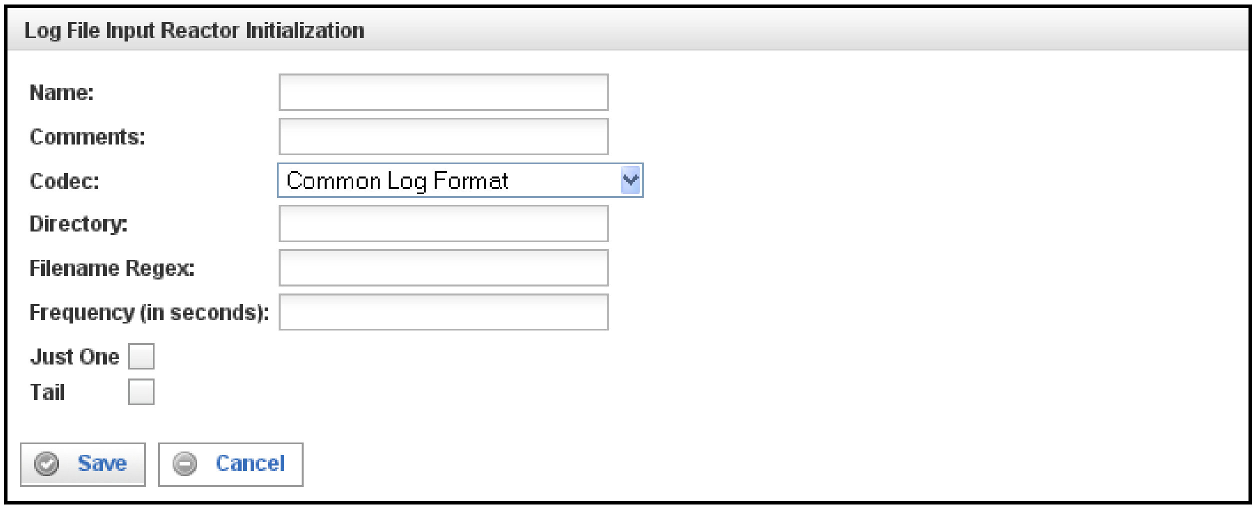
Pion understands all of the normal log formats associated with web server logs such as Extended Log Format, Combined Log Format, and Common Log Format out of the box as well as numerous other log formats. If Pion doesn’t already contain a Codec for the log format you need, you can easily create a new Codec that describes the log format.

Sometimes, log files are stored in compressed format. If Pion is trying to read a log file, and the name of the log file ends in one of the following file extensions, Pion recognizes that the file is compressed, and will decompress the file as a part of the read process.

.gz (gzip or GNU zip)

.bz2 (bzip2)

When you first add a the Log File Reactor you will be prompted for the following fields:


Log File Input Reactor Initialization

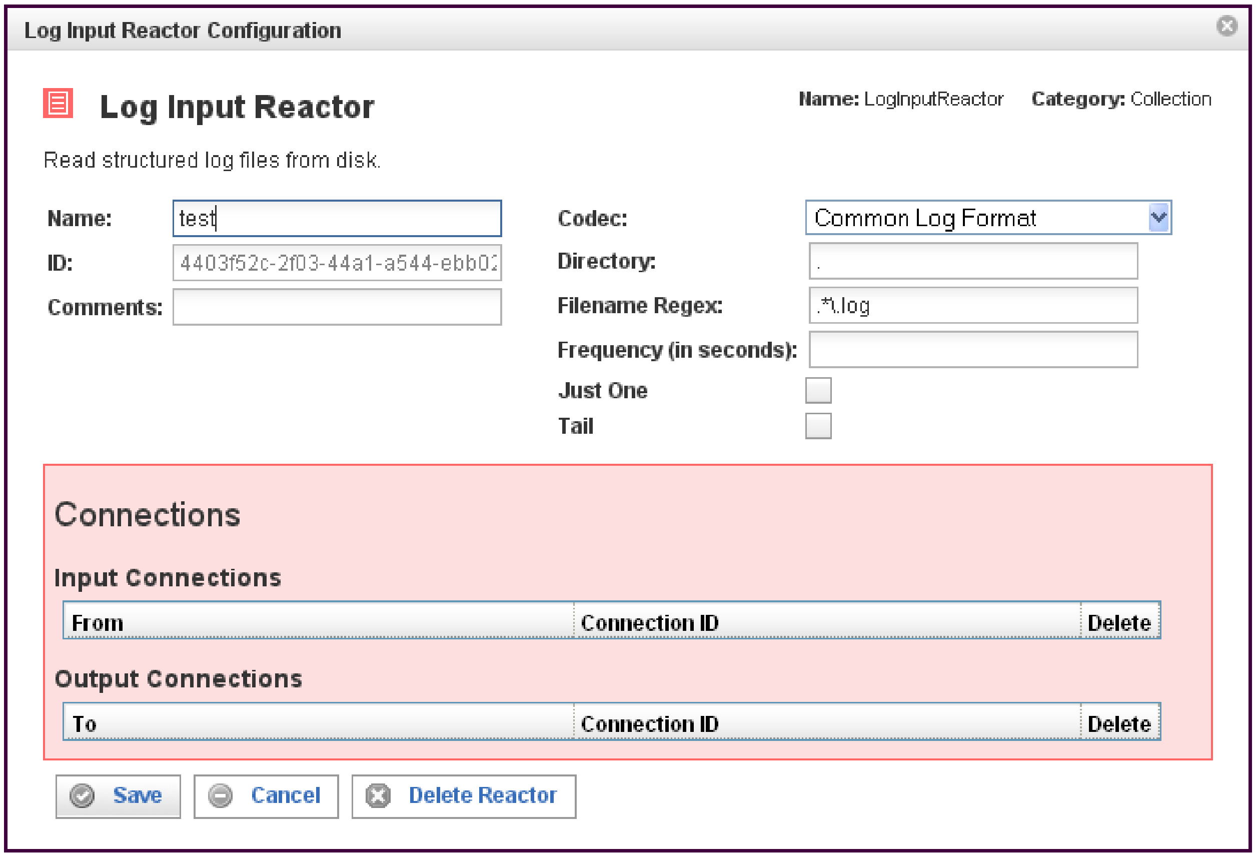
Name:	
Comments:	
Codec:	Common Log Format
Directory:	
Filename Regex:	
Frequency (in seconds):	
Just One	☐
Tail	☐

Field	**Description**
Name	Specify the name of this Log File Input Reactor. This can be any name you find descriptive, we recommend something which describes the log file(s) being read.
Comments	Enter some descriptive comments about this Log File Input Reactor. Any comment you feel will help you in the future, we

Field	Description
	recommend choosing something about the content or source of the log file(s), or the way in which the data in the files has been gathered or manipulated prior to being read.
Codec	Select the Codec to be used when reading the log file contents from this drop-down list. By default, the Log File Input reactor supports the following Codecs: • Common Log Format • Combined Log Format • Extended Log Format • Page View Log Format • Visitor Session Log Format • Webtrends (IIS) Log Format • Response Content Log • RSS Channel Extraction Codec • RSS Channel Log Format • RSS Item Extraction Codec • RSS Item Log Format • Atom Feed Extraction Codec • Atom Feed Log Format • Atom Entry Extraction Codec • Atom Entry Log Format • Stock Price Log • JSON Log Format • XML Log Format
Directory	Specify the complete path to the directory containing the log file(s) to be read.

Field	Description
Filename Regex	Enter a regular expression or a file name to be used to determine the log file(s) to be read. **Note**: A file name will be interpreted as a regular expression. If the file name contains a period, the period should be escaped.
Frequency	Define the amount of time (in seconds) for the interval between checking for new log files whose names match the Filename Regex.
Just One	Fill this check box to read only the first record of a log file. This converts the first record into an event, and repeatedly duplicates and outputs that event until the reactor is stopped. This is ONLY useful for performance benchmarking testing.
Tail	When this check box is filled, after reaching the end of a file, Pion keeps it open and periodically checks for newly appended data. (Based on the Unix '`tail -f`' command.)

This creates the Log File Input Reactor in the workspace. Double-clicking on the reactor brings you to the configuration screen:


Log Input Reactor Configuration

Log Input Reactor

Name: LogInputReactor Category: Collection

Read structured log files from disk.

Name:	test	Codec:	Common Log Format
ID:	4403f52c-2f03-44a1-a544-ebb02	Directory:	.
Comments:		Filename Regex:	.*\.log
		Frequency (in seconds):	
		Just One	☐
		Tail	☐

Connections

Input Connections

From	Connection ID	Delete

Output Connections

To	Connection ID	Delete

In addition to the fields you saw when adding the reactor initially (described above), you will see one additional field and the *Connections* section. The additional field is:

Field	**Description**
ID	The reactor ID, a unique ID generated by the Pion server. This ID identifies a particular reactor.

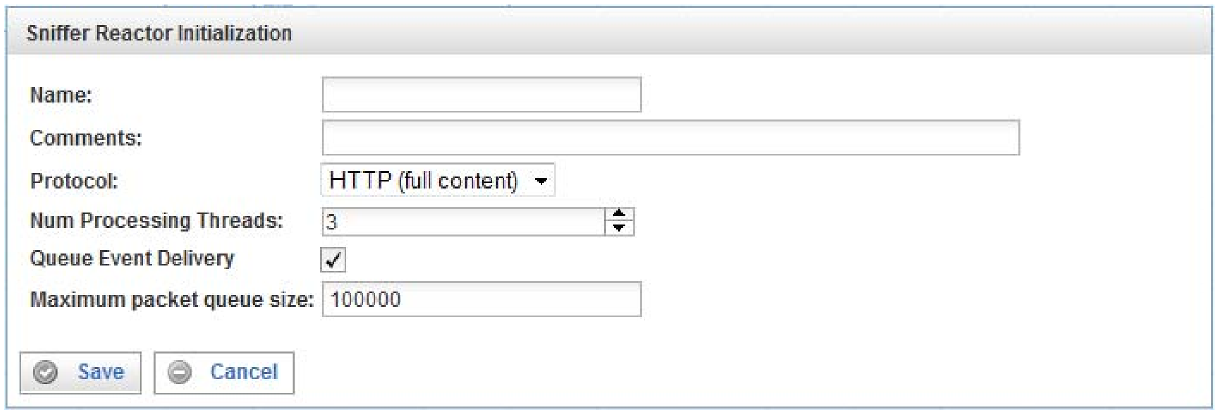
The Connections Section

The Connections section describes input and output connections for this Log Input Reactor. See “Reactor Input and Output Connections“ on page 6 for more information.

Sniffer Reactor

The Sniffer Reactor captures and decodes traffic from network interfaces. This can be live traffic traveling through a network in real time or previously captured traffic stored in a standard PCAP format. The Sniffer Reactor understands any network protocol for which you have installed a Pion plug-in (for example plug-ins are available for the HTTP, HTTPS, and SMTP protocols). With the proper plug in, Pion can see and understand traffic on the network, such as traffic between web servers and clients, and can generate events based on that traffic.

When you first add a the Sniffer Reactor you will be prompted for the following fields:


Sniffer Reactor Initialization

Name:	
Comments:	
Protocol:	HTTP (full content)
Num Processing Threads:	3
Queue Event Delivery	✓
Maximum packet queue size:	100000

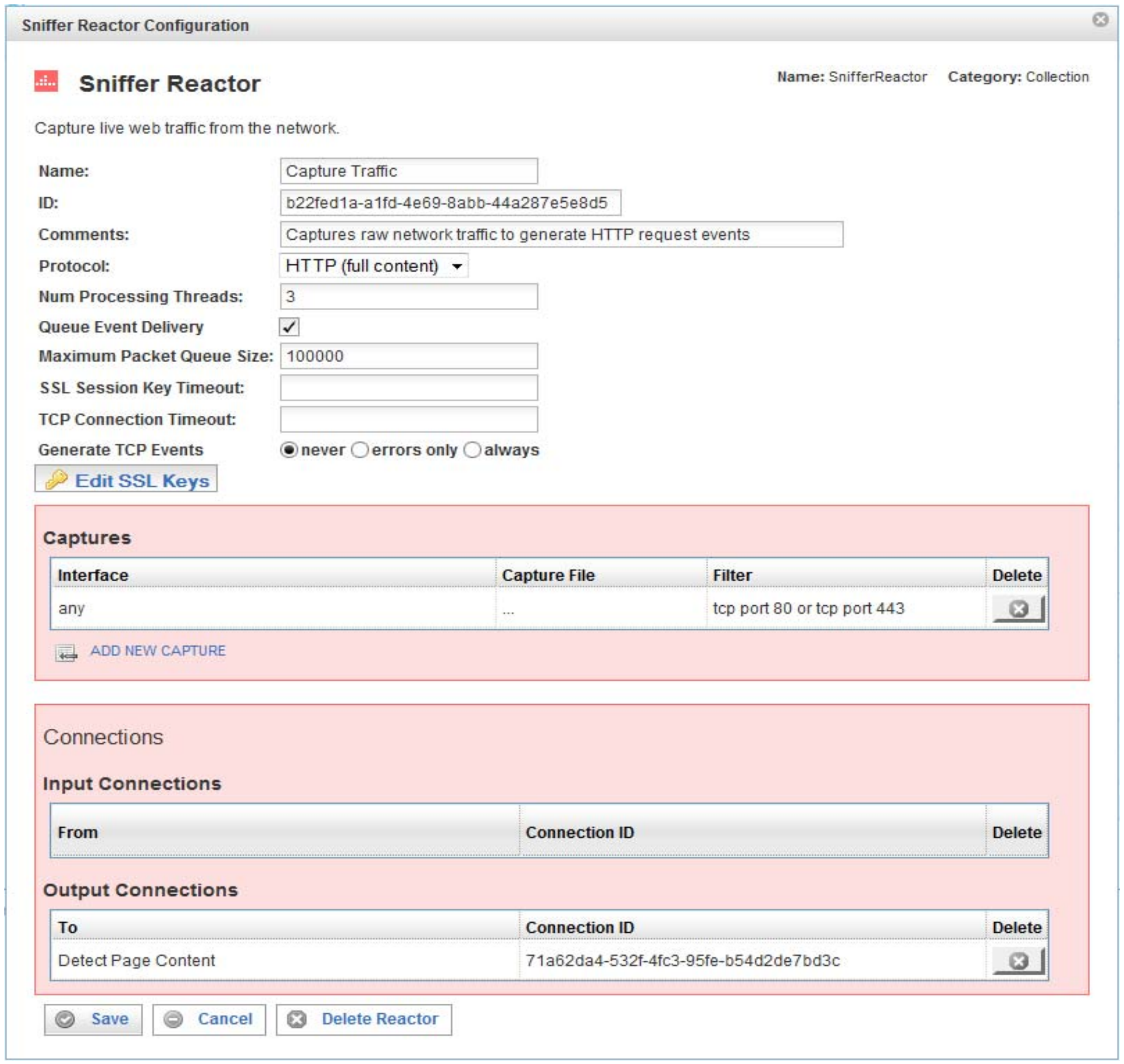
Field	**Description**
Name	Specify the name of this Sniffer Reactor. This can be any name you find descriptive, we recommend something which describes the network segment to which this Sniffer Reactor is listening.
Comments	Enter some descriptive comments about this Sniffer Reactor. Any comment you feel will help you in the future, we recommend choosing something that includes the contact person for the network segment and any associated work code that may be needed in the future.
Protocol	From this drop down list, select the protocol you want the Sniffer Reactor to use when listening to traffic. By default Pion supports the following protocols:

Field	Description
	• *HTTP (no content)* captures the critical fields needed by web analytics tools. These fields include those pertaining to URL, host, referrer, page title, user agent, cookie and query. • *HTTP (full content)* captures the critical fields needed by web analytics tools, and also captures the page content and the full client request. Page content is saved into a special field in the clickstream term. Both server to client and client to server traffic can be captured. • For server to client traffic, page content (from the server request and the client response) is saved into a special field in the clickstream term called *sc-content*. The maximum amount of data in sc-content can be defined in the Protocol configuration page, and the default is 512k for *sc-content*. Data in excess of the configured maximum is truncated. • For client to server traffic, page content (the client request and the server response) is saved into a special field in the clickstream term called *cs-content*. The maximum amount of data in sc-content can be defined in the Protocol configuration page. To change the maximum amount of data that is help for processing, go to the Protocol page, select HTTP (full content) and set the Max Request content length and the Max Response content length appropriately. Note that capturing using the *HTTP (full content)* protocol requires more memory than capturing using the *HTTP (no content)* protocol. More memory is required because more data must be captured and manipulated, and because the amount of data making up page content is often much more than any other field. • *SMTP (Email)* captures the critical fields from SMTP-

Field	Description
	based emails, allowing you to generate events based on emails. All standard SMTP header fields and the message body are captured. The maximum amount of data stored from the message body can be defined in the Protocol configuration (the default is 1MB). The default is *HTTP(full content)* because it contains all of the information, if you are limited in memory or don't need page content then it can be easily switched to improve performance.
Num Processing Threads (Number of Processing Threads)	Specify the number of processing threads a Sniffer reactor uses. It is useful for provisioning resources on the Pion server or in a shared environment. Adding more available processing threads can improve the performance of Pion significantly on machines with multiple processing cores. **Note**: Generally you should not use more threads than you have CPUs/Cores in the Pion server.
Queue Event Delivery	Fill this check box to cause events to be delivered to other reactors by threads not affiliated with this reactor (other threads in Pion). Enabling this feature distributes workload and adds an additional memory-based work queue, which may reduce overall performance. If this check box is empty, the feature is disabled and the processing threads of this reactor also handle the delivery of events to other reactors. Disabling this feature may improve overall performance because there will be less context switching.

Field	Description
Maximum packet queue size	Specify the maximum number of network packets that can be in the event queue per processing thread. We recommend that the total (this value * number of processing threads) be left at 100,000 (the default) as a reasonable start. Note that the queue size setting has a direct impact on possible memory consumption, and that it also determines how much you can buffer in a burst. You must set the queue size to balance between these considerations according to your needs and resources.

Once the Sniffer Reactor has been added to the workspace it must be configured to listen to network traffic. By double clicking on the reactor you will open the following configuration screen:



Sniffer Reactor

Name: SnifferReactor Category: Collection

Capture live web traffic from the network.

Name:	Capture Traffic
ID:	b22fed1a-a1fd-4e69-8abb-44a287e5e8d5
Comments:	Captures raw network traffic to generate HTTP request events
Protocol:	HTTP (full content)
Num Processing Threads:	3
Queue Event Delivery	✓
Maximum Packet Queue Size:	100000
SSL Session Key Timeout:	
TCP Connection Timeout:	
Generate TCP Events	[✓ never] errors only always

Captures

Interface	Capture File	Filter	Delete
any	...	tcp port 80 or tcp port 443	

Connections

Input Connections

From	Connection ID	Delete

Output Connections

To	Connection ID	Delete
Detect Page Content	71a62da4-532f-4fc3-95fe-b54d2de7bd3c	

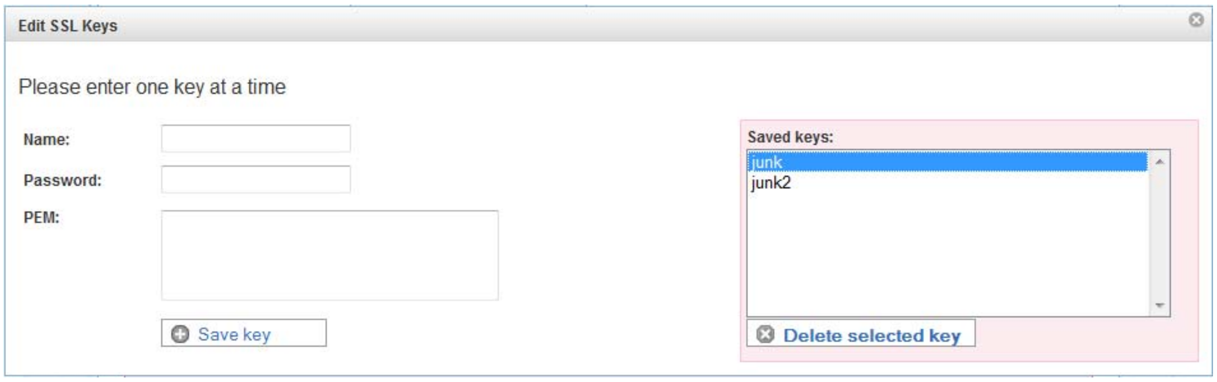
In addition to the fields you saw when adding the reactor initially you will see an *Edit SSL Keys* button, and sections for *Captures,* and *Connections.*

Edit SSL Keys

Clicking on the Edit SSL Keys button opens the Edit SSL Keys configuration screen, which allows you to save SSL key information, which then enables Pion to use those keys, Once Pion has the proper key, it can then decrypt the SSL encrypted data in real-time. The SSL keys are stored in a KeyStore vault (an encrypted database), and if a password is used to protect the keys, it must be entered at startup. You enter the password either in the SSL Key editor of the Sniffer Reactor configuration dialog (above), or the 'Setup SSL Keys' page of the installation wizard.

SSL keys are required only for sites that use SSL encrypted communications. Typically, SSL encrypted communications are used only by sites such as those common in retail, defense, and financial applications.

Note: Pion uses SSL keys that are in the PEM (Privacy Enhanced Mail) format, which is base 64 and the same format as the BER format, except that the PEM format uses header and footer lines to indicate the beginning and the end of the key.


Edit SSL Keys

Please enter one key at a time

Name:

Password:

PEM:

Saved keys:

junk

junk2

To specify an SSL key, specify the following:

Field	**Description**
Name	The name of the SSL key. The key name should be a short description, and the name can contain any characters. There is no restriction on the length of a key name.
Password	This is the password (or pass phrase) for this SSL key.

Field	Description
PEM	This field is where you enter the encryption key data. To avoid mistakes, it is best to cut and paste the key data using a text editor. The following format can be used for a PEM key: -----BEGIN CERTIFICATE----- Your certificate here. -----END CERTIFICATE----- -----BEGIN RSA PRIVATE KEY----- Your private key here. -----END RSA PRIVATE KEY----- Note that, in the example above, the certificate portion is optional, and is ignored by Pion. Only the key portion is actually used. **Note:** There are no blank lines between the beginning and the end of the certificate or the key.
Saved keys	This field lists previously saved SSL keys. To save a key, fill in the name, password, and PEM fields (above), and click Save key. To delete a key, select the key you want to delete, and click Delete selected key.

The Captures Section

This section specifies where your Sniffer Reactor will be getting data from. Each line represents a data source and a filter to be applied to that data source. There are two types of supported data sources; *Interfaces* and *Capture Files.*

Field	Description
Interface	Specifies the network interface to use when listening to traffic.

Field	Description
	Select the cell in the *Interface* column to display a drop down list of the network interface cards on the Pion server. You can then choose the interface that is receiving the live traffic you want to monitor. The notation used to identify an interface is dependent on the operating system (Linux, Mac, or Windows). Interfaces may include: • On Linux, Ethernet interfaces are usually eth0/eth1/eth*X* and lo for loopback. • On Mac, Ethernet interfaces are usually en0/en1/en*X* and lo0 for loopback. • wlan0 (or similar, for wireless interfaces) • Other names that do not fit one of the patterns described above. In Pion, interface names are arbitrary, and you can name them anything you like. **Note**: If you are running VMware, there will be many other virtual interfaces, between the VM and the host, and **running Pion in a VMware instance is not recommended**, because there may be too many lost packets. To see the devices, their IP addresses, and other interface information: • On Windows operating systems, run the command "`ipconfig /all`". • On Unix-based operating systems, log in as root, and run the command "`ifconfig -a`".
Capture File	Specifies a file that contains saved traffic. The content of this file is used instead of live traffic from an interface. A capture file is read once, then the reactor stops. To reread the capture file, restart the Sniffer reactor. If the traffic has already been captured in a file, simply type the full file path in the *Capture File* column (note: the file must be a standard .cap or .pcap file).

Field	Description
Filter	Defines the filters to apply to the traffic received on the interface specified above. After selecting your traffic sources, you can filter out traffic which Pion doesn't need to see. Although traffic can be filtered out later using other reactors, doing it at this stage (as a function of the Sniffer Reactor) is the most efficient way to remove unwanted information. The filter reactor uses the same syntax as the tcpdump utility. For example: • To filter out HTTP traffic, you would use "tcp port 80" as your filter string to remove all non HTTP traffic. • To filter out HTTPS traffic, you would use "tcp port 443" as your filter string to remove all non HTTPS traffic. Complete tcpdump syntax can be found at http://www.tcpdump.org/tcpdump_man.html.

Click on the *ADD NEW* CAPTURE icon to add a new line to the table.

To delete a capture, click on the "x" at the end of the row describing that capture.

The Connections Section

The Connections section describes input and output connections for this Sniffer Reactor. See Reactor Input and Output Connections on page 6 for more information.

Processing Reactors

After Pion has collected data it must do something with it in order to make it more useful. The Processing reactors are responsible for manipulating and enriching the raw data to create the desired information.

The Processing reactors include:

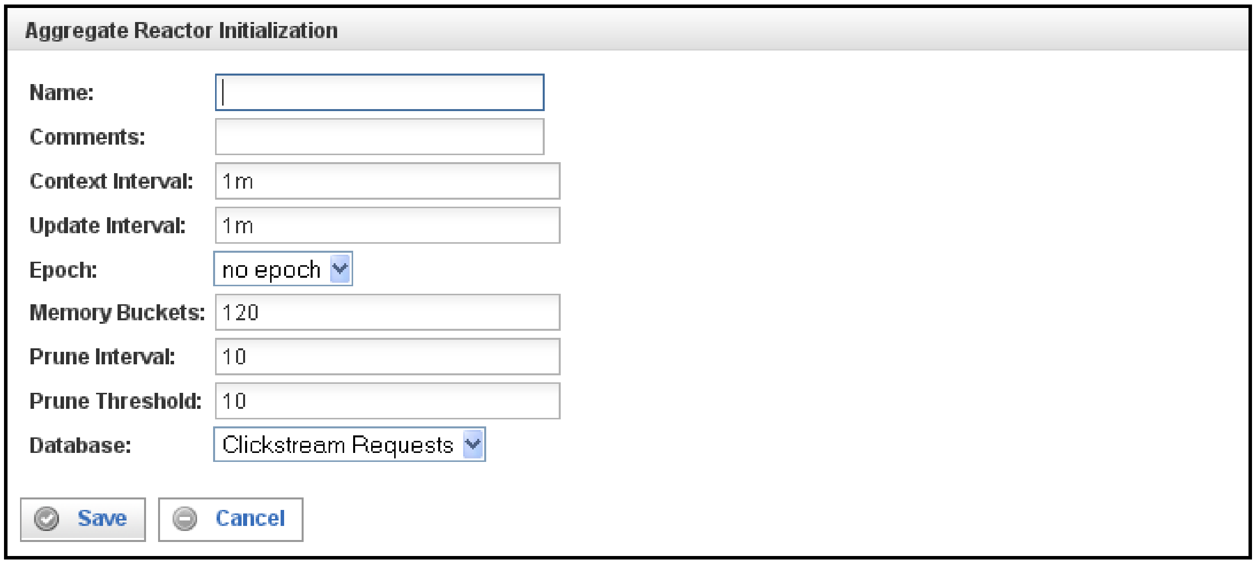
- **Aggregate Reactor**, which is able to take large volumes of raw data and distill critical pieces of information out of them.
- **Clickstream Reactor**, which is able take raw web traffic and organize it into logical pages and individual user sessions.
- **Content Hash Reactor**, which creates and then stores a hash identifier for a specified content term in an HTTP response. These identifiers are used by the Replay feature to eliminate duplicate responses and determine which ones to store into databases.
- **Filter Reactor**, which filters information based on any number of criteria you might set.
- **Fission Reactor**, which splits input events into sequences of output events.
- **Python Reactor**, which allows you to use Python to collect, manipulate and store events.
- **Script Reactor**, which enables you to leverage existing command line scripts to quickly add capabilities to Pion.
- **Session Filter Reactor**, which enables you hold entire sessions until they meet certain criteria.
- **SQL Reactor**, which lets Pion interact with any relational database based on real time information.
- **Transform Reactor**, which is able to do deep manipulation of the data in the data stream.

These reactors are described in the following sections.

Aggregate Reactor

The Aggregate Reactor can distill critical pieces of information from large volumes of raw data. For example, the Aggregate Reactor could count all of the web users on the site in the last hour, calculate the average page size downloaded and tell you the total amount of traffic the site consumed in the last day. The Aggregate Reactor could even tell you each unique user name that accessed the site. Because it performs these calculations inside Pion, the Aggregate Reactor can do real-time reports, which reduces stress on your backend networks and systems, which are critical for high volume applications.

When you add an Aggregate Reactor you will be prompted for the following fields:


Aggregate Reactor Initialization

Name:	
Comments:	
Context Interval:	1m
Update Interval:	1m
Epoch:	no epoch
Memory Buckets:	120
Prune Interval:	10
Prune Threshold:	10
Database:	Clickstream Requests

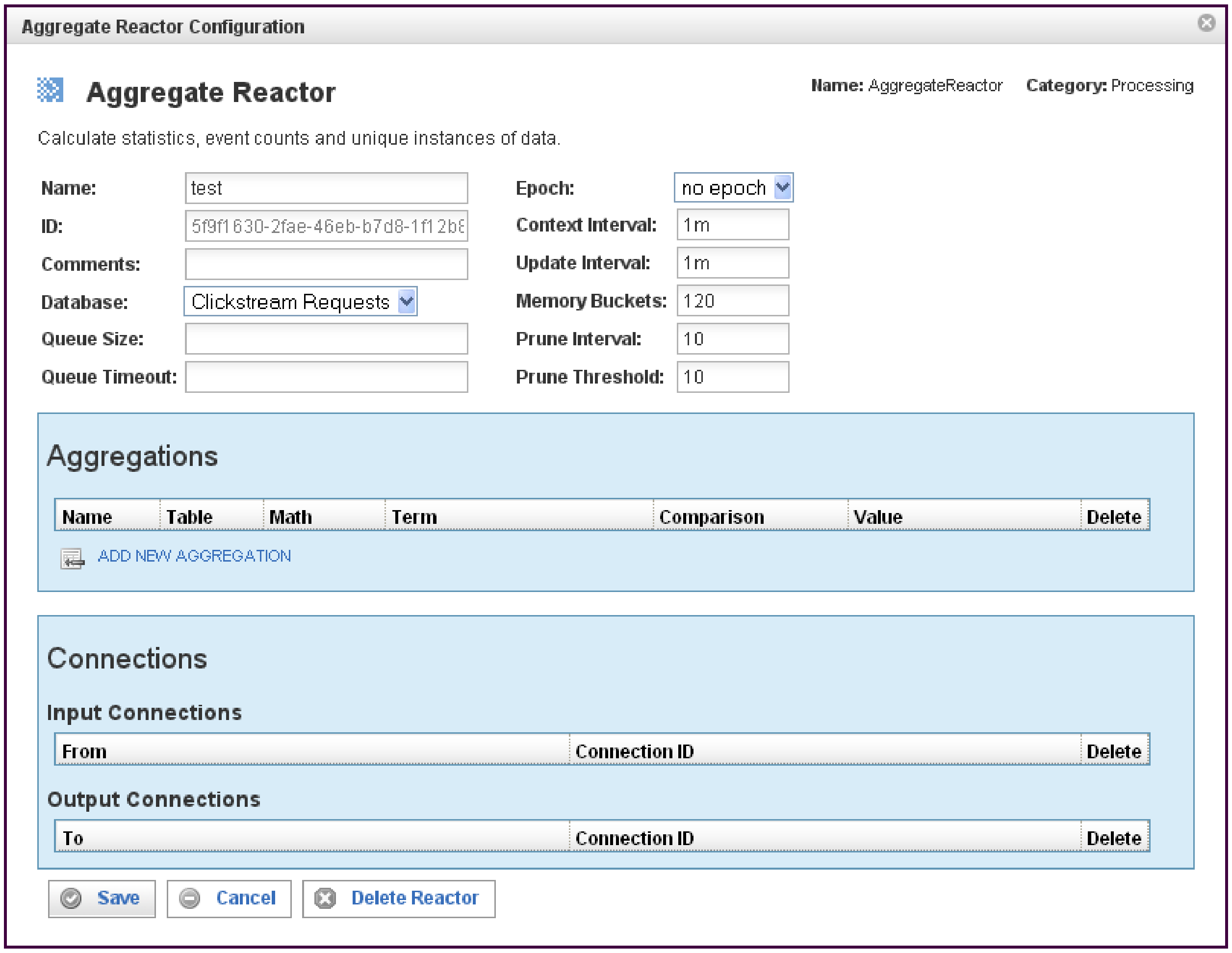
Field	**Description**
Name	Specify the name of this Aggregate Reactor. This can be any name you find descriptive, we recommend something which describes the data this Aggregate Reactor is gathering. The name may contain any combination of letters and numbers.

Field	Description
Comments	Enter some descriptive comments about this Aggregate Reactor. Any comment you feel will help you in the future, we recommend choosing something that describes the data being gathered and/or the data being ignored.
Context Interval	Specifies how often the Memory Bucket (the context) is changed. Use a numeric value, followed by a time unit, as indicated by one of the following letters: • **S** for Seconds (the default) • **M** for Minute • **H** for Hour • **D** for Day • **W** for Week • **Y** for Year (365 days) Once Context changes: An event of type aggregate#stat-event is sent with all values. For Unique, an event for *each* unique string is sent. A database row is added (if a database is enabled). A new "tick" will be present in graphs & queryService XML (for information on using the Query Service, see “Using the Query Service with the Aggregate Reactor” on page 30).
Update Interval	Indicates how often updates to the current Memory Bucket (update events, #update-event) are sent out. The Update Interval must be smaller than the Context Interval and is usable only if your Context Interval is very large, and you are processing update events. For the time unit notation used to specify the duration of the Update Interval, see “Context Interval” (above).

Field	**Description**
Epoch	Defines the unit of the starting point for the Memory Bucket. For example, an epoch of a “day” means that all buckets are based on the change of the day (when a new day begins, or hour “0”). If Epoch is not defined, first bucket starts at the program start-up time. Use the drop-down list to select the Epoch.
Memory Buckets	Specifies how many Context Intervals are kept in memory. This number can be quite large, 1MB of memory can hold ~ 20,000 - 30,000 buckets. Commonly referred as a “Bucket” or “Bin,” a Memory Bucket is a time-bound container for elements. The length of time the Memory Bucket covers (spans) is determined by the Context Interval. Total time interval is (Context Interval x Memory Buckets). When counting events, a Memory Bucket of 1 minute is used to count all events that occur within a minute. Using 120 Memory Buckets that have a Context Interval of 1-minute (120 1-minute buckets), you can graph event frequency for a two hour span. To make fine-grain graphs, use a large Memory Bucket value, a short Context Interval, and use Grouping in the query.
Prune Interval	Specifies how often the unique values are pruned. The Prune Interval is expressed as a multiple of the Context Interval (for example 10 x 1m = 10m).

Field	Description
Prune Threshold	Specifies the minimum number of times a unique value must be found in order to be counted. If the minimum number is not met, instances of that value are purged and no longer tracked. The Prune Threshold is intended to prune out (eliminate) values that are aberrations, and which would otherwise indefinitely consume an Aggregate Reactor, without providing meaningful amounts of data. For example, assume you are looking for a particular value, say page title. If there is a page title where the value is "blue moon" and a prune threshold of 10, then if there are less than 10 counted instances of "blue moon" then this value gets dropped, and is no longer tracked (until another instance).
Database	If defined; identifies a database (that must be one of the supported database types SQLite, MySQL, Oracle, etc.) in which statistical events will be stored. The database name (specified in the reactor configuration) defines where tables will be created, and all the statistical events will be stored in this database table. Note that the database table name matches the name of the aggregate, so aggregate names are restricted by same rules as database table names. For example, if you want to have Aggregate Reactor save its statistics on a continuous basis into a database table, then define a database, and it will happen. If you do not define a database, you will still get: Statistical events produced (you can use them for any normal processing, a log output reactor, or such). The graphs (click on the graph button on aggregate). The statistics in XML format. To see statistics in XML format, use SOAP style URI (see bottom of the graph dialog for the URI). Feed output from Aggregate Reactor.

This creates the Aggregate Reactor in the Workspace. Double-clicking the reactor brings you to the Aggregate Reactor Configuration screen:



Aggregate Reactor Configuration

Aggregate Reactor

Name: AggregateReactor **Category:** Processing

Calculate statistics, event counts and unique instances of data.

Name:	test	Epoch:	no epoch
ID:	5f9f1630-2fae-46eb-b7d8-1f12b8	Context Interval:	1m
Comments:		Update Interval:	1m
Database:	Clickstream Requests	Memory Buckets:	120
Queue Size:		Prune Interval:	10
Queue Timeout:		Prune Threshold:	10

Aggregations

Name	Table	Math	Term	Comparison	Value	Delete

ADD NEW AGGREGATION

Connections

Input Connections

From	Connection ID	Delete

Output Connections

To	Connection ID	Delete

In addition to the fields you saw when adding the reactor initially (described above), you will see some additional fields and sections for *Aggregations* and *Connections*. The additional fields are:

Field	**Description**
ID	The reactor ID, a unique ID generated by the Pion server. This ID identifies a particular reactor.

Field	Description
Queue Size	The number of events allowed in the queue before triggering a flush of the queue. When the queue is flushed, the queued events are inserted (bulk-inserted) into the database specified in the Database field. The default queue size is 1000 events.
Queue Timeout	The queue timeout determines how long to wait before forcing a queue flush (a bulk-insert of queue contents to the database into the database specified in the Database field. The default queue timeout is 10s (10 seconds).

The Aggregations Section

The Aggregations section specifies the data that will be processed from the database selected above. Click on the ADD NEW AGGREGATION icon to add a new line to the table. Each line represents a data item to be processed and the processing operations for that data item.

Field	Description
Name	Identifies the aggregation. This can be any name you find descriptive, we recommend something which describes this aggregation. The name may contain any combination of letters and numbers, and my contain underscores, but may not start with a number or an underscore.
Table	Specifies the name of the database table in which to store the Context records. The table name must be unique (within the specified database), and is not case sensitive. The name must be made up of any combination of letters and numbers.
Math	Specifies a mathematical operation to be applied to the value of the data item: **Count** applies to any term, and it causes Pion to count the total number of values (unique or repeated). If the term exists in an event, it is counted. Count returns a number ***n*** (the number of events containing the term) and a time (the amount of time in which the events occurred). Count may be

Field	Description
	used to graph how many events occurred in a specific time period. Count is a good way to count things like page hits, incidents, and frequency. **Sum** applies only to numeric terms, for example clickstream#bytes, and it enables Pion to compute any or all of the following:: 1. **SUM** (add the values together). 2. **MIN** (find the lowest value). 3. **MAX** (find the highest value). 4. **AVG** (calculate the average value). 5. **N** (counts the number of values). Sum can also compare the data values to a comparison term, such as greater-than and the value “0.” Sum is good for presenting traffic, volume, max load, and changes in average. **Unique** applies only to string terms/fields, for example clickstream#request, and it causes Pion to count the total number of unique (non-repeated) values: 6. Any unique value creates a new counter (also known as a “slot" or an "accumulator"). 7. Once a counter is created, the number of unique values is counted. 8. Upon a pruning interval, if the count is less than the pruning value, the accumulator is discarded. (See the description of prune threshold above.) Unique is good for presenting things like a spread of status messages and the ratio of pages viewed. Unique is most useful when combined with a Regular Expression, for example "GET (.+)&" and, when Unique finds a new unique string, it starts a new unique count for that string.

Field	**Description**
	Use Pruning to take out edge condition matches.
Term	Defines the value of the term to be used for the comparison logic. Specifies the data item to find. You can select a term (an identified data item) from an existing vocabulary, or you can add a new vocabulary and define the term(s) to be found. One event consists of multiple Terms. For example, a stock event consists of terms "symbol" and "price." Note that one term may contain multiple values.
Comparison	Specifies the type of comparison to test, such as true/false or is-defined/is-not-defined. (Note, do not use TRUE/FALSE in an Aggregate Reactor). Depending on the term that is specified in a row, an appropriate list of comparisons will appear in the comparison cell of that row. For example, if it's a date type, you'll get choices like "earlier than", whereas if it's a string type, you'll get choices like "starts with". Like most comparisons, these two examples require a value to compare against, but a few comparisons, like “is defined,” do not require a value to compare against. For more information on comparisons, see "Comparisons“ on page 5.
Value	Defines the data value to test against (or the regex to apply to the data values).

To delete a data item, click on the “x” in the delete column for the line describing that data item.

The Connections Section

The Connections section describes input and output connections for this Aggregate Reactor. See “Reactor Input and Output Connections“ on page 6 for more information.

Using the Query Service with the Aggregate Reactor

Aggregate Reactor can be queried using the Query Service (QueryService). To use the Query Service, construct the query and enter it as a URL In your browser. For example:

`http://`***`host:port`***`/query/reactors/`***`reactor-id`***`/`***`aggname`***`?VEC=`***`1/dt/x`***`,`***`1/n/y`***`&GRP=`***`grp`***

Where:

- `host` is the host name or IP of the Pion server.
- `port` is the port number of your Pion installation.
- `reactor-id` is the ID of this Aggregate Reactor. (To obtain the reactor ID, right click on the Aggregate Reactor and select “Show XML”.)
- `aggname` is the name of the aggregate.
- `VEC=1/dt/x,1/n/y` is the list of vectors.
- `grp` is the number of values to group into a single data point.

Query results are returned in the form of a graph.

Query Service Parameters

All Query Service parameters are optional. When specified, Query Service parameters are in the format `PARAMETER=VALUE`, and parameter/value combinations are separated by the ampersand character (&).

The Query Service parameters are as follows:

- VEC (list of vectors)

  The list of vectors is separated by commas. Vectors are specified using the format:

  `aggname?VEC=1/dt/x,1/n/y`

  Where:

  - `aggregate` is the name of the aggregate.
  - `VEC=` indicates the beginning of the list of vectors.
  - `dt` is the DateTime
  - `x` indicates to place this vector on the X axis of the graph.
  - `y` indicates to place this vector on the Y axis of the graph.

List as many vectors as necessary, separating vectors in the list with commas, for example:
`aggname1/dt/x`, `aggname2/dt/x`, `aggname3/dt/x`, `aggname4/dt/x`, `aggname5/dt/x`, `aggname6/dt/x`, `aggname7/dt/x[,...]`

- PTS (points to display)

  Points to display indicates how many data points to display on the graph. Specified as `PTS=`***`n`***, where ***`n`*** is the number of points to display. The total time frame of the graph is calculate as follows:: `((PTS * Context Interval) / GRP).`

  If PTS is greater than (Memory Buckets / GRP), the graph will be truncated.

- GRP (grouping)

  Grouping specifies how many samples are grouped.

  - 1 = No grouping
  - 2 = Groups two adjacent samples together
  - 3 = Groups three adjacent samples together

  For example, to group 10 minute samples into a data point covering one hour, use a value of 6.

- ENC (Encoding)

  Controls the encoding to be used. Specified as `ENC=`***`enc`***, where ***`enc`*** is one of the following:

  - XML (Full XML)
  - XML-S (Shorthand notation XML)
  - XML-C (Compressed XML)

  Note: The data sets can be enormous, so if your application can read it, Pion recommends that you use compressed or shorthand XML.

- DAT (Date String format)

  - 0 = Time expressed as milliseconds since 1970-01-01
  - 1 = Time expressed as seconds since 1970-01-01
  - 2 = Time and Date in format: "01 January 2008 17:25:01"

  - 3 = Time and Date in ISO format "20080101T172501"
- RND (Random data)

  Returns random data, for testing purposes.
- MVY (MultiVectorY)
  - 0 = Split each vector to a separate data set in XML.
  - 1 = Combined multiple Y vectors into one data set in XML.

Clickstream Reactor

The Clickstream Reactor is able take raw web traffic and organize it into logical pages and individual user sessions. It does this in order to identify users and overall session statistics such as visit length, pages viewed and more. The Clickstream reactor accomplishes this by understanding how browsers normally communicate with backend systems and how session state is maintained. Often the Clickstream Reactor works properly without changing any configuration options. The Clickstream reactor contains the ability to handle very complex and custom sessionization parameters, which may be employed by large applications to maintain state across globally distributed application instances.

The Clickstream Reactor functions as follows:

1. Page detection is controlled by the page object detection.
2. Requests are assigned to pages using timestamps and referrer headers -- the algorithm supports having several pages open at a time, up to the specified "maximum pages per session."
3. Requests grouped within a page are ordered using timestamps (request-num).
4. Pages are closed when either:
   - No requests are associated with the page for a period longer than time specified by the "page timeout" parameter.
   - The maximum number of pages per session is exceeded (oldest pages are closed first).
5. All requests associated with a particular page will have the same session-id and page-num values in their http-event.
6. When a page closes, a "page event" is generated that represents all the requests associated with a particular page.

   Page events (type "urn:vocab:clickstream#page-event") contain the following clickstream vocabulary fields:

   - date, time, date-time, epoch-time = timestamp for first http event associated with the page
   - page-num = sequence number for the page view, starting with 1
   - robot = 1 if the session is considered to be a "robot"
   - bytes = total bytes for all requests and responses
   - cp-rtt = the average round trip time in microseconds from the client to the Pion server

- cp-rtt-packets = the number of round trip time measurements from the client to the Pion server
- cp-rtt-sum = the sum of all round trip time measurements from the client to the Pion server
- cs-bytes = total bytes for all requests
- sc-bytes = total bytes for all responses
- page-title = title extracted from HTML content
- page-load = total time to load the page, from start of first request to end of last response
- page-load-redirect = total time for all redirected (302) page requests
- page-load-base = time-taken for the base HTML file
- page-hits = total number of HTTP requests associated with the page
- page-load-content = total time for all page object (not base HTML or redirected) requests
- page-dwell = total time from the last response to the first request for the next page view
- ps-rtt = the average round trip time in microseconds from the Pion server to the protocol server
- ps-rtt-packets = the number of round trip time measurements from the Pion server to the protocol server
- ps-rtt-sum = the sum of all round trip time measurements from the Pion server to the protocol server
- referer = copied from first http event associated with the page
- time-taken = the sum of microseconds that it took to complete all HTTP events associated with the page from the end user's perspective

The following fields are all copied from base HTML file request: session-id, session-group, uri-stem, uri-query, host, user-agent, client-ip, server-ip, status, uri, request, server-port, content-id, page-title, and content-type.

Page events may contain "sticky page fields" which are copied from http-events.

7. Requests are assigned to sessions using the cookies configured in the Clickstream Reactor, and if no cookies are found, a combination of the user agent and the IP address are used for matching.

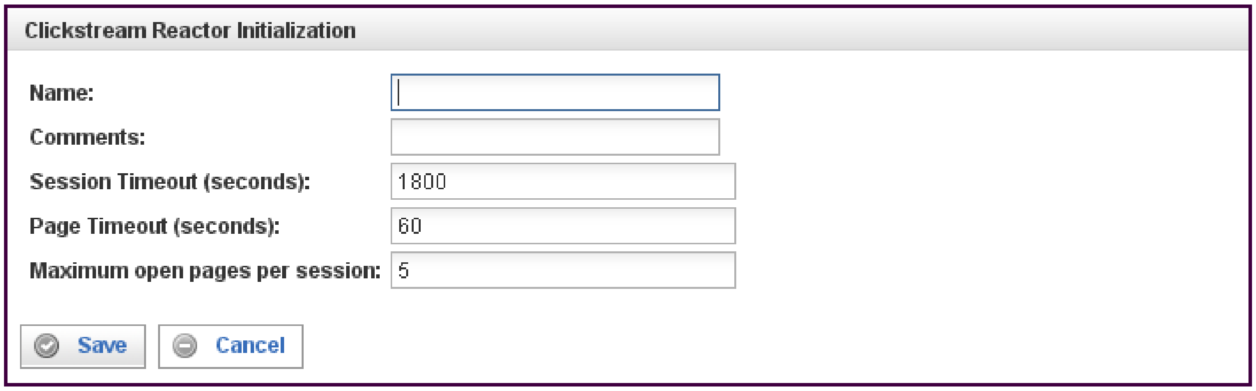
8. When a session closes, a "session event" is generated that represents all the requests associated with a particular visitor session.

   Session events (type "urn:vocab:clickstream#session-event") contain the following clickstream vocabulary fields:

   - date, time, date-time, epoch-time = timestamp for first http event associated with the session
   - session-id, session-group, date, time, date-time, referer
   - session-pages = number of pages in the session
   - session-requests = number of http requests in the session
   - session-dwell = total dwell time for all page views
   - session-length = total length of the session
   - robot = 1 if the session is considered to be a "robot"
   - bytes = total bytes for all requests and responses
   - cs-bytes = total bytes for all requests
   - sc-bytes = total bytes for all responses
   - cookie-id = sessionizing cookies detected for the session

   Session events may also contain "sticky fields." When found in a session event, the sticky field values are copied.

When you add a Clickstream Reactor you will be prompted for the following fields:

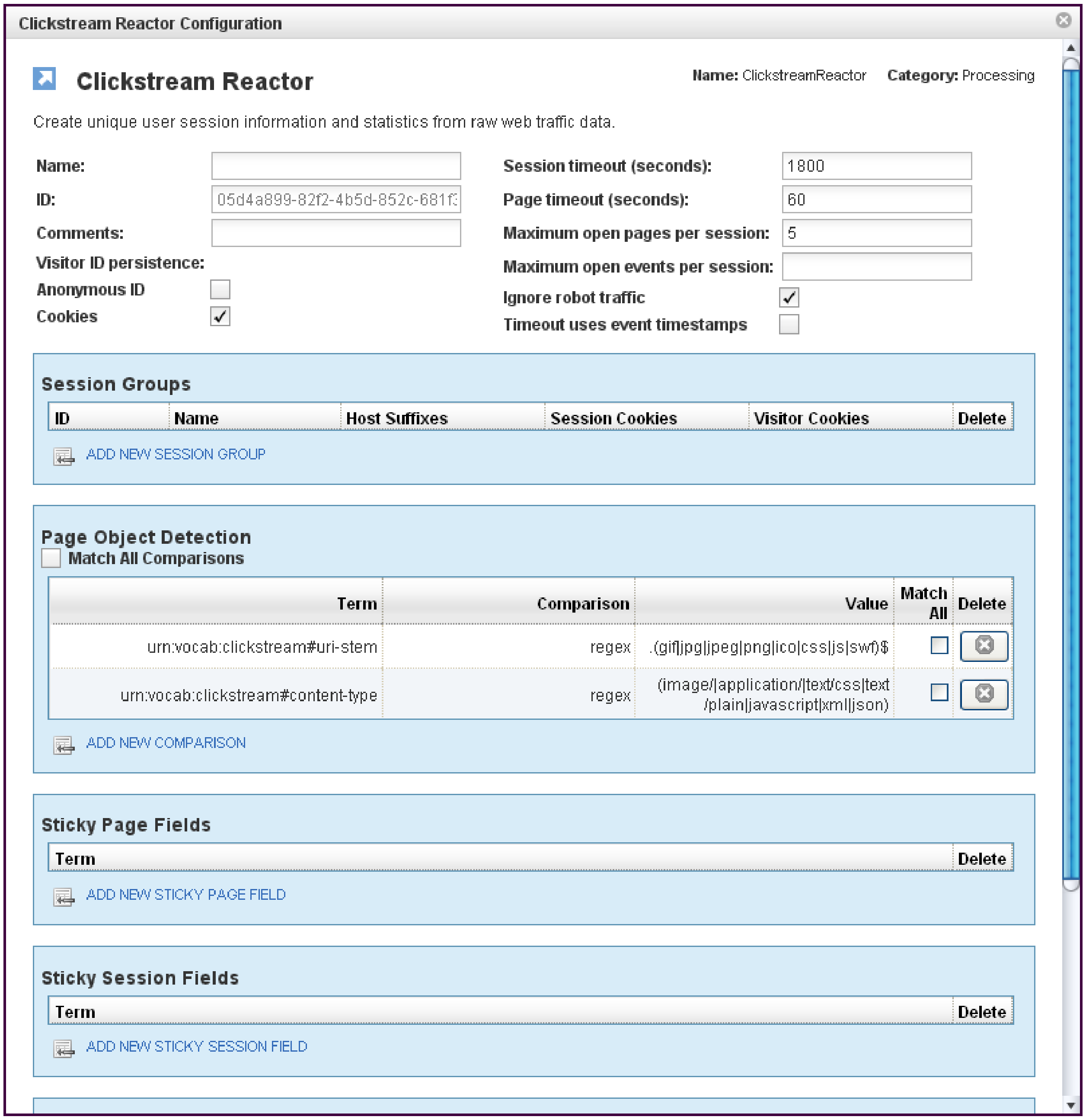


Clickstream Reactor Initialization

Name:	
Comments:	
Session Timeout (seconds):	1800
Page Timeout (seconds):	60
Maximum open pages per session:	5

Field	Description
Name	Specify the name of this Clickstream Reactor. This can be any name you find descriptive, we recommend something which describes the session data this Clickstream Reactor is producing. The name may be any combination of letters and numbers.
Comments	Enter some descriptive comments about this Clickstream Reactor. Any comment you feel will help you in the future, we recommend choosing something that describes the data being produced and/or the sessions being analyzed.
Session Timeout	Amount of time (in seconds) to wait before a session is considered to have timed-out. For example, if a user/browser is browsing a web site, the Clickstream Reactor will consider the user having abandoned/quit browsing the site after the specified number of seconds. The default timeout is 1800 seconds (30 minutes).
Page Timeout	Specifies the amount of time to wait (in seconds) before a page is considered to be closed. (Note that the time is measured from the last packet received, not the start time of the page.) When a page is closed, a page event is generated and delivered, along with all of its corresponding http request events, to the reactor's output connections. Any requests for objects within the page reset this timer.
Maximum open pages per session	The maximum number of concurrently open pages in any single user/browser session that the Clickstream Reactor will analyze.

This creates the Clickstream Reactor in the workspace. Double-clicking on the reactor brings you to the configuration screen:


Clickstream Reactor Configuration

Clickstream Reactor

Name: ClickstreamReactor Category: Processing

Create unique user session information and statistics from raw web traffic data.

Name:		Session timeout (seconds):	1800
ID:	05d4a899-82f2-4b5d-852c-681f3	Page timeout (seconds):	60
Comments:		Maximum open pages per session:	5
Visitor ID persistence:		Maximum open events per session:	
Anonymous ID	☐	Ignore robot traffic	✓
Cookies	✓	Timeout uses event timestamps	☐

Session Groups

ID	Name	Host Suffixes	Session Cookies	Visitor Cookies	Delete

ADD NEW SESSION GROUP

Page Object Detection

☐ Match All Comparisons

Term	Comparison	Value	Match All	Delete
urn:vocab:clickstream#uri-stem	regex	.(gif|jpg|jpeg|png|ico|css|js|swf)$	☐	⊗
urn:vocab:clickstream#content-type	regex	(image/|application/|text/css|text/plain|javascript|xml|json)	☐	⊗

ADD NEW COMPARISON

Sticky Page Fields

Term	Delete

ADD NEW STICKY PAGE FIELD

Sticky Session Fields

Term	Delete

ADD NEW STICKY SESSION FIELD

In addition to the fields you saw when adding the reactor initially (described above), you will see some additional fields and sections for *Session Groups, Page Object Detection, Sticky Page Fields* and *Sticky Session Fields*, and *Connections* (not shown above). The additional fields are:

Field	Description
ID	The reactor ID, a unique user-defined ID. This ID identifies a particular reactor.

Field	Description
Visitor ID Persistence	Allows Pion to detect repeat visitors (across sessions) and tag all the associated events (session-events, page-events and request-events) with a common "visitor-id" field so that they can be matched. Two local databases are created and used to perform this mapping, one for "anonymous visitors" (visitors who have no cookie defined so they are identified only by IP address and user agent), and "Cookied Visitors" (visitors that do have a defined cookie). This area contains two checkboxes: • Anonymous ID Fill this check box to cause **all** visitors with the same 16 bits of IP address and user-agent to be tagged with the same "visitor-id." **Note**: This method of ID persistence is fairly inaccurate, and should be used only if no cookies are available.) • Cookies Fill this check box to cause repeat visitors with the same cookie values (for "visitor cookies" defined below in the session groups) to be tagged with the same visitor-id. **This is the default and is the recommended setting unless are certain that you don't need it.**
Maximum open events per session	If the total number of events being cached for a particular session exceeds this value, it forces the oldest page to close and flush the associated events. This is basically a sanity check to make sure events caused by spiders and other non-human visitors don't consume too much memory.

Field	Description
Ignore robot traffic (check box)	When this check box is filled, all sessions detected as "robots" or "spiders" are discarded by the Clickstream Reactor. When this check box is empty, events from sessions detected as "robots" or "spiders" are tagged with the "robot" field, but are not be discarded. A configuration file, "robots.xml" is used to control how Clickstream Reactor detects "robots."
Timeout uses event timestamps (check box)	When this check box is filled, page and session timeouts use the timestamps from events rather than system clock times to determine age. When using PCAP file (with Sniffer Reactor) or web server log file (with Log File Input Reactor), this check box should always be filled. When using live network interfaces, this check box should always be empty.

The Session Groups Section

The Session Groups section describes criteria which are used to determine sessions (to sessionize events).

Field	Description
ID	A short identifier that is assigned by Pion to the session-group field in all corresponding events. A "default" identifier is reserved for use by the default session group. Sessions that do not match the Host Suffixes of any other groups will be assigned to the "default" group, regardless of the Host HTTP request header. Note that a "default" session group is automatically generated (without cookies or host suffixes) if one is not otherwise specified in the configuration.

Field	Description
Name	A descriptive name for the session group.
Host Suffixes	Used to evaluate the Host HTTP request header to determine if an event belongs to this session group. Host suffixes are specified as a comma-separated list. One or more suffixes must be defined for each session group. Note that, if a suffix of "`atomiclabs.com`" is specified, "`atomiclabs.com`" will match, and so would "`pion.atomiclabs.com`" and "`www.atomiclabs.com`" as well.
Session Cookies	Used to sessionize traffic, the names of these cookies are specified in a comma-separated list. **For session cookies, you should only specify the names of cookies which have a lifetime of only the current session.** Cookies of this type are discarded after the session expires.
Visitor Cookies	Used to sessionize traffic, the names of these cookies are specified in a comma-separated list. **For visitor cookies, you should specify the names of cookies which have a lifetime that lasts beyond the current session.** These cookies are used for repeat-visitor detection (see "Visitor ID Persistence“ above). If the visitor returns the next day with the same visitor cookie, they will get tagged with the same visitor id (assuming "Cookies" visitor ID persistence is enabled above). Visitor cookies are persisted in an embedded cookies database within Pion. The cookies database contains a timestamp, and can be pruned using the purge_cookies.pl script.

To add a Session Group, click on the ADD NEW SESSION GROUP link. The Session Group Initialization screen appears.

To delete a Session Group, click on the “x” at the end of the row describing that Session Group.

The Page Object Detection Section

The Page Object Detection section describes conditions that must be met in order for an event to be considered a valid page object. This section works in the same manner as a comparison (a Filter Rule Chain), see "Comparisons“ on page 5 for more information. If the event matches the comparison criteria, then the http request is considered to be a "page object" or component of another page view request. If the event does not match the comparison criteria, then the http request is considered to represent a new page view.

Field	**Description**
Match All Comparisons (check box)	When filled, indicates that ALL of the comparisons defined below must be matched.
Term	Defines the value of the term to be used for the comparison logic. Specifies the data item to find. You can select a term (an identified data item) from an existing vocabulary, or you can add a new vocabulary and define the term(s) to be found.
Comparison	Specifies the type of comparison to test, such as true/false or is-defined/is-not-defined. Depending on the term that is specified in a row, an appropriate list of comparisons will appear in the comparison cell of that row. For example, if it's a date type, you'll get choices like "earlier than", whereas if it's a string type, you'll get choices like "starts with". Like most comparisons, these two examples require a value to compare against, but a few comparisons, like “is defined,” do not require a value to compare against. For more information on comparisons, see "Comparisons“ on page 5.
Value	
Match All (check box)	When filled, indicates that AT LEAST this comparison must be matched.

To add a row for a new comparison in the Page Object Detection section, click on the ADD NEW COMPARISON link.

To delete a Page Object Detection comparison, click on the “x” at the end of the row describing that comparison.

The Sticky Page Fields Section

The Sticky Page Fields section describes a list of terms to be copied from http-events into page-events.

Field	**Description**
Term	Defines the name of the Term to be copied from http events into page events. Note that Sticky Page Fields work in a similar fashion as the "Terms to copy" section of the Fission Reactor.

To add a new Sticky Page Field, click on the ADD NEW STICKY PAGE FIELD link.

To delete a Sticky Page Field, click on the “x” at the end of the row describing that field.

The Sticky Session Fields Section

The Sticky Session Fields section describes a list of terms to be copied from page-events into session-events.

Field	**Description**
Term	Defines the name of the Term to be copied from page events into session events. Note that Sticky Page Fields work in a similar fashion as the "Terms to copy" section of the Fission Reactor.

To add a new Sticky Session Field, click on the ADD NEW STICKY SESSION FIELD link.

To delete a Sticky Session Field, click on the “x” at the end of the row describing that field.

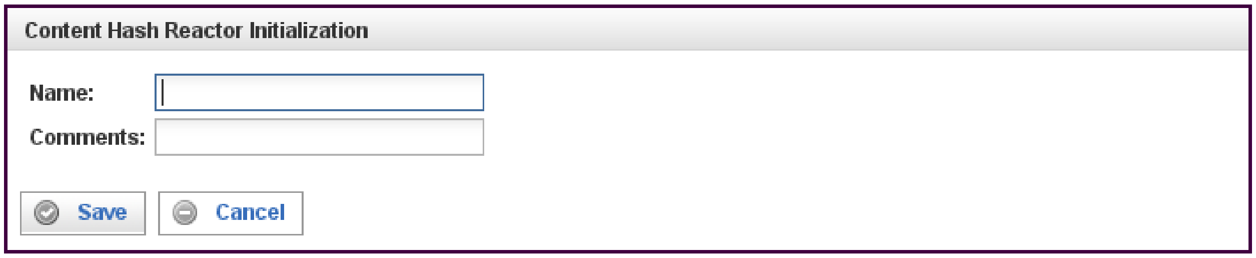
The Connections Section

The Connections section describes input and output connections for this Clickstream Reactor. See “Reactor Input and Output Connections“ on page 6 for more information.

Content Hash Reactor

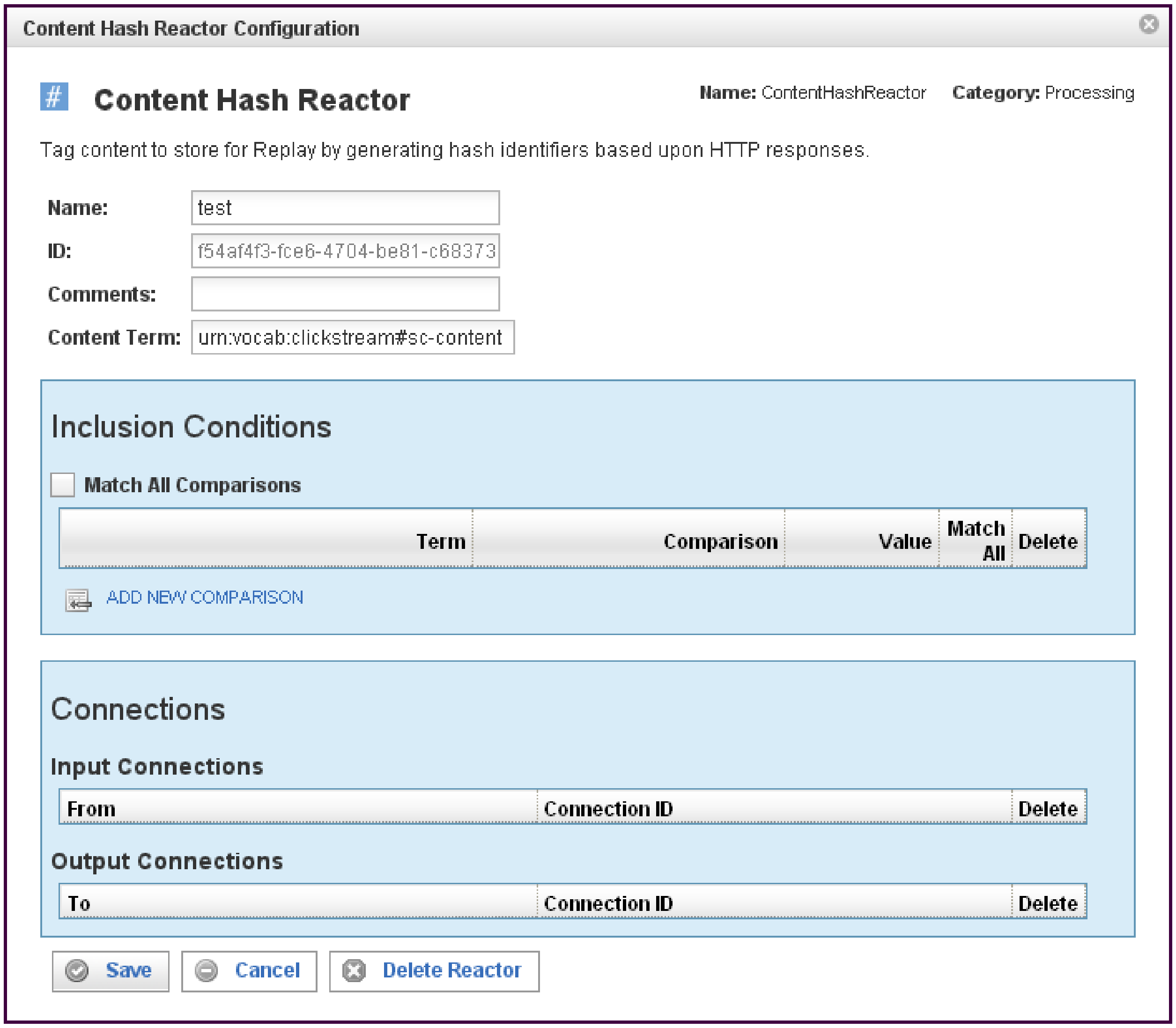
The Content Hash Reactor creates and then stores an md5 hash as an identifier (a content_id) for a specified content term in an HTTP response. This identifier is stored in a database, which is then used by the Replay feature when reviewing the user experience.

When you add a Content Hash Reactor you will be prompted for the following fields:



Field	Description
Name	Specify the name of this reactor. This can be any name you find descriptive, we recommend something which describes the data this reactor is processing. The name may be any combination of letters and numbers.
Comments	Enter some descriptive comments about this reactor. Any comment you feel will help you in the future, we recommend choosing something that describes the data being processed.

This creates the Content Hash Reactor in the workspace. Double-clicking on the reactor brings you to the configuration screen:


Content Hash Reactor Configuration

Content Hash Reactor

Name: ContentHashReactor Category: Processing

Tag content to store for Replay by generating hash identifiers based upon HTTP responses.

Name:	test
ID:	f54af4f3-fce6-4704-be81-c68373
Comments:	
Content Term:	urn:vocab:clickstream#sc-content

Inclusion Conditions

Match All Comparisons

Term	Comparison	Value	Match All	Delete

ADD NEW COMPARISON

Connections

Input Connections

From	Connection ID	Delete

Output Connections

To	Connection ID	Delete

In addition to the fields you saw when adding the reactor initially (described above), you will see some additional fields and sections for *Inclusion Conditions* and *Connections*. The additional fields are:

Field	**Description**
ID	The reactor ID, a unique ID generated by the Pion server. This ID identifies a particular reactor.
Content Term	The source term used to generate a content hash identifier.

The Inclusion Conditions Section

The Inclusion Conditions section describes conditions that must be met in order for a prospective Content Term to be considered a new Content Term (which causes a new identifier to be generated). This section works in the same manner as a comparison (a Filter Rule Chain), see “Comparisons“ on page 5 for more information.

If an event both contains a value for the Content Term and matches the Inclusion Conditions, a hash identifier will be generated and added to the event.

If the event does not contain a value for the Content Term or it does not match the Inclusion Conditions, the event is delivered to the reactor's output connections unmodified.

Field	**Description**
Match All Comparisons (check box)	When filled, indicates that ALL of the comparisons defined below must be matched.
Term	Defines the value of the term to be used for the comparison logic. Specifies the data item to find. You can select a term (an identified data item) from an existing vocabulary, or you can add a new vocabulary and define the term(s) to be found.
Comparison	Specifies the type of comparison to test, such as true/false or is-defined/is-not-defined. Depending on the term that is specified in a row, an appropriate list of comparisons will appear in the comparison cell of that row. For example, if it's a date type, you'll get choices like "earlier than", whereas if it's a string type, you'll get choices like "starts with". Like most comparisons, these two examples require a value to compare against, but a few comparisons, like “is defined,” do not require a value to compare against. For more information on comparisons, see “Comparisons“ on page 5.
Value	
Match All (check box)	When filled, indicates that AT LEAST this comparison must be matched.

To add a row for a new comparison in the *Inclusion Conditions* section, click on the ADD NEW COMPARISON link.

To delete a comparison, click on the “x” at the end of the row describing that comparison.

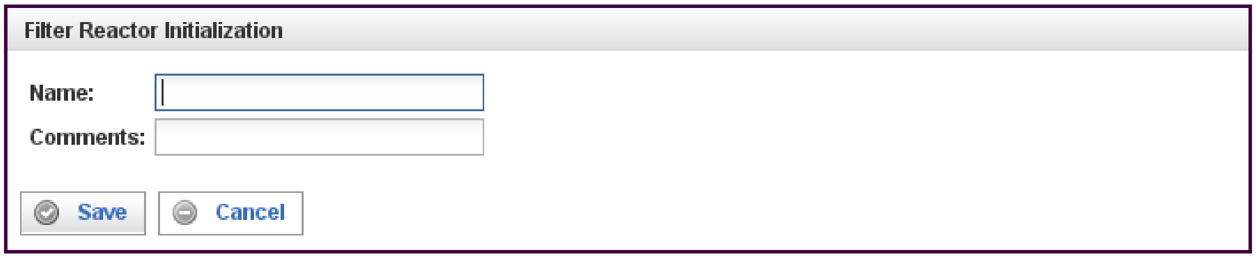
The Connections Section

The Connections section describes input and output connections for this Content Hash Reactor. See “Reactor Input and Output Connections“ on page 6 for more information.

Filter Reactor

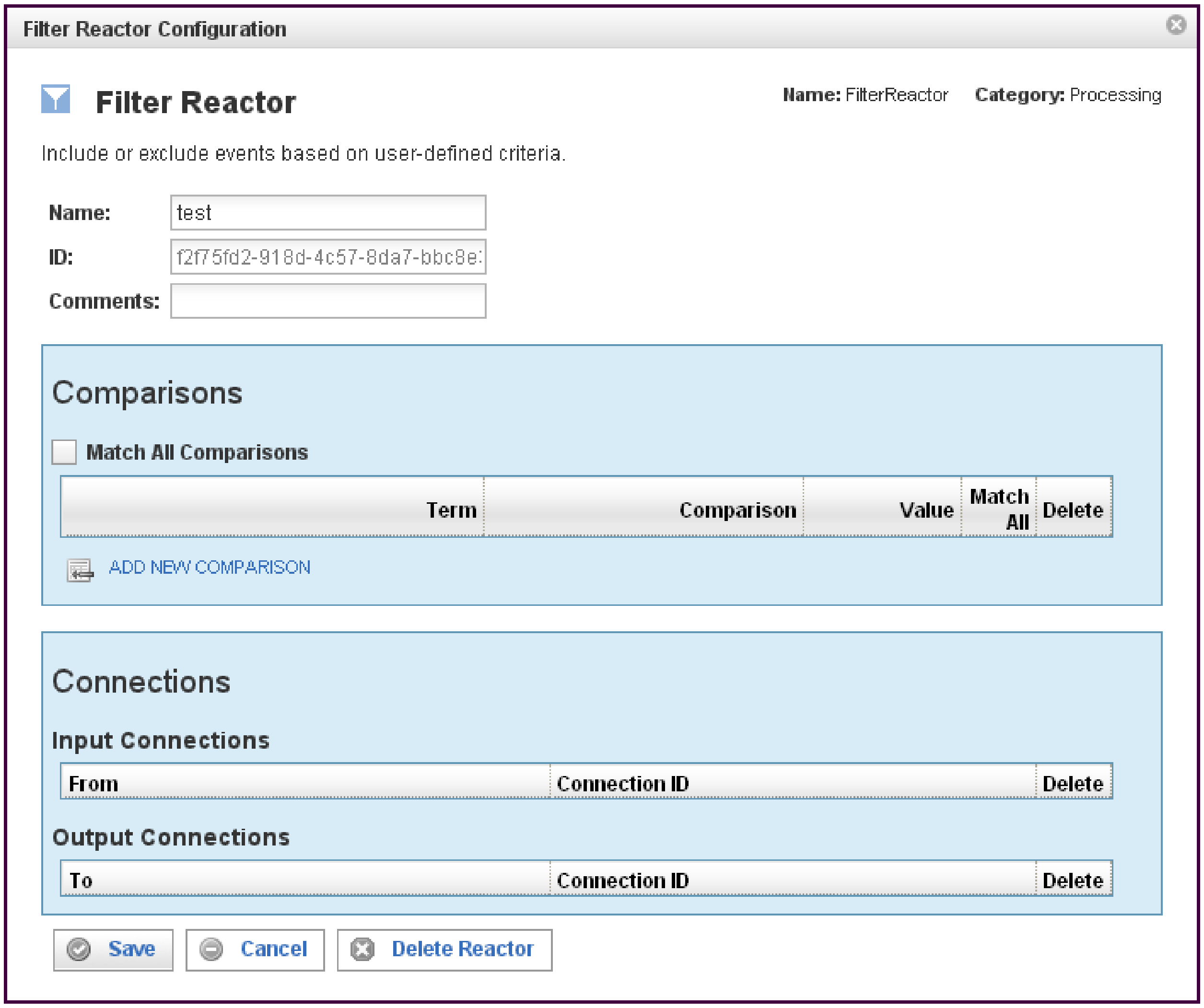
The Filter Reactor can filter in or out any information based on any number of criteria you might set. For example, you could do something simple like only looking at a particular set of users of web servers. Or you could do something much more complicated such as only tracking certain individuals who are spending more than $50 on kitchen ware in their sessions.

When you add a Filter Reactor you will be prompted for the following fields:



Field	**Description**
Name	Specify the name of this reactor. This can be any name you find descriptive, we recommend something which describes the data this reactor is processing. The name may be any combination of letters and numbers.
Comments	Enter some descriptive comments about this reactor. Any comment you feel will help you in the future, we recommend choosing something that describes the data being processed.

This creates the Filter Reactor in the workspace. Double-clicking on the reactor brings you to the configuration screen:


Filter Reactor Configuration

Filter Reactor

Name: FilterReactor Category: Processing

Include or exclude events based on user-defined criteria.

Name:	test
ID:	f2f75fd2-918d-4c57-8da7-bbc8e
Comments:	

Comparisons

Match All Comparisons

Term	Comparison	Value	Match All	Delete

ADD NEW COMPARISON

Connections

Input Connections

From	Connection ID	Delete

Output Connections

To	Connection ID	Delete

In addition to the fields you saw when adding the reactor initially (described above), you will see some additional fields and sections for *Comparisons* and *Connections*. The additional fields are:

Field	**Description**
ID	The reactor ID, a unique ID generated by the Pion server. This ID identifies a particular reactor.

The Comparisons Section

The Comparisons section describes the comparisons to be made in this Filter Reactor. If the event matches the comparison criteria, it will be delivered to all Output Connections. If the event does not match the comparison criteria, it is dropped. See “Comparisons“ on page 5 for more information.

Field	**Description**
Match All (check box)	When filled, indicates that ALL of the comparisons defined below must be matched.
Term	Defines the value of the term to be used for the comparison logic. Specifies the data item to find. You can select a term (an identified data item) from an existing vocabulary, or you can add a new vocabulary and define the term(s) to be found.
Comparison	Specifies the type of comparison to test, such as true/false or is-defined/is-not-defined. Depending on the term that is specified in a row, an appropriate list of comparisons will appear in the comparison cell of that row. For example, if it's a date type, you'll get choices like "earlier than", whereas if it's a string type, you'll get choices like "starts with". Like most comparisons, these two examples require a value to compare against, but a few comparisons, like “is defined,” do not require a value to compare against. For more information on comparisons, see “Comparisons“ on page 5.
Value	Specifies a value to be compared against the value of the selected term.
Match All (check box)	When filled, and if the term has multiple values (a vare rare occurrence).

To add a row for a new comparison in the *Comparison* section, click on the ADD NEW COMPARISON link.

To delete a comparison, click on the “x” at the end of the row describing that comparison.

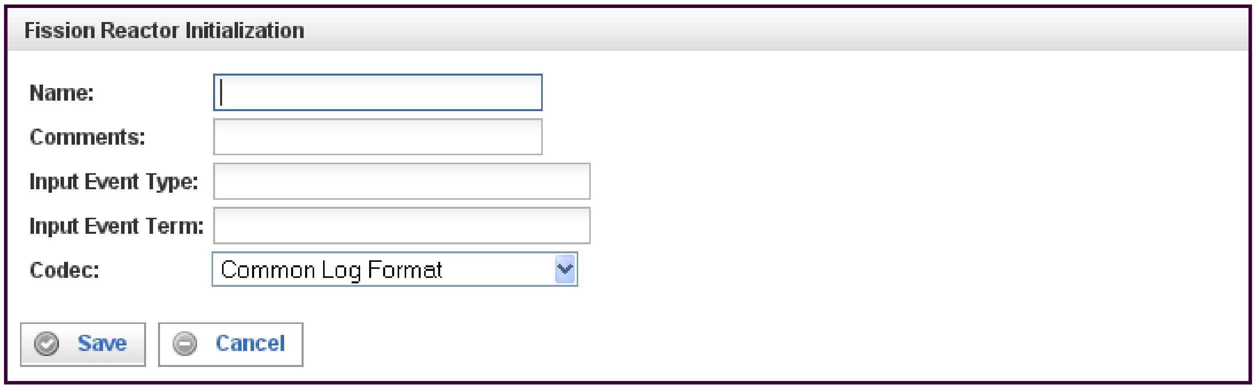
The Connections Section

The Connections section describes input and output connections for this Filter Reactor. See “Reactor Input and Output Connections” on page 6 for more information.

Fission Reactor

Filter Reactor, splits input events into sequences of output events based on any number of criteria you might set.

When you add a Fission Reactor you will be prompted for the following fields:

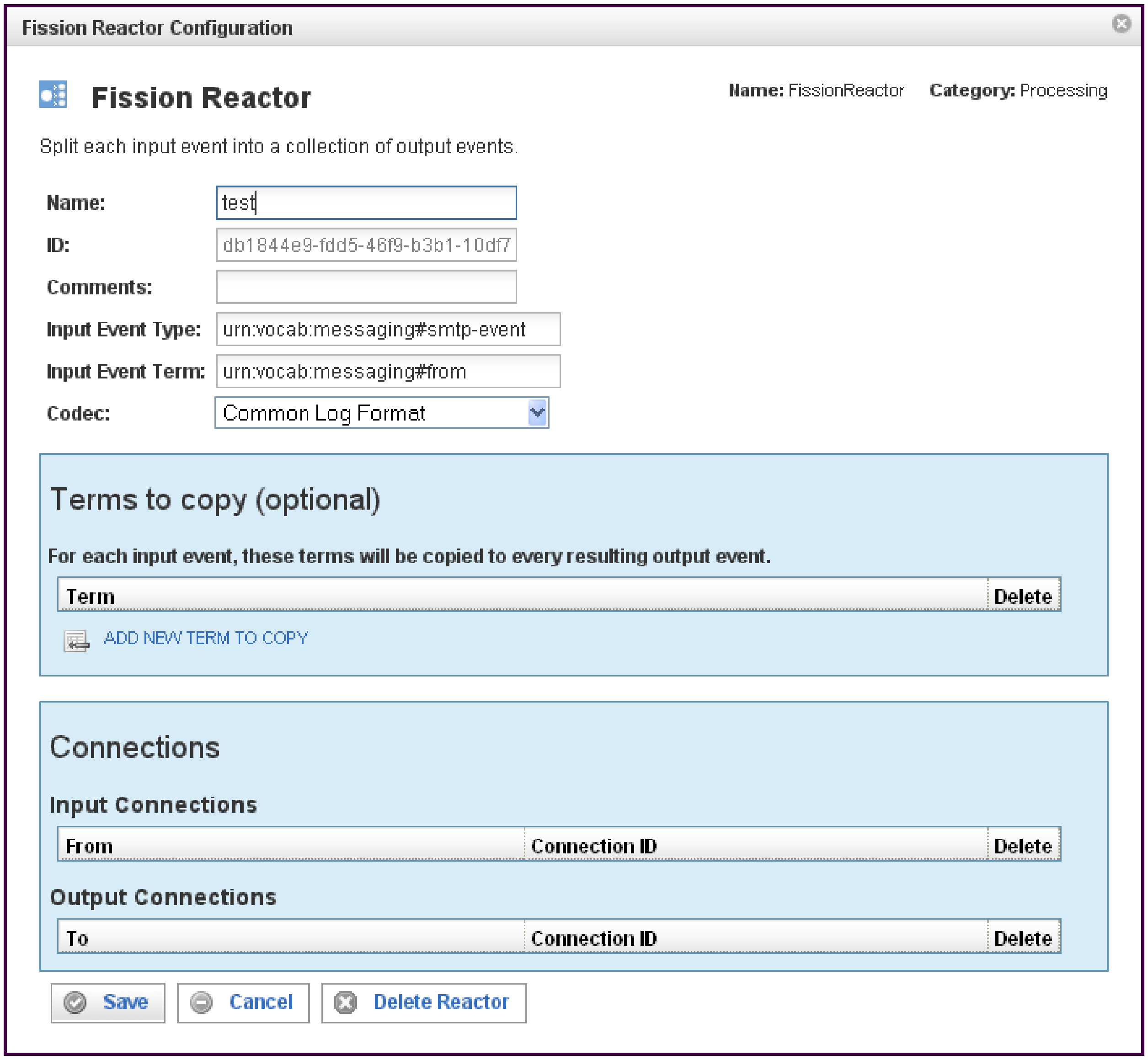

Fission Reactor Initialization

Name:	
Comments:	
Input Event Type:	
Input Event Term:	
Codec:	Common Log Format

Field	Description
Name	Specify the name of this reactor. This can be any name you find descriptive, we recommend something which describes the data this reactor is processing. The name may be any combination of letters and numbers.
Comments	Enter some descriptive comments about this reactor. Any comment you feel will help you in the future, we recommend choosing something that describes the data being processed.
Input Event Type	The input event type must be a Term of type "object" (this rule is enforced by the Pion user interface). Only input events of this type are processed; any other events are dropped. An Input Event Type must be specified.

Field	Description
Input Event Term	A term to be parsed from the Input Event (described above). The type of the input event term must be one of the string types (this rule is enforced by the Pion user interface). One or more values from each Input Event Term in the input event are parsed by the Codec to create output events. An Input Event Term must be specified.
Codec	Defines the codec used to parse the Input Event Term(s) specified above. A Codec must be specified.

This creates the Fission Reactor in the workspace. Double-clicking on the reactor brings you to the configuration screen:



Fission Reactor Configuration

Fission Reactor

Name: FissionReactor **Category:** Processing

Split each input event into a collection of output events.

Name:	test
ID:	db1844e9-fdd5-46f9-b3b1-10df7
Comments:	
Input Event Type:	urn:vocab:messaging#smtp-event
Input Event Term:	urn:vocab:messaging#from
Codec:	Common Log Format

Terms to copy (optional)

For each input event, these terms will be copied to every resulting output event.

Term	Delete

ADD NEW TERM TO COPY

Connections

Input Connections

From	Connection ID	Delete

Output Connections

To	Connection ID	Delete

In addition to the fields you saw when adding the reactor initially (described above), you will see some additional fields and sections for *Terms to copy* and *Connections*. The additional fields are:

Field	**Description**
ID	The reactor ID, a unique ID generated by the Pion server. This ID identifies a particular reactor.

The Terms to Copy Section

The Inclusion Conditions section describes a list of terms whose values will be copied into all output events.

Field	**Description**
Terms (to copy)	The value or values of each of the terms defined will be copied from the original event into all derived events.

To add a new term in the *Terms to copy* list, click on the ADD NEW TERM TO COPY link. When you click on the new row, a list of vocabularies and their associated terms appears. Select the vocabulary and term you want to add, then click Select term.

To delete a term from the list, click on the “x” at the end of the row containing that term.

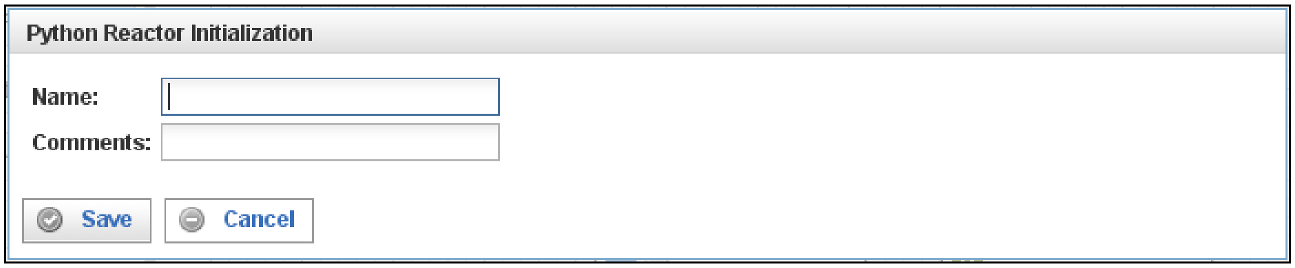
The Connections Section

The Connections section describes input and output connections for this Fission Reactor. See “Reactor Input and Output Connections“ on page 6 for more information.

Python Reactor

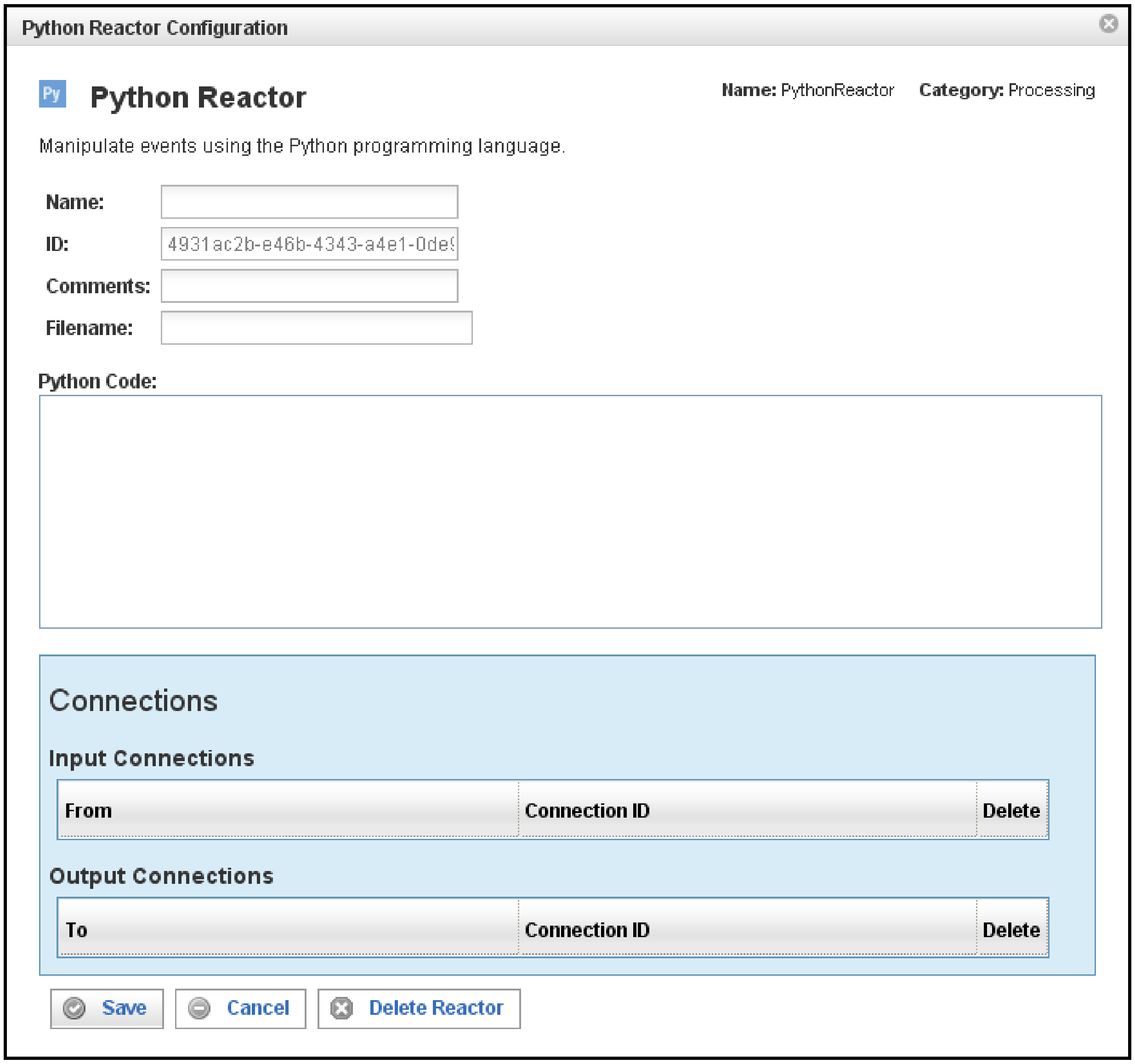
The Python Reactor allows you to use the popular Python programming language to collect, manipulate and store events within Pion.

When you add a Python Reactor you will be prompted for the following fields:



Field	Description
Name	Specify the name of this reactor. This can be any name you find descriptive, we recommend something which describes the data this reactor is processing. The name may be any combination of letters and numbers.
Comments	Enter some descriptive comments about this reactor. Any comment you feel will help you in the future, we recommend choosing something that describes the data being processed.

This creates the Python Reactor in the workspace. Double-clicking on the reactor brings you to the configuration screen:



Python Reactor Configuration

Python Reactor

Name: PythonReactor Category: Processing

Manipulate events using the Python programming language.

Name:	
ID:	4931ac2b-e46b-4343-a4e1-0de
Comments:	
Filename:	

Python Code:

Connections

Input Connections

From	Connection ID	Delete

Output Connections

To	Connection ID	Delete

In addition to the fields you saw when adding the reactor initially (described above), you will see one additional field and the *Connections* section. The additional field is:

Field	**Description**
ID	The reactor ID, a unique ID generated by the Pion server. This ID identifies a particular reactor.
Filename	If specified, this file should contain the Python source code used to manipulate events.

Field	Description
Python Code	If a file name is not specified above, you may enter your Python code into this box (see About Your Source Code in the Python Reactor, below). Code in this box is ignored if a file name is specified above.

About Your Source Code in the Python Reactor

The Python Reactor automatically compiles your source code into a byte code format that is stored in memory. Your code is automatically imported into a Pion module which supports two special data types, `Event` and `Reactor`, and three "callback" functions, `start()`, `stop()`, and `process()`.

The Event Data Type

The Event data type is a subclass of the Python dict (dictionary) type. It contains one additional attribute:

```
(name / type    / description)
-------------------------------
type    str       type of Event (i.e. "urn:vocab:clickstream#http-event")
```

Events must always have a valid "type" (as configured in a Pion vocabulary). The keys of Event items must be string types representing a particular Term identifier (i.e. "urn:vocab:clickstream#uri-stem"). The value of an Event item must be a Python sequence (tuple or list) containing zero or more objects (note that in most cases it will contain a single item). All objects in a value sequence must be of a type corresponding to the Pion data type assigned to the corresponding Term.

The following are the Python types that correspond to Pion Vocabulary data types:

Pion Type	Python Type
undefined	int = 1
event type	int = 1
signed int (8-bit)	int
positive int (8-bit)	int
signed int (16-bit)	int

Pion Type	**Python Type**
positive int (16-bit)	int
signed int (32-bit)	int
positive int (32-bit)	long
signed int (64-bit)	long
positive int (64-bit)	long
small real number	float
medium real number	float
large real number	float
small string	str
medium string	str
large string	str
fixed-length string	str
binary large object	str
blob (stored compressed)	str
specific date	date
specific time	time
specific time & date	datetime

The Reactor Data Type

The Reactor data type supports the attributes "`id`," "`name`," and "`deliver`" where:

- `id`: Is a string (`str`) value of the unique identifier of the Python Reactor.
- `name`: Is a string (`str`) value of the name assigned to the Python Reactor.
- `deliver`: Is a function that takes a single Event object argument, and delivers the event to all of the Python Reactor's output connections.

The Callback Functions

The following three callback functions are optional, and can be used to define the behavior of this reactor:

- `start(Reactor)`

  Pion calls this function whenever the reactor is started. It is given a single parameter, a Reactor object representing the Python Reactor.

- `stop(Reactor)`

  Pion calls this function whenever the reactor is stopped. It is given a single parameter, a Reactor object representing the Python Reactor.

- `process(Reactor, Event)`

  Pion calls this function whenever the reactor receives a new event to process via an input connection. The first parameter is a Reactor object ID representing the Python Reactor. The second parameter is an Event object representing the original Pion event.

  - If no `process()` function is provided, the Python Reactor passes all events it receives to its output connections without modification.
  - If a `process()` function is defined, you must explicitly deliver any events to the reactor's output connections by calling the deliver method of the Reactor object ("`r.deliver(e)`").

  Any exceptions raised by a custom `process()` function will result in an ERROR message being logged and the event being dropped.

The following is an example of callback functions that can be used by the Python Reactor:

```
URI_STEM = 'urn:vocab:clickstream#uri-stem'

def start(r):
    print 'starting reactor', r.name

def stop(r):
    print 'starting reactor', r.name

def process(r, e):
    print 'reactor', r.name, 'received an event of type', e.type
    e[URI_STEM] = ['/new_uri_stem']
    r.deliver(e)
```

Note that all code defined outside of the three callback functions will be executed when the reactor starts. Any classes or functions defined are reusable by the callback functions, and any attributes persist until the reactor is stopped.

For more information on the Python programming language, please see: http://www.python.org or the many books available.

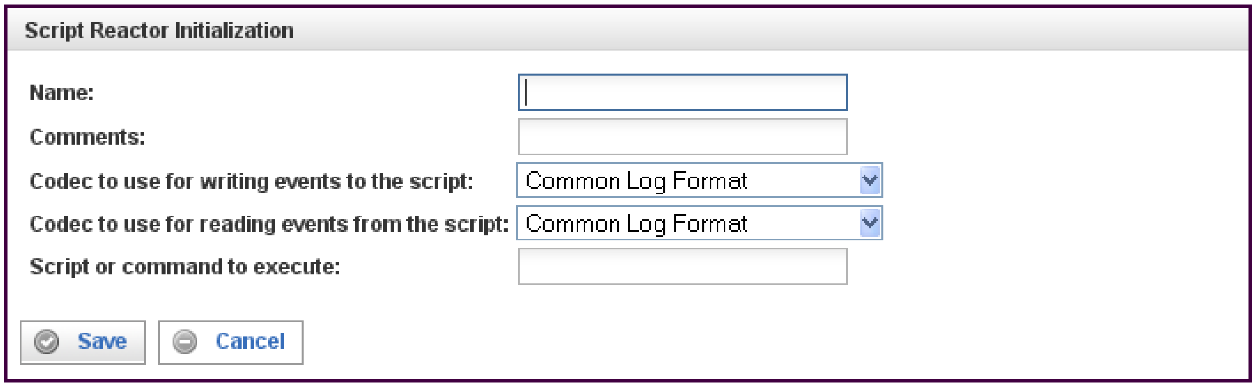
The Connections Section

The Connections section describes input and output connections for this Python Reactor. See “Reactor Input and Output Connections“ on page 6 for more information.

Script Reactor

The Script Reactor enables you to leverage existing command line scripts to quickly add capabilities to Pion. Through the use of the script reactor you can do things such as create follow-up queues based on customer behavior or retrieve additional information from disparate sources on the fly.

When you add a Script Reactor you will be prompted for the following fields:



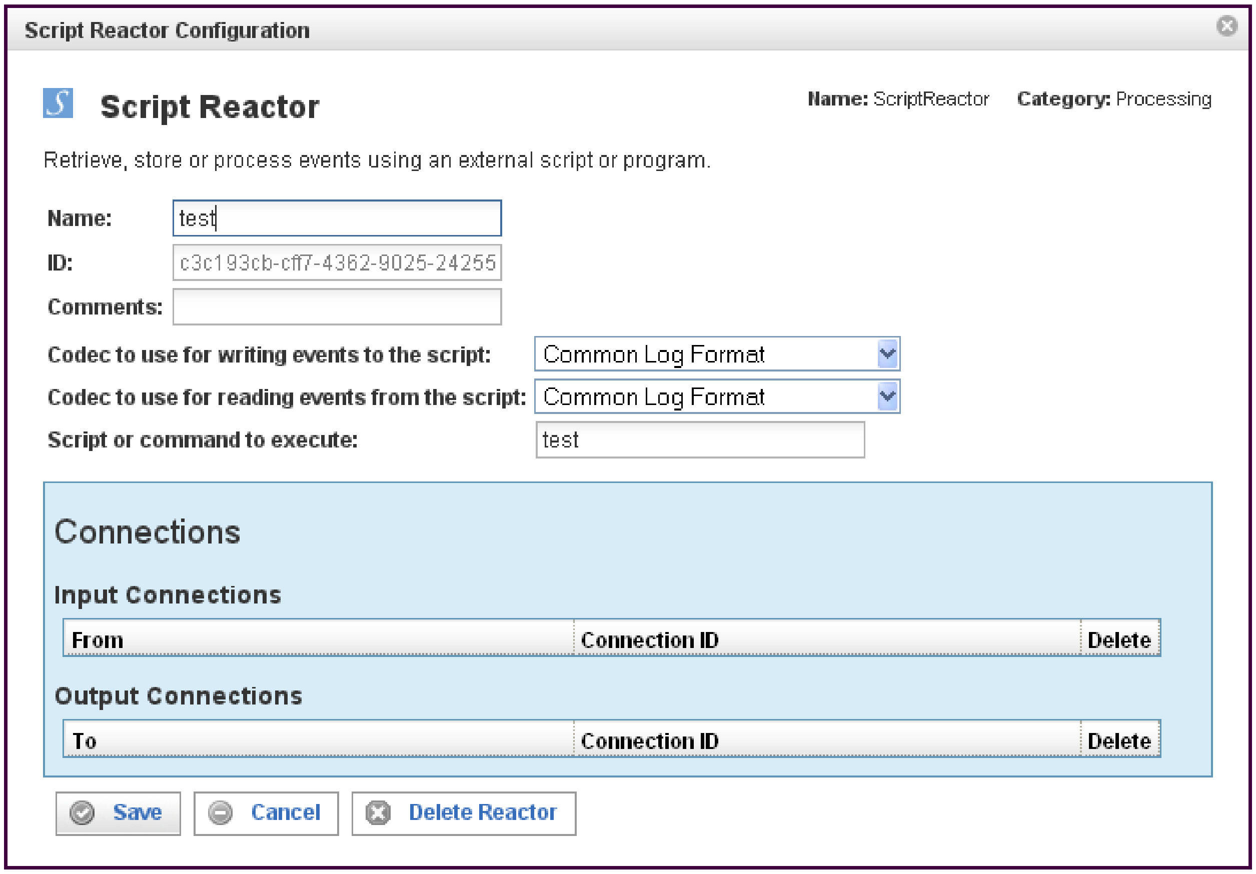
Script Reactor Initialization

Name:	
Comments:	
Codec to use for writing events to the script:	Common Log Format
Codec to use for reading events from the script:	Common Log Format
Script or command to execute:	

Field	Description
Name	Specify the name of this reactor. This can be any name you find descriptive, we recommend something which describes the data this reactor is processing. The name may be any combination of letters and numbers.
Comments	Enter some descriptive comments about this reactor. Any comment you feel will help you in the future, we recommend choosing something that describes the data being processed.
Codec to use for writing events to the script	Events from other reactors will be delivered to the script's stdin file descriptor. These events will be serialized using the specified codec.
Codec to use for reading events from the script	Events going to other reactors from this Script Reactor will be read using the script's stdout file descriptor. These events will be serialized using the specified codec.

Field	Description
Script or command to execute	When running, the Script Reactor will execute the command specified and pipe events through the command. The command is executed once and will remain running until the reactor is stopped (or the command terminates by itself).

This creates the Script Reactor in the workspace. Double-clicking on the reactor brings you to the configuration screen:



Script Reactor Configuration

Script Reactor

Name: ScriptReactor Category: Processing

Retrieve, store or process events using an external script or program.

Name:	test
ID:	c3c193cb-cff7-4362-9025-24255
Comments:	
Codec to use for writing events to the script:	Common Log Format
Codec to use for reading events from the script:	Common Log Format
Script or command to execute:	test

Connections

Input Connections

From	Connection ID	Delete

Output Connections

To	Connection ID	Delete

In addition to the fields you saw when adding the reactor initially (described above), you will see one additional field and the *Connections* section. The additional field is:

Field	Description
ID	The reactor ID, a unique ID generated by the Pion server. This ID identifies a particular reactor.

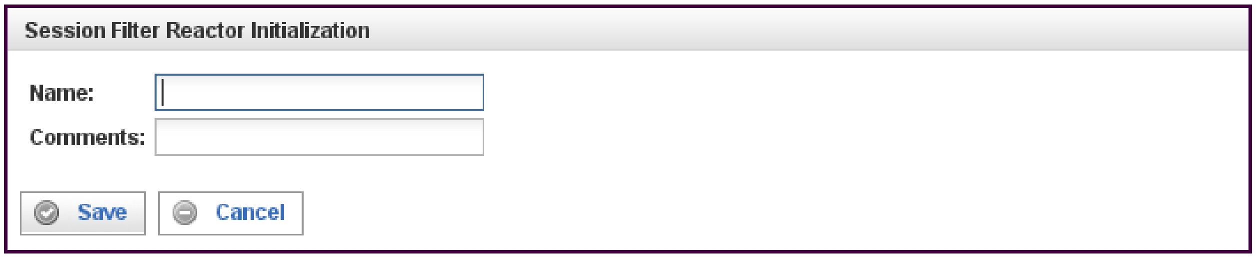
The Connections Section

The Connections section describes input and output connections for this Script Reactor. See “Reactor Input and Output Connections“ on page 6 for more information.

Session Filter Reactor

The Session Filter Reactor enables you to hold entire sessions until they meet certain criteria. The Session Filter Reactor must always be used after a Clickstream Reactor to ensure that the sessions are created. Using the Session Filter Reactor, you can do things such as saving full page view detail on only users that purchased a certain book or encountered an error.

When you add a Filter Reactor you will be prompted for the following fields:

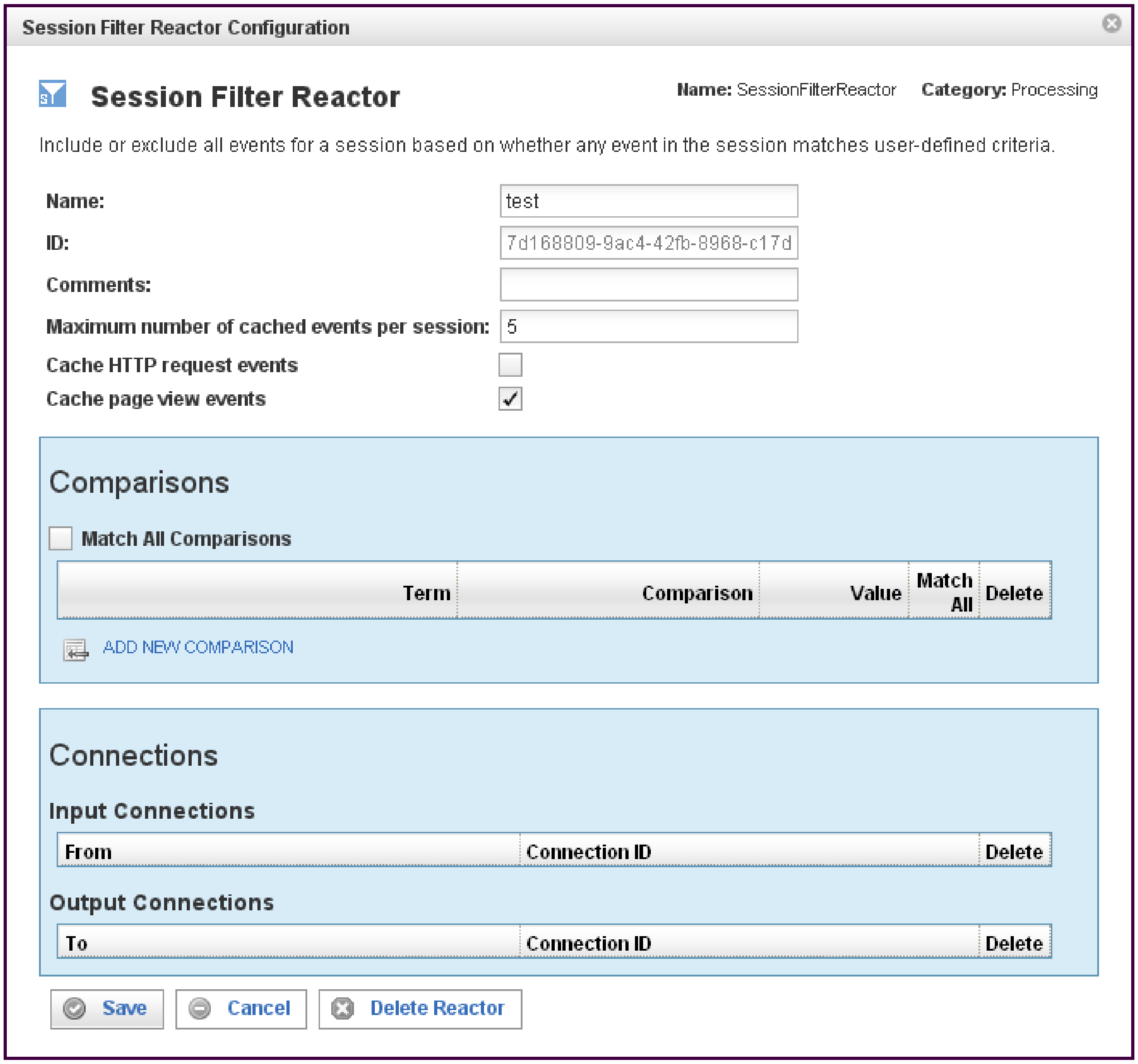

Session Filter Reactor Initialization

Name:

Comments:

Field	**Description**
Name	Specify the name of this reactor. This can be any name you find descriptive, we recommend something which describes the data this reactor is processing. The name may be any combination of letters and numbers.
Comments	Enter some descriptive comments about this reactor. Any comment you feel will help you in the future, we recommend choosing something that describes the sessions being held.

This creates the Filter Reactor in the workspace. Double-clicking on the reactor brings you to the configuration screen:


Session Filter Reactor Configuration

Session Filter Reactor

Name: SessionFilterReactor **Category:** Processing

Include or exclude all events for a session based on whether any event in the session matches user-defined criteria.

Name:	test
ID:	7d168809-9ac4-42fb-8968-c17d
Comments:	
Maximum number of cached events per session:	5
Cache HTTP request events	☐
Cache page view events	✓

Comparisons

☐ Match All Comparisons

Term	Comparison	Value	Match All	Delete

ADD NEW COMPARISON

Connections

Input Connections

From	Connection ID	Delete

Output Connections

To	Connection ID	Delete

In addition to the fields you saw when adding the reactor initially (described above), you will see some additional fields and sections for *Comparisons* and *Connections*. The additional fields are:

Field	**Description**
ID	The reactor ID, a unique ID generated by the Pion server. This ID identifies a particular reactor.

Field	Description
Maximum number of cached events per session	Specifies the maximum number of cached events per session. If the total number of cached events (individual HTTP requests AND page view events) for a particular session exceeds this value, the oldest events are discarded. This essentially creates a "rolling window" of events leading up to the point where the session passes the comparison rules. All events for the session are delivered immediately after the comparison rules are met (without caching). Setting this value to 0 will cache all events indefinitely, but we strongly recommend that a reasonable maximum be defined unless you have VERY little traffic, because this can otherwise quickly consume all the memory available.
Cache HTTP request events (check box)	When the check box is filled, HTTP request events are cached until the comparisons are met. After the comparisons are met, all events for the session are delivered immediately.
Cache page view events (check box)	When the check box is filled, page view events are cached until the comparisons are met. After the comparisons are met, all events for the session are delivered immediately.

The Comparisons Section

The Comparisons section describes comparison to be made in this Session Filter Reactor. See “Comparisons“ on page 5 for more information.

Field	Description
Match All (check box)	When filled, indicates that ALL of the comparisons defined below must be matched.

Field	**Description**
Term	Defines the value of the term to be used for the comparison logic. Specifies the data item to find. You can select a term (an identified data item) from an existing vocabulary, or you can add a new vocabulary and define the term(s) to be found.
Comparison	Specifies the type of comparison to test, such as true/false or is-defined/is-not-defined. Depending on the term that is specified in a row, an appropriate list of comparisons will appear in the comparison cell of that row. For example, if it's a date type, you'll get choices like "earlier than", whereas if it's a string type, you'll get choices like "starts with". Like most comparisons, these two examples require a value to compare against, but a few comparisons, like “is defined,” do not require a value to compare against. For more information on comparisons, see “Comparisons“ on page 5.
Value	
Match All (check box)	When filled, indicates that AT LEAST this comparison must be matched.

To add a row for a new comparison in the *Comparison* section, click on the ADD NEW COMPARISON link.

To delete a comparison, click on the “x” at the end of the row describing that comparison.

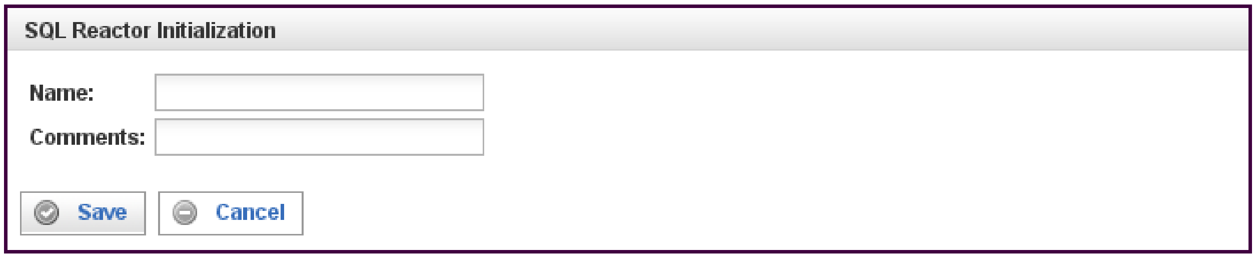
The Connections Section

The Connections section describes input and output connections for this Session Filter Reactor. See ”Reactor Input and Output Connections“ on page 6 for more information.

SQL Reactor

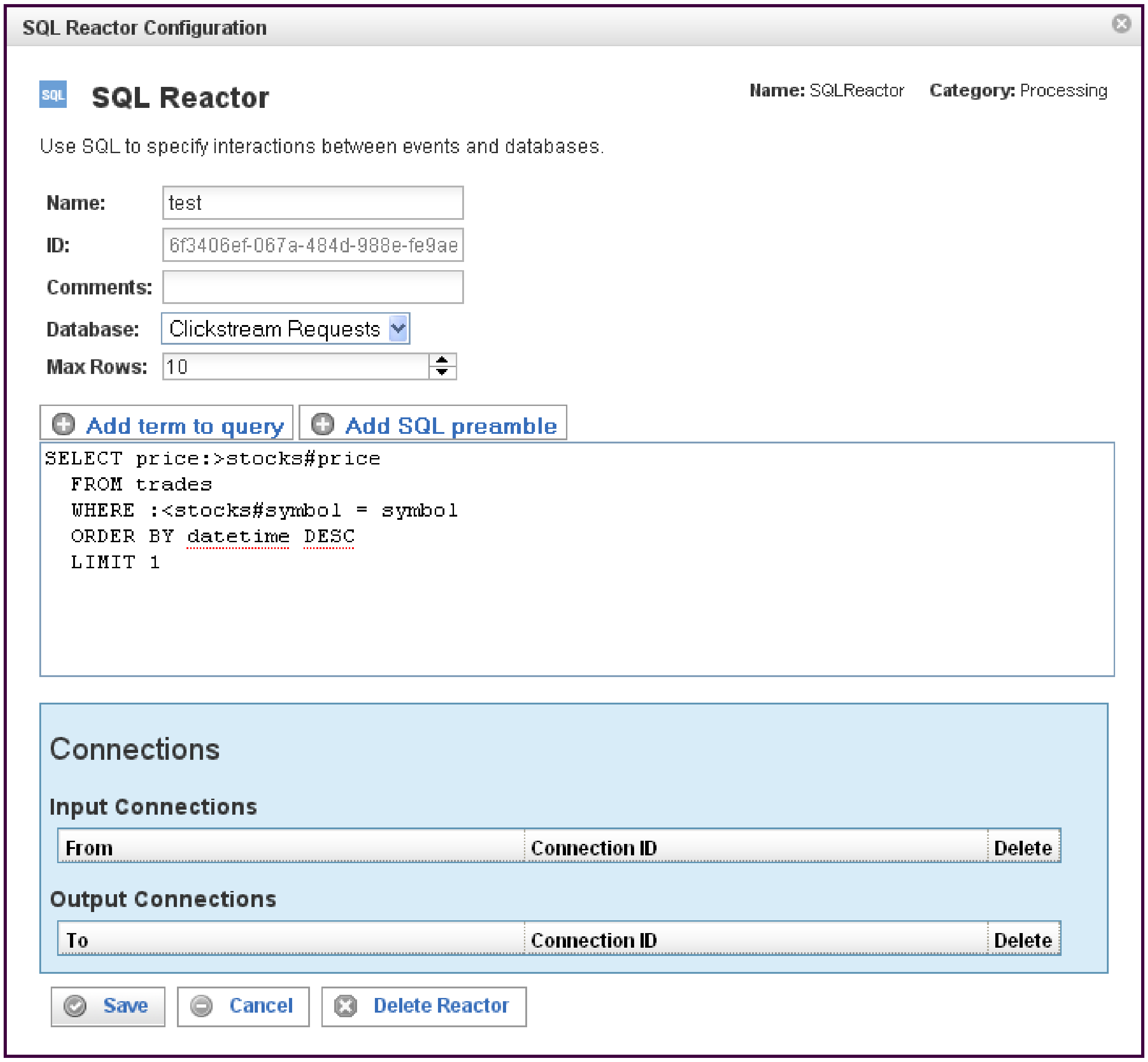
SQL Reactor, which lets Pion interact with any relational database based on real time information. This means that Pion can gather additional information on sessions based on past activity. For example, you can see how much a customer spent during the last 2 years to add context to their session and to improve segmentation for analytics. Based on real-time information, the SQL Reactor can also update or insert data in the database to ensure that the information you have is always the most current possible.

When you add a SQL Reactor you will be prompted for the following fields:



Field	**Description**
Name	Specify the name of this reactor. This can be any name you find descriptive, we recommend something which describes the data this reactor is processing. The name may be any combination of letters and numbers.
Comments	Enter some descriptive comments about this reactor. Any comment you feel will help you in the future, we recommend choosing something that describes the data being processed.

This creates the SQL Reactor in the workspace. Double-clicking on the reactor brings you to the configuration screen:



Name: SQLReactor **Category:** Processing

Use SQL to specify interactions between events and databases.

Name:	test
ID:	6f3406ef-067a-484d-988e-fe9ae
Comments:	
Database:	Clickstream Requests
Max Rows:	10

```
SELECT price:>stocks#price
  FROM trades
  WHERE :<stocks#symbol = symbol
  ORDER BY datetime DESC
  LIMIT 1
```

Connections

Input Connections

From	Connection ID	Delete

Output Connections

To	Connection ID	Delete

In addition to the fields you saw when adding the reactor initially (described above), you will see some additional fields and sections for *Comparisons* and *Connections*. The additional fields are:

Field	Description
ID	The reactor ID, a unique ID generated by the Pion server. This ID identifies a particular reactor.
Database	The database on which queries are executed.

Field	Description
Max Rows	The maximum number of rows allowed in a result set, limiting the size of a result set. One of the most damaging aspects of database query results is the possibility of a large result set size, which generally locks the table, or consumes resources until the result set has been transferred. Even though is advisable to put a LIMIT *n* statement into the query to limit the result set size, Pion allows you to define the result set size using Max Rows. When Max Rows is specified, after *n* results have been received and processed into the event, the rest of the result set is discarded and resources are freed.
Add term to query	When selected, opens up a term selector dialog, exposing the available vocabularies and the terms in the vocabularies. When a term is selected, shorthand notation is inserted into the query. Preface the term with "`:<`" for input, or "`:>`" for output.
Add SQL preamble	The SQL preamble allows you to configuring a set of SQL queries. The SQL queries in the SQL preamble are executed as the SQL Reactor starts. The SQL preamble is commonly used to establish things such as: Pragma's (in SQLite); to limit result set, locking, caching, etc. Views, if you are using materialized views. Keep-connection-open configurations. Linking tables in MySQL(federated tables) or SQLite (attaching databases).

The SQL query	The SQL query to be executed, in "Pion SQL."
	
	In the sample above:
	
	`SELECT price:>stocks#price`
	`   FROM trades`
	`   WHERE :<stocks#symbol = symbol`
	`   ORDER BY datetime DESC`

Field	Description
	`LIMIT 1`
	The table "trades" is assumed to be available via the database connection.
	The columns "price" and "symbol" are assumed to exist in the “trades” table.
	A query is executed using the Pion term "`urn:vocab:stocks#symbol`" (represented in short-hand form as "`stocks#symbol`").
	The term `stocks#symbol` is used as an INPUT parameter (":<" denotes input), meaning that the statement "`:<stocks#symbol`" will be replaced with the actual value of the term, in a SQL injection safe manner.
	The value of the column "price" will be put into the Pion term "`stocks#price`" (the full Pion notation is “`urn:vocab:stocks#price`" upon result.
	Note: If more than one line were permitted (LIMIT 1), the term `stocks#price` may contain multiple values.
	There is no practical limit of number of parameters that can be passed into the database, nor a limit on parameters coming back.
	Note: There is no inherent requirement that the query is actually a query (SELECT). Instead, the SQL statement can be:
	`INSERT: "INSERT INTO trades VALUES (:<stocks#symbol,:<stocks#price)`
	DELETE: "DELETE FROM trades WHERE symbol = :<stocks#symbol"
	Also, note that a table name is not required at all, the SQL Reactor can be used against "DUAL" or shadow table:

Field	Description
	`"SELECT :<stocks#price + :<stocks#price :>stocks#price"` Which effectively will take the `stocks#price` and multiply by two. The same method can be used for tricks, such as concatenating strings, doing string functions, math, etc. Remember that the actual code is dependent on the flavor of SQL used. Pion has a complete version of SQLite embedded, which provides a rich set of math, string, and date functions.

The Connections Section

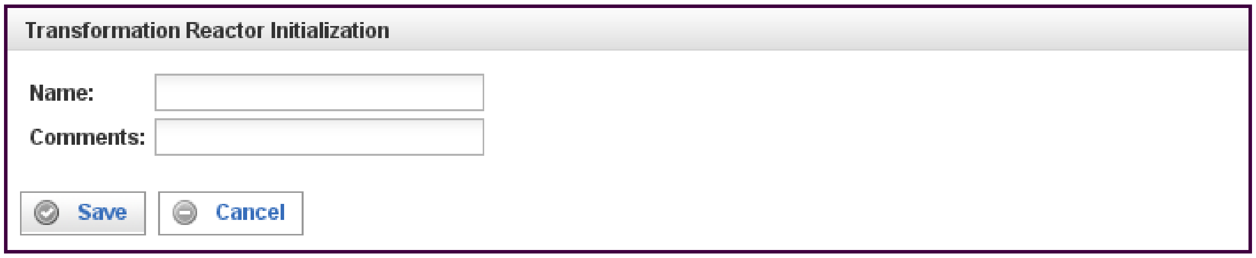
The Connections section describes input and output connections for this SQL Reactor. See ”Reactor Input and Output Connections“ on page 6 for more information.

Transform Reactor

Transform Reactor, which is able to do deep manipulation of the data in the data stream. For example, this can be as simple as formatting a URL to be something more easily readable or consistent if sites are constantly changing, or calculating dollar values from a page to see total spend (even taking into account if the currency must be converted). Because the Transform Reactor is very flexible, it can also do more complex things such as capturing and translating a bank branch or user identification on the fly.

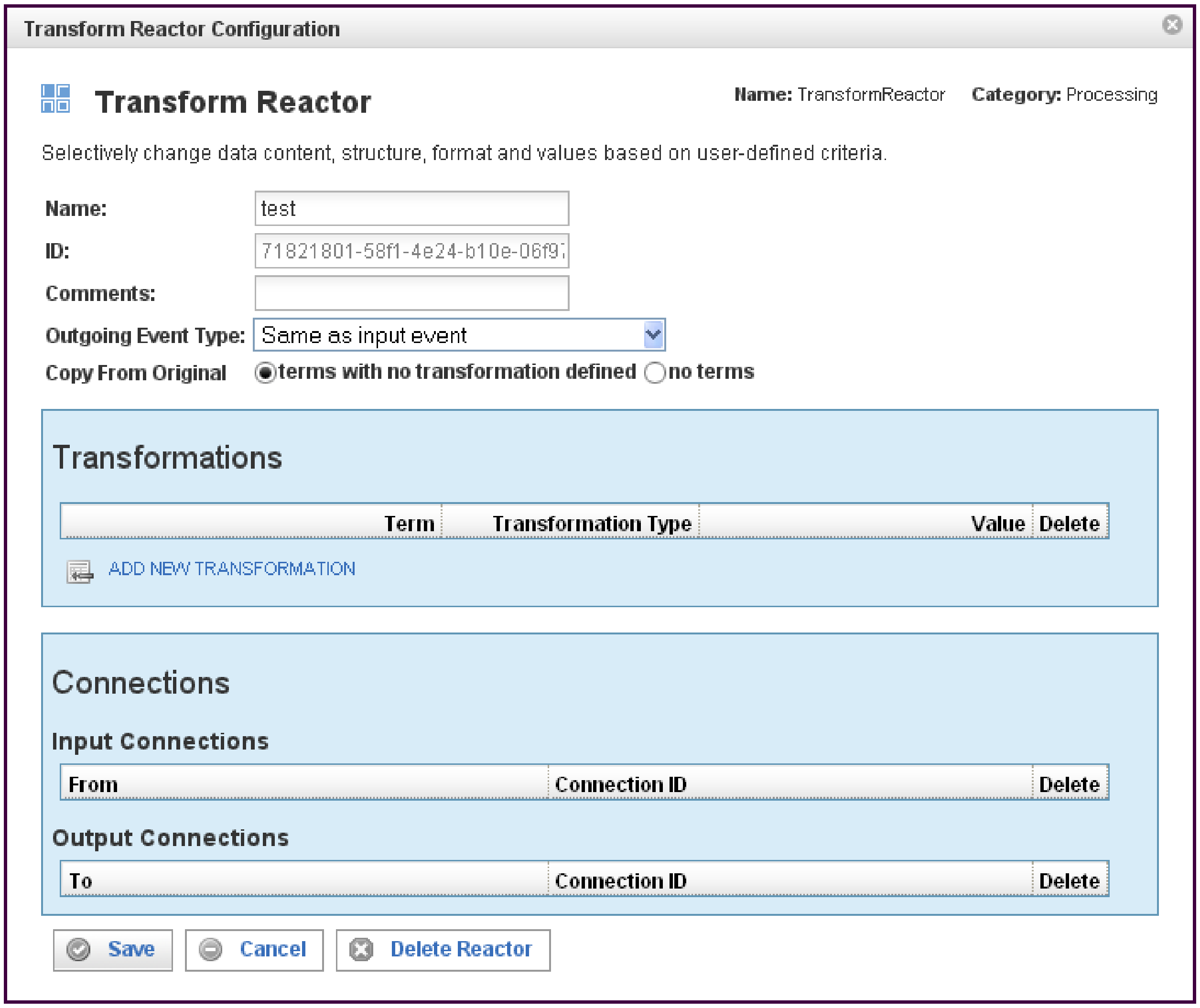
Note: An event processed by the Transform Reactor is transformed. The original (unmodified) event is not passed through. The Transform Reactor's only output is transformed events. If the original (unmodified) event is desired, make an additional connection from the reactor preceding the Transform Reactor to the reactor(s) following the Transform Reactor.

When you add a Transform Reactor you will be prompted for the following fields:



Field	Description
Name	Specify the name of this reactor. This can be any name you find descriptive, we recommend something which describes the data this reactor is processing. The name may be any combination of letters and numbers.
Comments	Enter some descriptive comments about this reactor. Any comment you feel will help you in the future, we recommend choosing something that describes the data being processed.

This creates the Transform Reactor in the workspace. Double-clicking on the reactor brings you to the configuration screen: this reactor gets



Name: TransformReactor Category: Processing

Selectively change data content, structure, format and values based on user-defined criteria.

Name:	test
ID:	71821801-58f1-4e24-b10e-06f97
Comments:	
Outgoing Event Type:	Same as input event
Copy From Original	[✓ terms with no transformation defined] no terms

Term	Transformation Type	Value	Delete

From	Connection ID	Delete

To	Connection ID	Delete

In addition to the fields you saw when adding the reactor initially (described above), you will see some additional fields and sections for *Transformations* and *Connections*. The additional fields are:

Field	**Description**
ID	The reactor ID, a unique ID generated by the Pion server. This ID identifies a particular reactor.
Outgoing Event Type	Event that is output (transformed) may either be of same type as the incoming event, or a user definable type.

Field	Description
Copy From Original	Terms (fields) in the original event are: Copied-if-not-present. Copy all the terms for which there are no transformations. Note that transformations are not always successful, so some terms with transformations may not exist in the output event (example: lookup, with a failure and no default). Not copied. Only terms that are transformed are copied, and if there is no transformation, the term will not exist in the transformed event. Copied in full, and transformations *add* to the copied terms. **Note**: Do not use unless you know how to handle multiple values on terms. This option is now deprecated.

The Transformations Section

The Transformations section describes the transformation processing to be done on the term(s).

Field	Description
Term	Defines the name of the term that will be set to the resulting value of the Transformation Rule. You can select a term (an identified data item) from an existing vocabulary, or you can add a new vocabulary and define the term(s) to be found.
Transformation Type	Depending on the vocabulary and term selected, there are transformation types available: • AssignValue Assigns a constant value to the term defined. Appropriate lexical casting is performed, converting the value to a string, numeric or date value as appropriate. AssignValue will fail to assign only if the cast fails (for example a text string into a numeric value).

Field	Description
	• AssignTerm Assigns a source term (from the original event) to a destination term (resulting event). The value is re-cast, allowing type conversions (text to numeric, or numeric to text). AssignTerm will fail to create the destination term if the source term does not exist, or if the transformation fails to cast. If there are multiple matching source terms, all are copied to destination. • Lookup Applies a regular expression (optional) with a format string (optional, default $&) to a source term (the original event), producing a key. The key is applied (case sensitive) against a table of key/value pairs. If a matching key is found, the value is used. If a key is not found, one of the default actions is taken. If the key is left undefined, the original (pre-regex) value is used with the regex 'p'd (post-regex) value, and a fixed value is used. The result is cast and assigned. Lookup is repeated for each matching source term. • Rules [StopOnFirstMatch] Rules will either execute through all rules in a transformation, or stop on the first successfully matched rule. The source term (original event), is matched against a comparison, which can be any of the normal comparisons, true/false/defined/less-than/is-equal/etc... If the comparison succeeds, the SetValue is assigned like AssignValue. If the comparison fails, the regex is evaluated,

Field	Description
	and SetValue defines the output format. **Note**: Not-RegExp assigns SetValue as a literal, since the regex failed. **Note**: To copy source-term value(s), use RegExp with `".*"` (true will assign a literal value). If source term has multiple values, the rule is applied to each value individually (StopOnFirstMatch does not apply to multiple term values). If a rule was successful (even for one term value), and StopOnFirstMatch was defined, processing of rules stops. Setting StopOnFirstMatch not true will cause many values to be compounded in a destination term. This setting is useful for "adding tags" to a term. **Note**: Use with caution! Assignment of values (SetValue or RegExp/format-output) is re-cast and assigned. **Note**. All rules, all conditions always use the source term from the original event. Multi-step, or progressive processing can only be done with successive separate Transform Reactors. • Regex Uses a source term (must be of string type), and executes a number of regular expressions (with optional format strings). For each regular expression, if the expression fails, the source value (for the expression) is used instead of a transformed value. Expressions are executed for each value in the source term individually, and the resulting strings are copied to the destination term.

Field	Description
	Note: Both Lookup and Rules already have a regex, this is a serial regex. The non-use of StopOnFirstMatch (short-circuit) in Rules is available, and can be used when creating "tag lists" in terms. Do not, however, uncheck it if you don't know how to use the result.
Value	The meaning of this column differs based upon the Transformation Type selected. This allows you to input a value for "AssignValue", select a term for "AssignTerm", and opens additional configuration windows for Lookup, Rules and Regex.

To add a row for a new transformation in the *Transformations* section, click on the ADD NEW transformation link.

To delete a transformation, click on the “x” at the end of the row describing that term.

Regular Expressions in the Transform Reactor

The Transform Reactor uses regex in three different ways:

- In Lookup, regex can be used initially
- In Rules, regex is one of the many options
- In Regex, regex is used in a successive manner

In all cases, the regex being used is the Boost.org regex.

For more information about the basic (Perl) regex by boost, see: http://www.boost.org/doc/libs/1_39_0/libs/regex/doc/html/boost_regex/syntax/perl_syntax.html.

For information on the format string as it is used in the regular expressions, see: http://www.boost.org/doc/libs/1_39_0/libs/regex/doc/html/boost_regex/format/perl_format.html.

The Connections Section

The Connections section describes input and output connections for this Transform Reactor. See “Reactor Input and Output Connections“ on page 6 for more information.

Storage Reactors

Storage Reactors are needed whenever data needs to be kept for later use (for example reporting or record keeping). For web analytics to take place, a Pion workflow must always end with a Storage Reactor.

The Storage Reactors include:

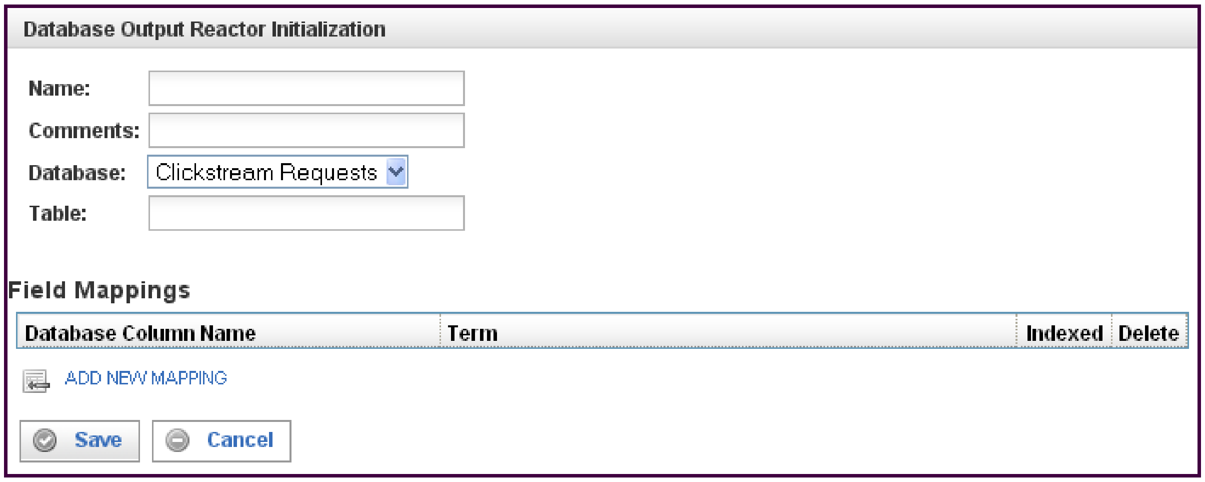
- **Database Output Reactor**, which enables Pion to store information in almost any database.
- **Google Analytics Reactor**, which is able to take web user behavior information and send it to Google Analytics servers via its native web services interface in real-time.
- **Log Output Reactor**, which enables you to write any data to a structured flat file of your choice in real time.
- **Multi Database Reactor**, which enables Pion to store information in several databases (or several tables in a single database).
- **Omniture Analytics Reactor**, which is able to take web user behavior information and send it through the Omniture Analytics integration system to SiteCatalyst via its native web services interface (in real-time) or the XML loading interface.

These reactors are described in the following sections.

Database Output Reactor

The Database Output Reactor enables Pion to store information in almost any database. The Database Output Reactor handles the format translation, table organization and performance tuning aspects required to ensure Pion events are written with the best performance and quality possible. To ensure high performance, native database communication is available for MySQL, Oracle, Microsoft, Sybase, IBM, Informix, Centura, Interbase, PostgreSQL and SQL Lite databases. The Database Output Reactor can also send data to any ODBC compliant database for storage. Because Pion enables direct database output, it can be easily integrated into any business intelligence implementation or data warehouse without requiring custom code or a separate extract, transfer and load (ETL) tool.

When you add a Database Output Reactor you will be prompted for the following fields:


Database Output Reactor Initialization

Name:	
Comments:	
Database:	Clickstream Requests
Table:	

Field Mappings

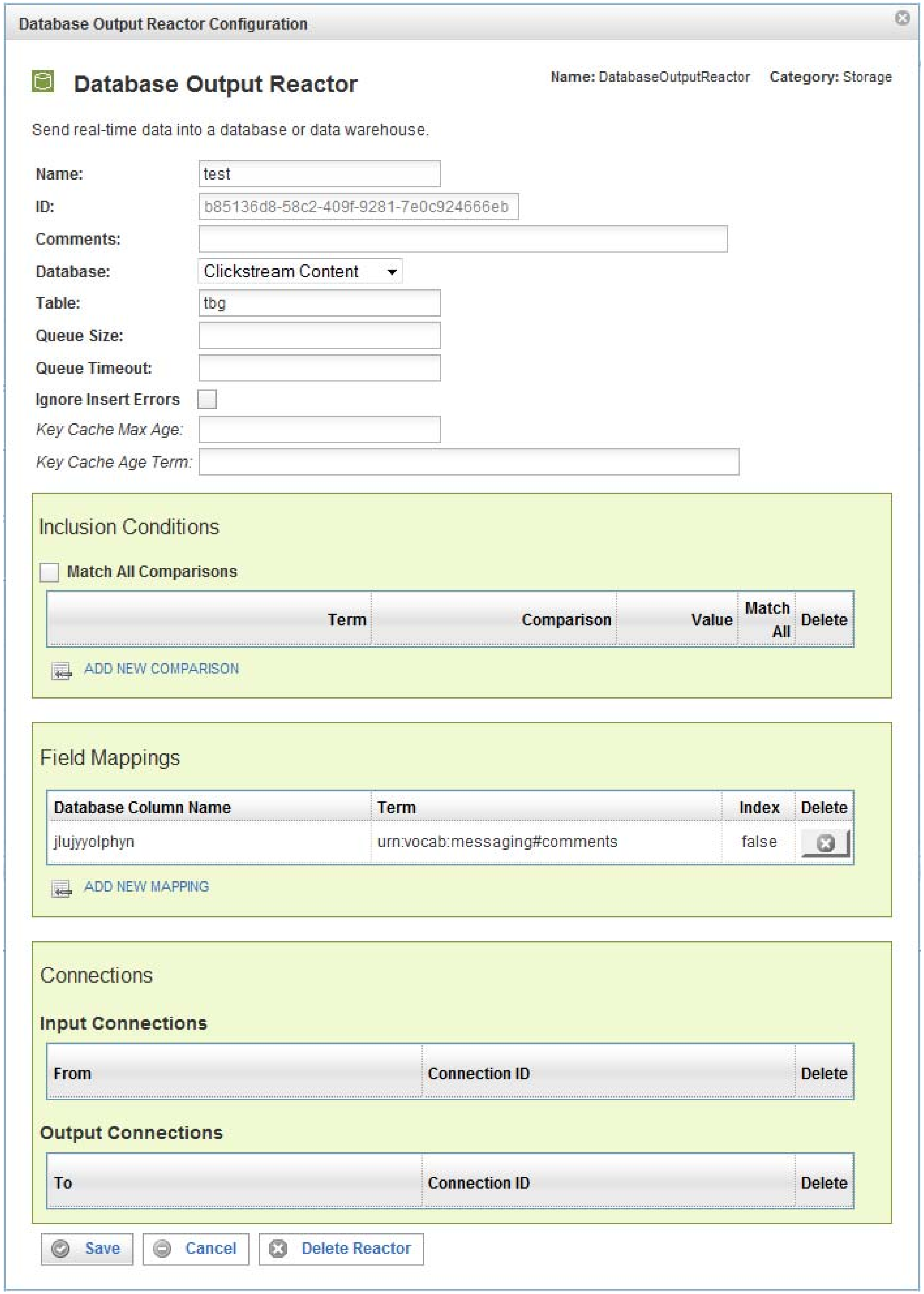
Database Column Name	Term	Indexed	Delete

Field	**Description**
Name	Specify the name of this reactor. This can be any name you find descriptive, we recommend something which describes the data this reactor is storing. The name may be any combination of letters and numbers.
Comments	Enter some descriptive comments about this reactor. Any comment you feel will help you in the future, we recommend choosing something that describes the data being stored.

Field	Description
Database	Select the database into which you want to write events.

Field	Description
Table	Specify the table in the database named above. The events will be written to this table.

This creates the Database Output Reactor in the workspace. Double-clicking on the reactor brings you to the configuration screen:



Database Output Reactor Configuration

Database Output Reactor

Name: DatabaseOutputReactor **Category:** Storage

Send real-time data into a database or data warehouse.

Name:	test
ID:	b85136d8-58c2-409f-9281-7e0c924666eb
Comments:	
Database:	Clickstream Content
Table:	tbg
Queue Size:	
Queue Timeout:	
Ignore Insert Errors	☐
Key Cache Max Age:	
Key Cache Age Term:	

Inclusion Conditions

☐ **Match All Comparisons**

Term	Comparison	Value	Match All	Delete

ADD NEW COMPARISON

Field Mappings

Database Column Name	Term	Index	Delete
jlujyyolphyn	urn:vocab:messaging#comments	false	☒

ADD NEW MAPPING

Connections

Input Connections

From	Connection ID	Delete

Output Connections

To	Connection ID	Delete

In addition to the fields you saw when adding the reactor initially (described above), you will see one additional field and the *Inclusion Conditions*, *Field Mappings*, and *Connections* sections.:

Field	**Description**
ID	The reactor ID, a unique ID generated by the Pion server. This ID identifies a particular reactor.
Queue Size	The number of events allowed in the queue before triggering a bulk insert to the table specified above. The default queue size is 1000 events.
Queue Timeout	The number of seconds to wait (the interval) before triggering a bulk insert to the table specified above. The default queue timeout is 10s (10 seconds).
Ignore Insert Errors (check box)	Inserts may fail if records already exist within the table which have the same unique index values. Fill this check box to ignore these errors.
Key Cache Max Age	Used only if a unique index is defined and this value is non-zero. Pion will maintain a memory-based cache of the most recent unique index values inserted into the database. Items in this cache are removed after the specified number of seconds has elapsed since the previous insert attempt.
Key Cache Age Term	If the key cache is enabled, this must be assigned to the event term corresponding to a timestamp for the epoch (for example `urn:vocab:clickstream#epoch-time`).

The Inclusion Conditions Section

The Inclusion Conditions section is used to specify the data (the terms) that are saved in the database specified. If the conditions are TRUE, the event will be inserted into the database table. If FALSE, the event will be discarded.

Field	Description
Match All Comparisons (check box)	When filled, indicates that ALL of the comparisons defined below must be matched.
Term	The vocabulary term whose value will be stored in the database column for each event inserted.
Comparison	Specifies the type of comparison to test, such as true/false or is-defined/is-not-defined. Depending on the term that is specified in a row, an appropriate list of comparisons will appear in the comparison cell of that row. For example, if it's a date type, you'll get choices like "earlier than", whereas if it's a string type, you'll get choices like "starts with". Like most comparisons, these two examples require a value to compare against, but a few comparisons, like “is defined,” do not require a value to compare against. For more information on comparisons, see “Comparisons“ on page 5.
Value	
Match All (check box)	When filled, indicates that AT LEAST this comparison must be matched.

To delete a connection, click on the “x” at the end of the row describing that connection.

The Field Mappings Section

The Field Mappings section maps event vocabulary terms to database table columns..

Field	Description
Database Column Name	Specifies the name of the database column corresponding to the event term.
Term	The vocabulary term whose value will be stored in the database column for each event inserted.

Index	Controls database indexing settings. You can choose from any of the following: True, which means an index is added to the database table for this field, making searches more efficient while making inserts less efficient. We strongly recommend this setting for all fields that you want to search using the Replay interface. False, which means no index is used for this database field. We recommend this setting for all fields that you will not normally search using the Replay interface. Unique, which means the database field is a unique index for the table.

To delete a field mapping, click on the “x” at the end of the row describing that mapping.

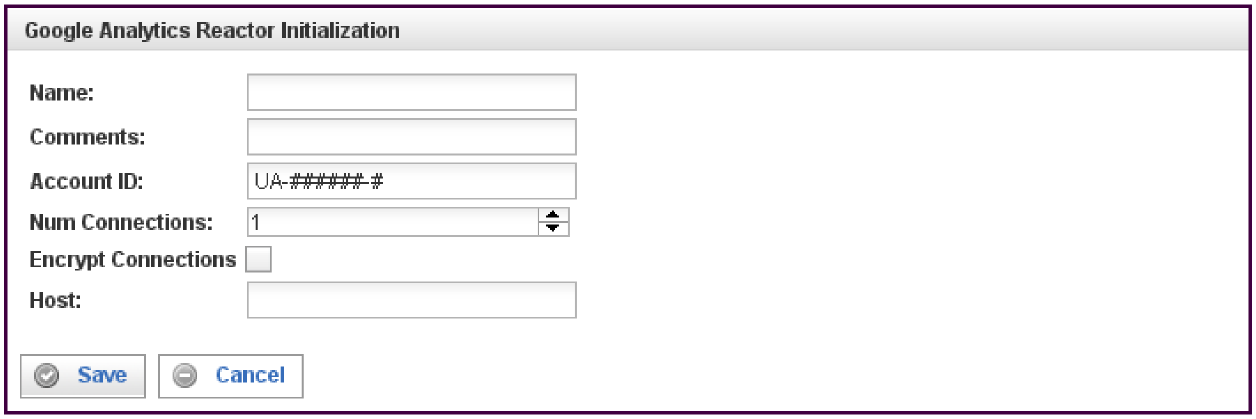
The Connections Section

The Connections section describes input and output connections for this Database Output Reactor. See “Reactor Input and Output Connections“ on page 6 for more information.

Google Analytics Reactor

The Google Analytics Reactor takes web user behavior information and sends it to Google Analytics servers via its native web services interface in real time. From Google Analytics perspective, data collected by Pion is indistinguishable from that collected via page tags. All you need to supply is the correct Google Analytics Account ID and site name to connect Pion to Google Analytics.

When you add a Google Analytics Reactor you will be prompted for the following fields:

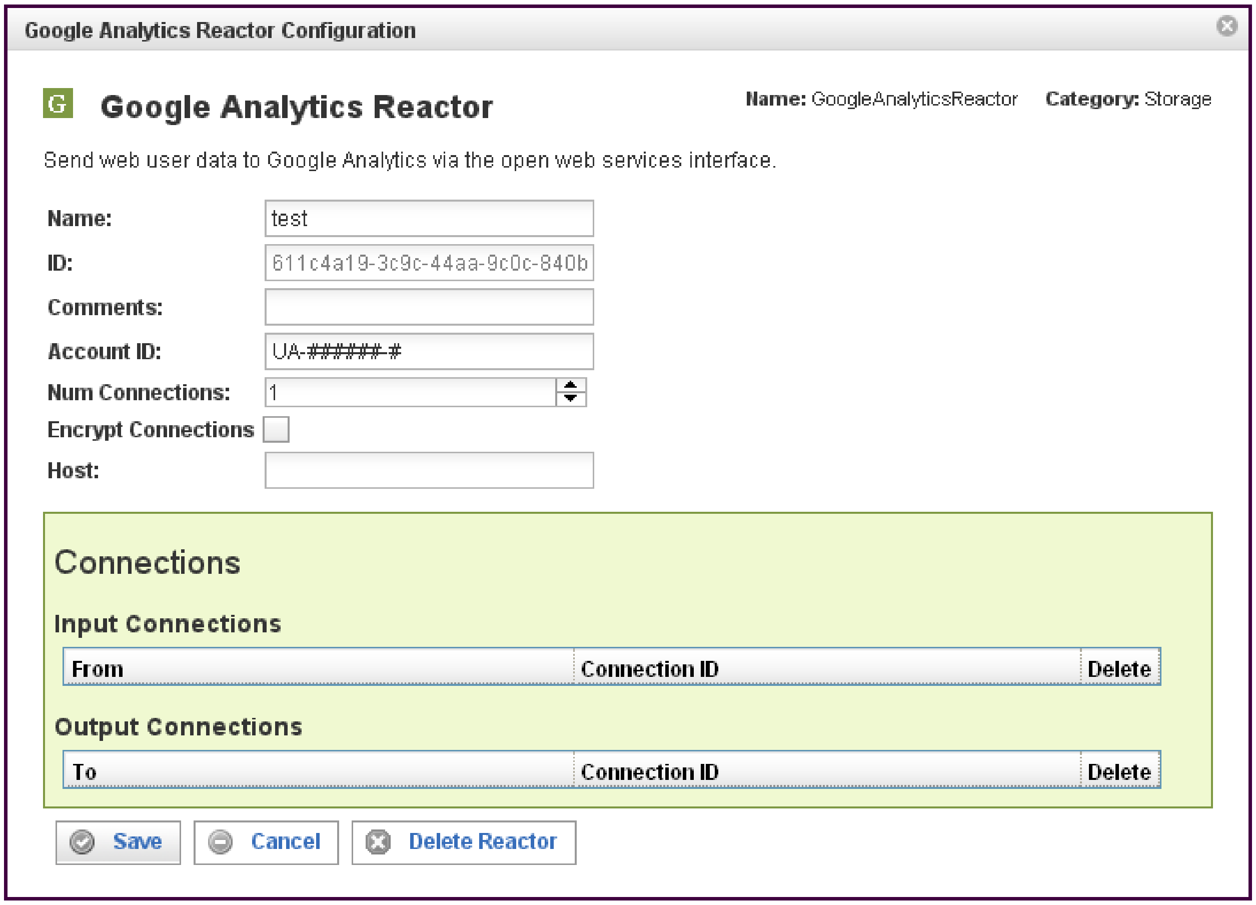


Name:	
Comments:	
Account ID:	UA-######-#
Num Connections:	1
Encrypt Connections	
Host:	

Field	Description
Name	Specify the name of this reactor. This can be any name you find descriptive, we recommend something which describes the data this reactor is processing. The name may be any combination of letters and numbers.
Comments	Enter some descriptive comments about this reactor. Any comment you feel will help you in the future, we recommend choosing something that describes the data being stored.
Account ID	Your Google Analytics account number. This is typically displayed next to each host name for each of your website profiles, and begins with "`UA-`"

Field	Description
Num Connections	Specify the maximum number of concurrent threaded connections that will be established to deliver traffic to the Google Analytics servers. One connection is sufficient for most websites, although sites with a large volume of traffic should increase this number accordingly.
Encrypt Connections (check box)	Fill this check box to encrypt connections between Pion and the Google Analytics servers. Normally this check box is not filled (encryption is not enabled).
Host	This can be used to override the Pion host name string delivered to the Google Analytics servers. Normally left blank.

This creates the Google Analytics Reactor in the workspace. Double-clicking on the reactor brings you to the configuration screen:


Google Analytics Reactor Configuration

Google Analytics Reactor

Name: GoogleAnalyticsReactor Category: Storage

Send web user data to Google Analytics via the open web services interface.

Name:	test
ID:	611c4a19-3c9c-44aa-9c0c-840b
Comments:	
Account ID:	UA-######-#
Num Connections:	1
Encrypt Connections	☐
Host:	

Connections

Input Connections

From	Connection ID	Delete

Output Connections

To	Connection ID	Delete

In addition to the fields you saw when adding the reactor initially (described above), you will see one additional field and the *Connections* section. The additional field is:

Field	**Description**
ID	The reactor ID, a unique ID generated by the Pion server. This ID identifies a particular reactor.

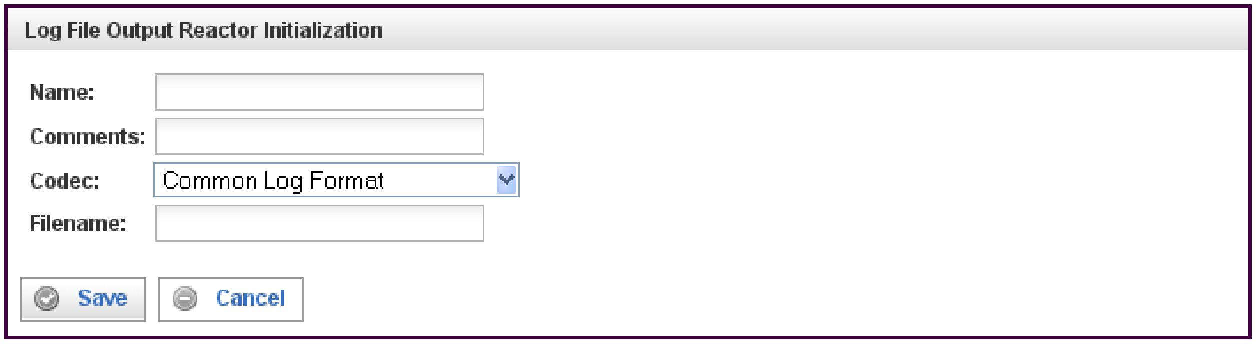
The Connections Section

The Connections section describes input and output connections for this Google Analytics Reactor. See “Reactor Input and Output Connections“ on page 6 for more information.

Log Output Reactor

The Log Output Reactor enables you to write any data to a structured flat file of your choice in real time. This is often very useful for web analytics, because most tools are written to take in web server files as their default input format. Pion can capture, process and enrich user information before creating the distilled log files for use by other tools. This enables Pion to easily integrate with almost any on-site web analytics package including the open source efforts.

When you add a Log Output Reactor you will be prompted for the following fields:


Log File Output Reactor Initialization

Name:

Comments:

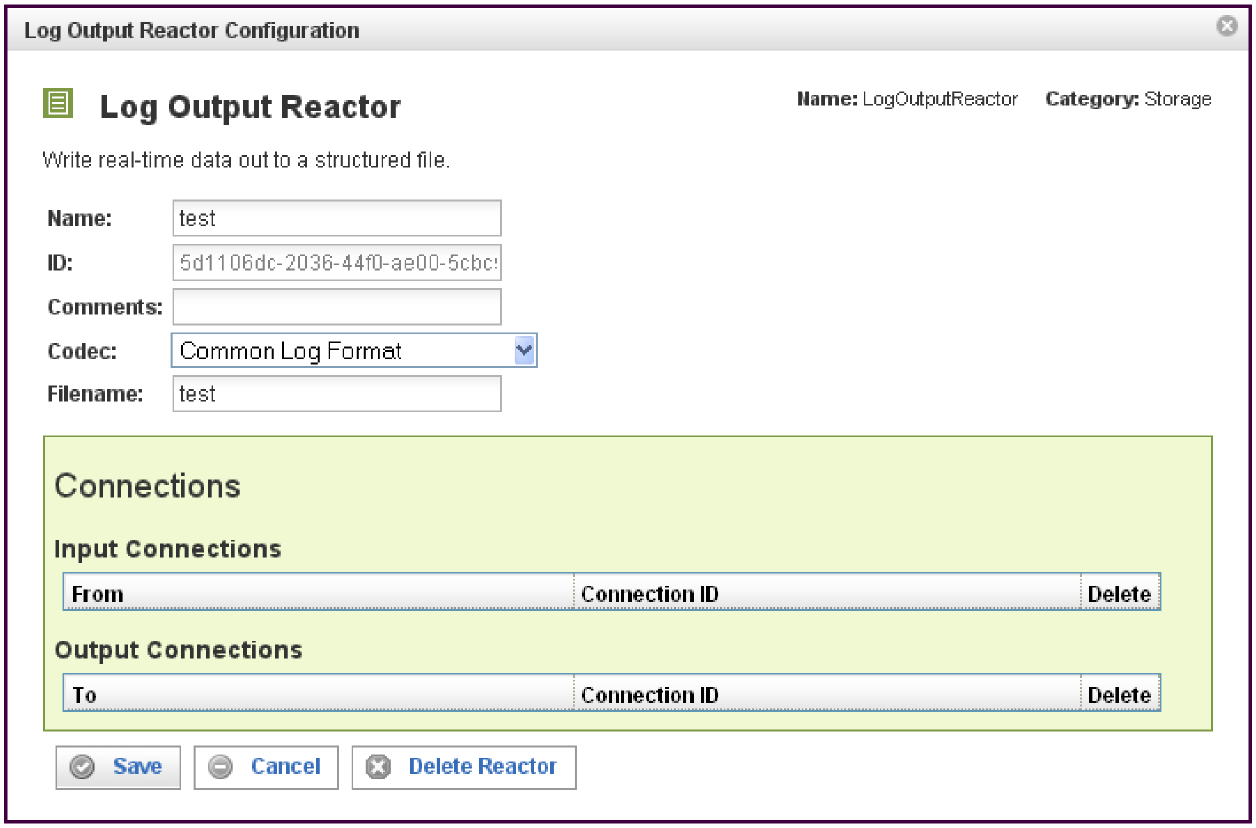
Codec: Common Log Format

Filename:

Field	Description
Name	Specify the name of this reactor. This can be any name you find descriptive, we recommend something which describes the data this reactor is processing. The name may be any combination of letters and numbers.
Comments	Enter some descriptive comments about this reactor. Any comment you feel will help you in the future, we recommend choosing something that describes the data being stored.
Codec	The name of the codec that controls the output format used to write the data to the log file named in filename.

Field	**Description**
Filename	The name of the log file to which the data will be written. This name may be fully qualified, or it may also be relative. When a relative file name is specified, the base path is equal to the "DataDirectory" defined in your platform.xml file. On Unix machines, this is `/var/lib/pion/`.

This creates the Log Output Reactor in the workspace. Double-clicking on the reactor brings you to the configuration screen:



Log Output Reactor Configuration

Log Output Reactor

Name: LogOutputReactor Category: Storage

Write real-time data out to a structured file.

Name:	test
ID:	5d1106dc-2036-44f0-ae00-5cbc
Comments:	
Codec:	Common Log Format
Filename:	test

Connections

Input Connections

From	Connection ID	Delete

Output Connections

To	Connection ID	Delete

In addition to the fields you saw when adding the reactor initially (described above), you will see one additional field and the *Connections* section. The additional field is:

Field	Description
ID	The reactor ID, a unique ID generated by the Pion server. This ID identifies a particular reactor.

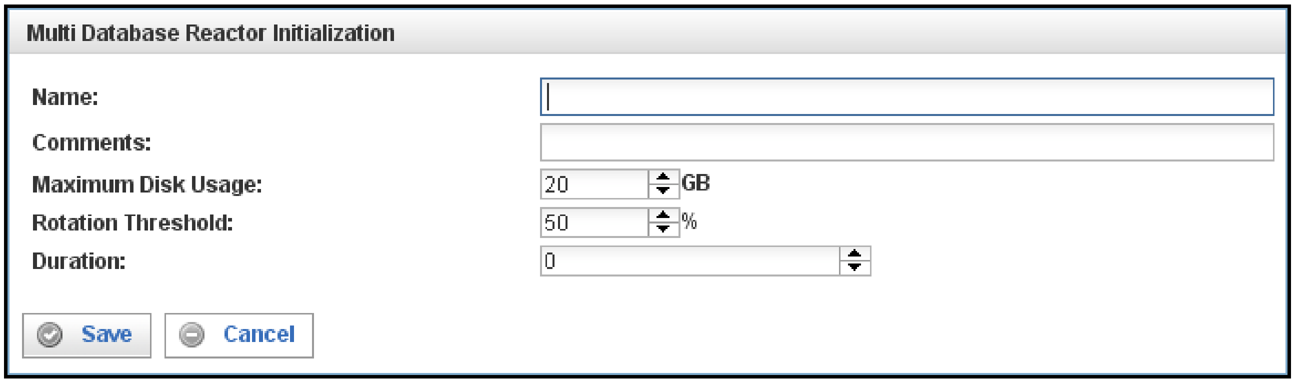
The Connections Section

The Connections section describes input and output connections for this Log Output Reactor. See “Reactor Input and Output Connections“ on page 6 for more information.

Multi Database Reactor

The Multi Database Reactor enables Pion to store data in several databases, based on the data being stored. The Multi Database Reactor stores using rotating partitions, such that the partitions for all tables are divided synchronously based upon sessionizing criteria, size and time.

When you add a Multi Database Reactor you will be prompted for the following fields:



Multi Database Reactor Initialization

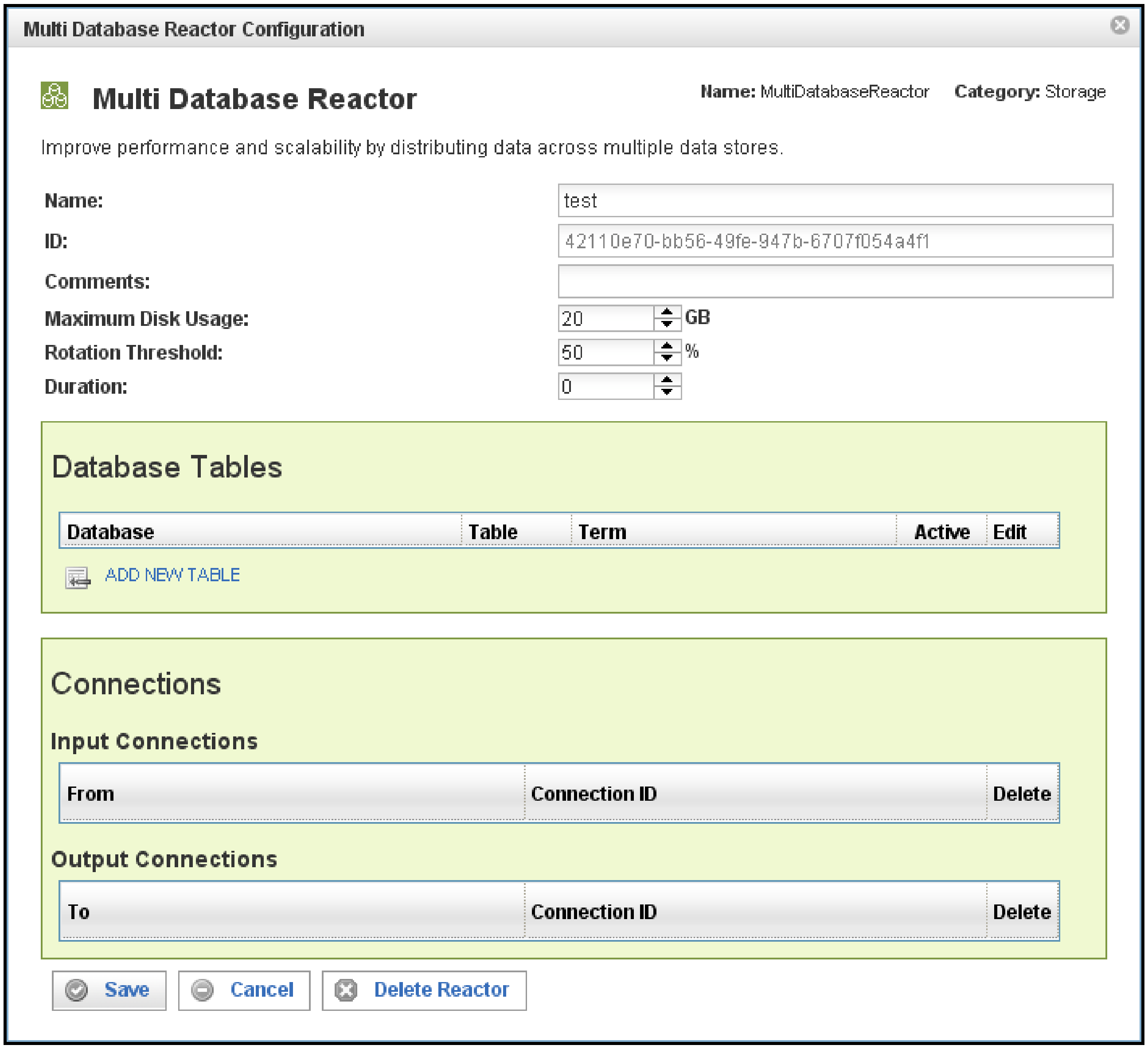
Name:	
Comments:	
Maximum Disk Usage:	20 GB
Rotation Threshold:	50 %
Duration:	0

Field	Description
Name	Specify the name of this reactor. This can be any name you find descriptive, we recommend something which describes the data this reactor is processing. The name may be any combination of letters and numbers.
Comments	Enter some descriptive comments about this reactor. Any comment you feel will help you in the future, we recommend choosing something that describes the data being stored.

Field	Description
Maximum Disk Usage	The number of gigabytes this particular Multi Database Reactor can use to store data. **Note**: Pion makes a best effort attempt to comply with this setting, but you should expect the actual amount of space to to vary ± 10% from the specification. For example, if you specify 50GB, the server can easily use 55 GB, or it may only use 45 GB. **Note also**: If you have two (or more) Multi Database Reactors, **this is an individual, per-reactor setting**. For example, if you have 100 GB of free space, and you have two Multi Database Reactors, you should not give more than 40 GB to each reactor (or however you want to balance it). **Do not over-commit your available storage**. Regarding the size given: • Do not commit more than 80% of your total free disk space. • If you are using local Database Output Reactors or Log Output Reactors, remember to take into account their storage needs.

Field	**Description**
Rotation Threshold	Rotation Threshold is a measure of cache utilization, which is calculated by: 1. The number of columns indexed (in the Multi Database Reactors configuration information) multiplied by size of the content of those columns. 2. Adjusted for the overhead of SQLite. 3. Compared to the configured database index cache size (dbengines.xml) Rotation Threshold is expressed as a percentage: • **For high volumes of data** (400Mbps), you should not exceed 50%. • **For low volumes of data** (<200Mbps), you can probably go 200%. See Error: Reference source not found. on page Error: Reference source not found for information on configuring Pion for high volumes of data.
Duration	The amount of time (in seconds) after which the Multi Database Reactor rotates partitions.
SessionTimeout	Specifies the number of seconds between pruning of the session cache maintained by the Multi Database Reactor. This value should always be just slightly higher than the session timeout defined in your Clickstream Reactor.
Prune Interval	Specifies how often (in seconds) the Multi Database Reactor will prune its session cache.
Session Term	The vocabulary term used to uniquely identify a visitor session (`urn:vocab:clickstream#session-id`).

This creates the Multi Database Reactor in the workspace. Double-clicking on the reactor brings you to the configuration screen:



Multi Database Reactor Configuration

Multi Database Reactor

Name: MultiDatabaseReactor **Category:** Storage

Improve performance and scalability by distributing data across multiple data stores.

Name:	test
ID:	42110e70-bb56-49fe-947b-6707f054a4f1
Comments:	
Maximum Disk Usage:	20 GB
Rotation Threshold:	50 %
Duration:	0

Database Tables

Database	Table	Term	Active	Edit

ADD NEW TABLE

Connections

Input Connections

From	Connection ID	Delete

Output Connections

To	Connection ID	Delete

In addition to the fields you saw when adding the reactor initially (described above), you will see one additional field and the *Connections* section. The additional field is:

Field	**Description**
ID	The reactor ID, a unique ID generated by the Pion server. This ID identifies a particular reactor.

The Database Tables Section

The Database Tables section specifies what data gets stored in which databases. By specifying a database, a table within that database, and a term to be stored in that table, you control what information is stored in which database.

Note: Each Database Table is essentially a separate instance of the Database Output Reactor.

Field	**Description**
Database	Select the database into which you want to write events.
Table	Specify the table in the database named above. The events will be written to this table.
Term	Defines the type of event to be stored in the database and table specified above.
Active	If this check box is filled, the table is active and events of the type specified will be stored in the database and table specified above. If this check box is empty, the table is temporarily disabled and no events will be stored.
Edit	Opens another window to specify the database field mappings and other characteristics for the table.

To delete a connection, click on the “x” at the end of the row describing that connection.

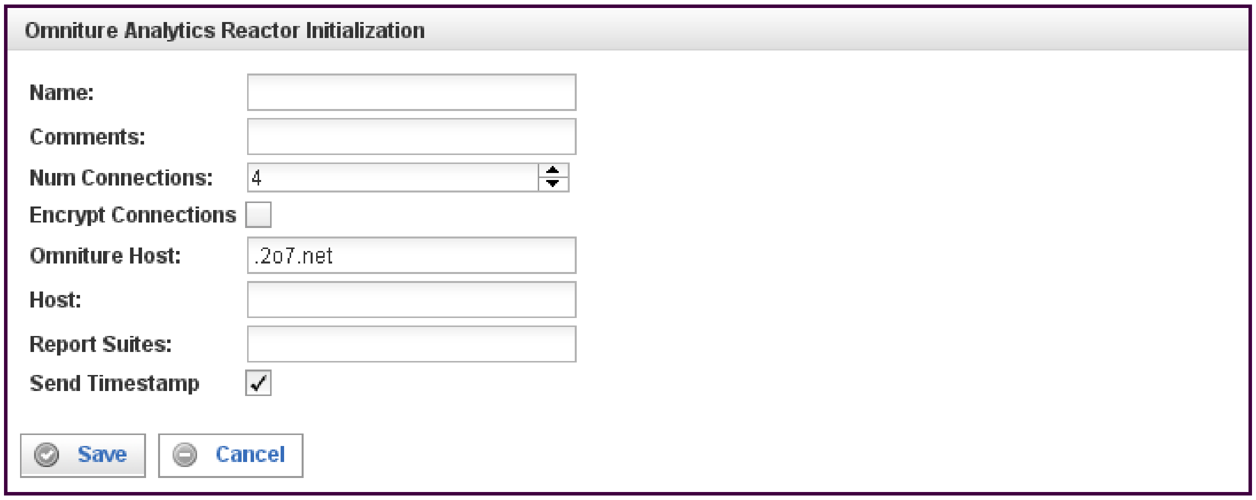
The Connections Section

The Connections section describes input and output connections for this Multi Database Reactor. See “Reactor Input and Output Connections“ on page 6 for more information.

Omniture Analytics Reactor

The Omniture Analytics Reactor is able to take web user behavior information and send it through the Omniture Genesis integration system to SiteCatalyst via its native XML web service interface in real-time. All you need to supply is the relevant account information and site name to connect Pion to Omniture.

When you add a Omniture Analytics Reactor you will be prompted for the following fields:


Omniture Analytics Reactor Initialization

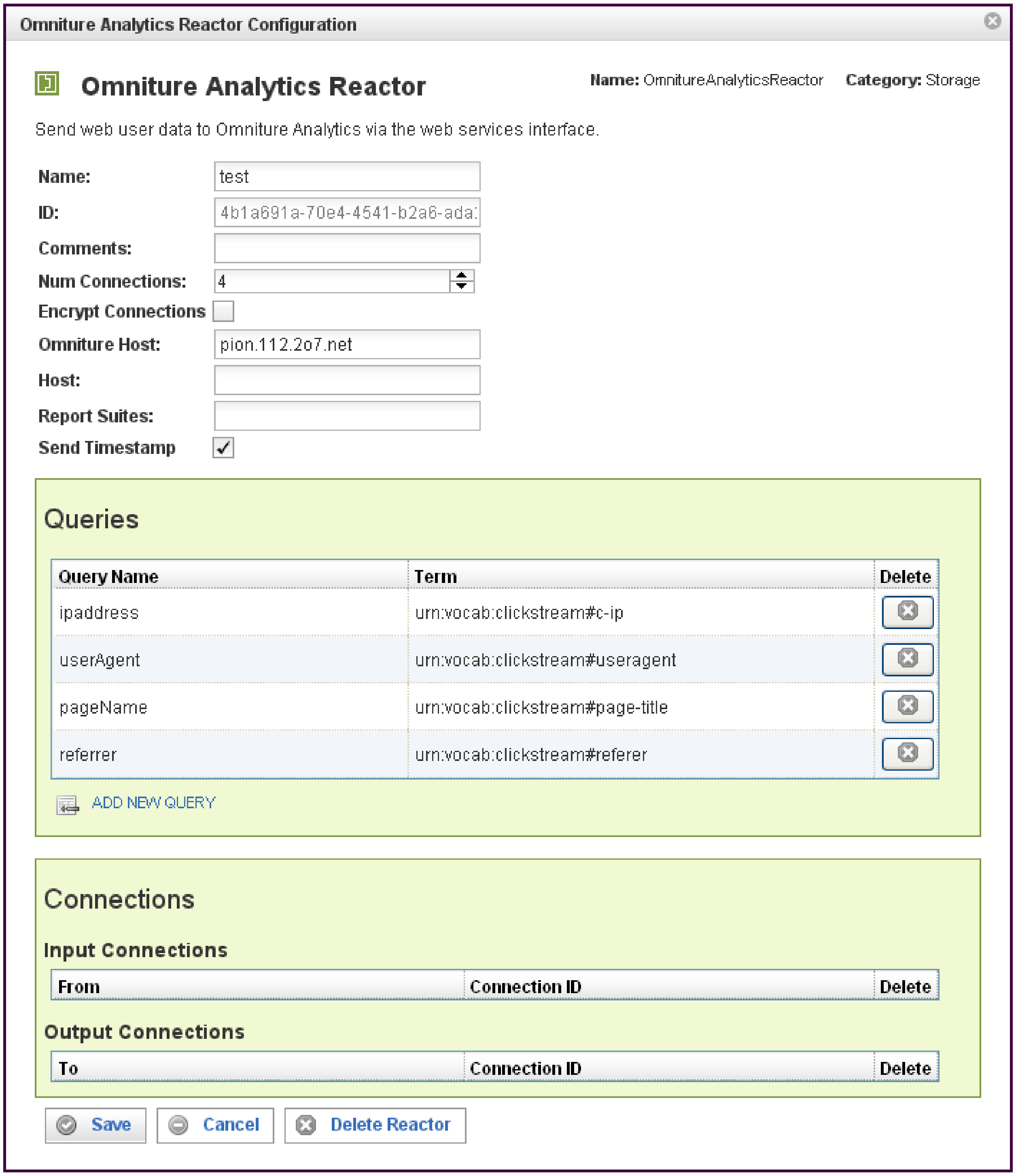
Name:	
Comments:	
Num Connections:	4
Encrypt Connections	☐
Omniture Host:	.2o7.net
Host:	
Report Suites:	
Send Timestamp	✓

Field	Description
Name	Specify the name of this reactor. This can be any name you find descriptive, we recommend something which describes the data this reactor is processing. The name may be any combination of letters and numbers.
Comments	Enter some descriptive comments about this reactor. Any comment you feel will help you in the future, we recommend choosing something that describes the data being stored.

Field	**Description**
Num Connections	Specify the maximum number of concurrent connections. With Omniture; you can deliver 50 events/second with one connection. If you need higher speed (taking bursts/peaks into account), use a higher number of connections. Using a higher number of connections will use more bandwidth, and you may have to consult with Omniture to make sure that they are able to receive the traffic. Generally, a value of 2 or 4 and up to 16 is reasonable.
Encrypt Connections (check box)	Fill this check box to encrypt connections between Pion and the Omniture Genesis host.
Omniture Host	Omniture Analytics Reactor; this is the “Omniture Host” that you get from the Omniture SiteCatalyst [check verbiage from Omniture] as the destination where all the page hits are sent for processing. Must be of the form ***`customername`***`.122.2o7.net` or ***`customername`***`.112.2o7.net.` **Note**: The ‘o’ in 2o7 is the letter, not a zero.
Host	The Omniture Host to which your reporting is to be sent. Specified as a URI, and given by the Omniture "wizard" at the time of registering your account. The Omniture Host URI is of the form accountid.122.2o7.net or accountid.112.2o7.net (do not try to guess it, use what Omniture supplies).
Report Suites	Check with Omniture for description of a Report Suite. In Omniture Analytics Reactor, you would fill in ALL the Report Suites you want the page views to be delivered to. Separate multiple Report Suite names with commas.

Field	Description
Send Timestamp (check box)	For Omniture accounts used with Pion, you must enable time stamps (this check box should be filled). Pion will supply the time stamps for all events (in UTC format), adjusting for the local time zone. If you are experimenting, with an Omniture account not used with Pion, Timestamps must be set off (this check box should be empty). **Note**: Only Omniture technical support can change the Timestamp setting on the Omniture account. **If the Timestamp setting does not match the Omniture back end, all events are silently dropped.**

This creates the Omniture Analytics Reactor in the workspace. Double-clicking on the reactor brings you to the configuration screen:


Omniture Analytics Reactor Configuration

Omniture Analytics Reactor

Name: OmnitureAnalyticsReactor **Category:** Storage

Send web user data to Omniture Analytics via the web services interface.

Name:	test
ID:	4b1a691a-70e4-4541-b2a6-ada:
Comments:	
Num Connections:	4
Encrypt Connections	☐
Omniture Host:	pion.112.2o7.net
Host:	
Report Suites:	
Send Timestamp	✓

Queries

Query Name	Term	Delete
ipaddress	urn:vocab:clickstream#c-ip	⊗
userAgent	urn:vocab:clickstream#useragent	⊗
pageName	urn:vocab:clickstream#page-title	⊗
referrer	urn:vocab:clickstream#referer	⊗

Connections

Input Connections

From	Connection ID	Delete

Output Connections

To	Connection ID	Delete

In addition to the fields you saw when adding the reactor initially (described above), you will see one additional field and the *Queries* and *Connections* sections. The additional field is:

Field	Description
ID	The reactor ID, a unique ID generated by the Pion server. This ID identifies a particular reactor.

The Queries Section

The Queries section allows you match Omniture queries with Pion terms. This matching allows you to send Pion data directly to Omniture Analytics.

Field	Description
Query Name	Select one of the available queries from the list. In previous versions, some query terms were hard coded to “[computed]" and as such were not visible. Now, however, the "[computed]" can be overridden, with a term, but you must be sure to compute the proper value for such term. Generally, this is ***very*special*** usage, most users should not attempt it. **Contact Pion professional services for advice before using this functionality.** The [computed] values are: • visitorID Computed to be a persistent visitor ID, Omniture enforces some strict requirements on what it can be. Do not override unless confirming with Omniture what is allowed. • server Computed to be the "host" configuration parameter. The only time you might want to override this is if you have multiple servers, **and** logic programmed to determine the correct server. • pageURL Computed to be the PageURL, complete with http/https, port (if applicable), URL, and query parameters. • timestamp If applicable, computed to be the timestamp of the event, in ISO-8601 format, adjusted for Zulu timezone. For more information, see: http://en.wikipedia.org/wiki/ISO-8601. • reportSuiteID The same as Account ID, useful only if you are feeding multiple accounts, and have to logic to replace the Account ID All of the parameters above will be XML_Encoded on

Field	Description
Term	Select the term to match with the query.

To add a row for a new query in the *Queries* section, click on the ADD NEW QUERY link.

To delete a query, click on the “x” at the end of the row describing that query.

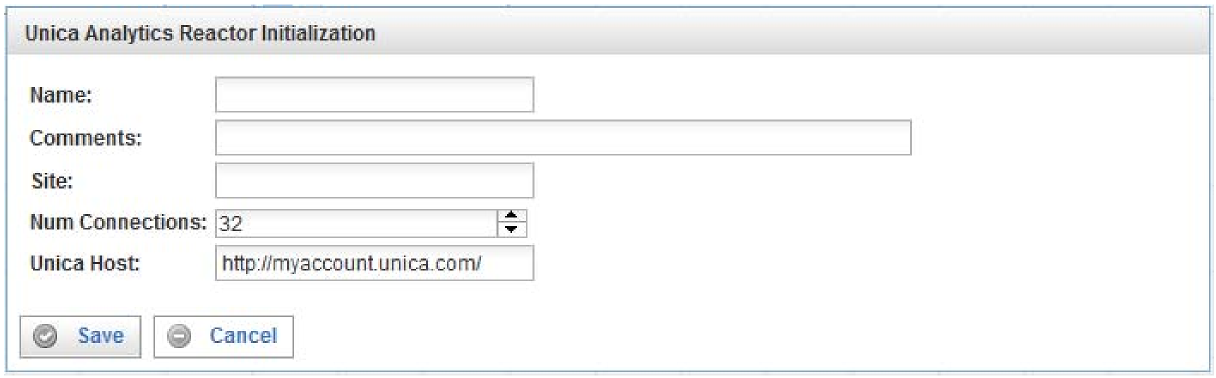
The Connections Section

The Connections section describes input and output connections for this Omniture Analytics Reactor. See “Reactor Input and Output Connections“ on page 6 for more information.

Unica OnDemand Reactor

The Unica OnDemand Reactor is able to take web user behavior information (page views) and send it through the Unica OnDemand marketing management system at Unica.com via HTTP GET QueryParameter interface in real-time. All you need to supply is the relevant site and Unica account name to connect Pion to Unica OnDemand.

When you add a Unica OnDemand Reactor you will be prompted for the following fields:

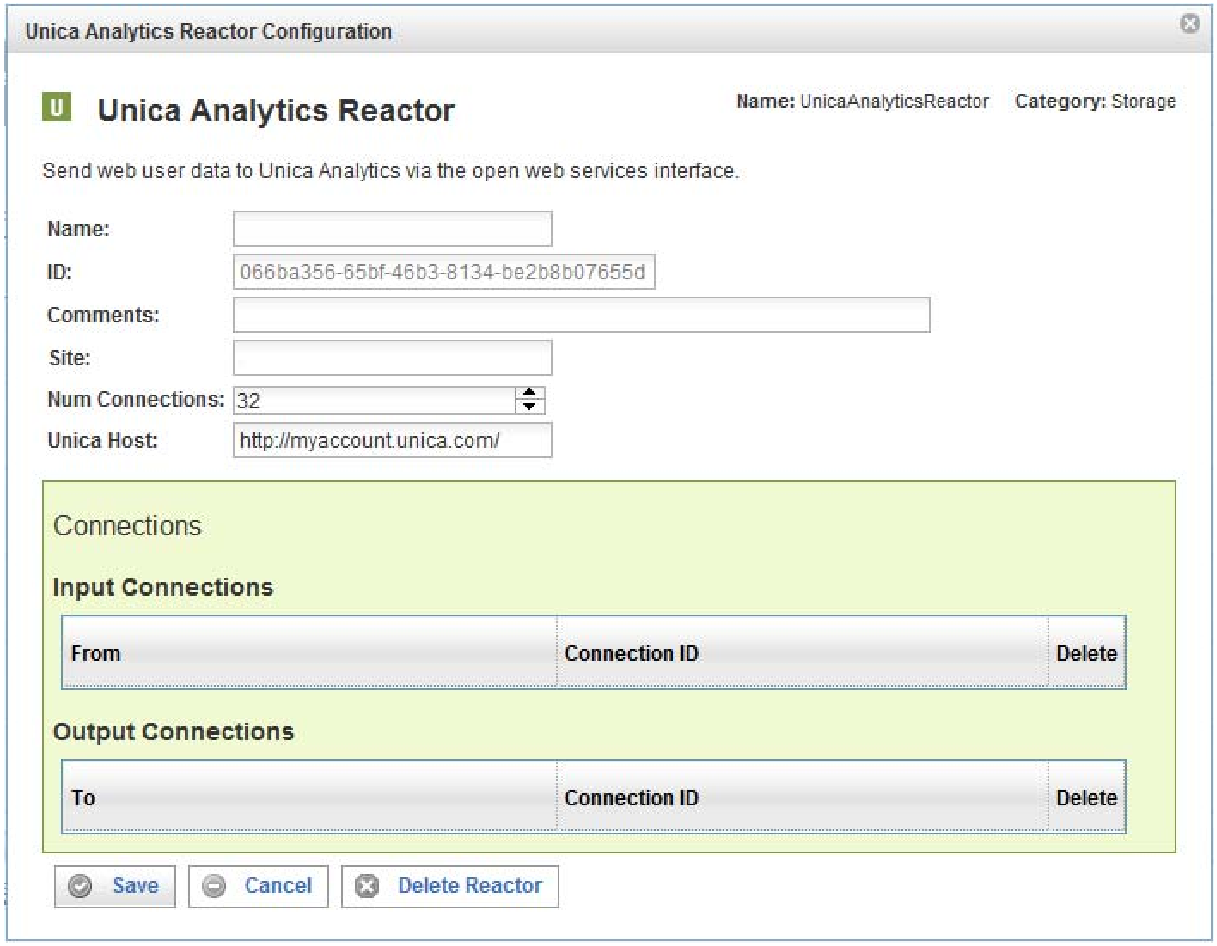

Unica Analytics Reactor Initialization

Name:	
Comments:	
Site:	
Num Connections:	32
Unica Host:	http://myaccount.unica.com/

Field	Description
Name	Specify the name of this reactor. This can be any name you find descriptive, we recommend something which describes the data this reactor is processing. The name may be any combination of letters and numbers.
Comments	Enter some descriptive comments about this reactor. Any comment you feel will help you in the future, we recommend choosing something that describes the data being stored.
Site	The name of the site being monitored.
Num Connections	Specify the maximum number of concurrent connections. If you need more than a single simultaneous connection to deliver events to Unica, increase the number of connections.

Field	**Description**
Unica Host	This is the Unica OnDemand host that you connect to (you get the account information from Unica for this host). This host is the destination where all the page hits are sent for processing. The Unica Host is usually specified in the form ***`accountname`***`.unica.com`.

This creates the Unica OnDemand Reactor in the workspace. Double-clicking on the reactor brings you to the configuration screen:


Unica Analytics Reactor Configuration

Unica Analytics Reactor

Name: UnicaAnalyticsReactor Category: Storage

Send web user data to Unica Analytics via the open web services interface.

Name:	
ID:	066ba356-65bf-46b3-8134-be2b8b07655d
Comments:	
Site:	
Num Connections:	32
Unica Host:	http://myaccount.unica.com/

Connections

Input Connections

From	Connection ID	Delete

Output Connections

To	Connection ID	Delete

In addition to the fields you saw when adding the reactor initially (described above), you will see one additional field and the *Connections* sections. The additional field is:

Field	Description
ID	The reactor ID, a unique ID generated by the Pion server. This ID identifies a particular reactor.

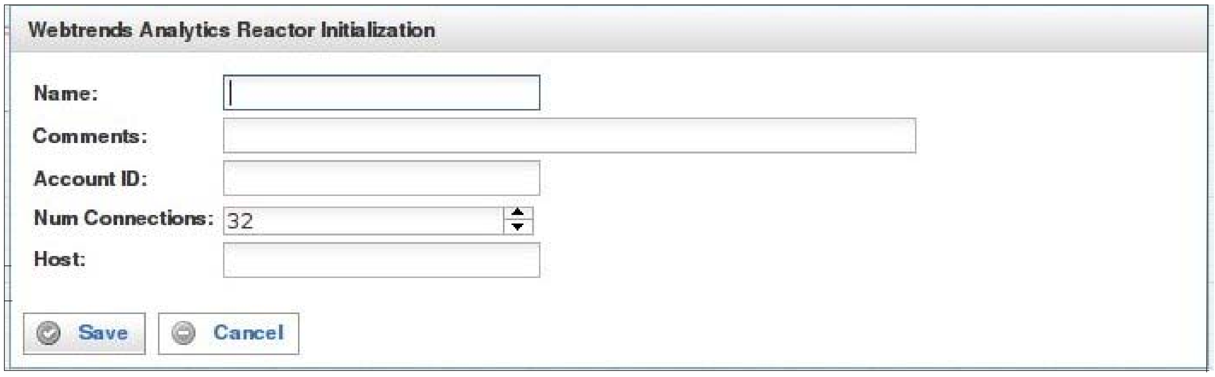
The Connections Section

The Connections section describes input and output connections for this Unica OnDemand Reactor. See “Reactor Input and Output Connections“ on page 6 for more information.

Webtrends Analytics Reactor

The Webtrends Analytics Reactor takes web user behavior information and sends it to Webtrends' servers via its native web services interface in real time. From Webtrends perspective, data collected by Pion is indistinguishable from that collected via page tags. All you need to supply is the correct Webtrends Account ID and site name to connect Pion to Webtrends.

When you add a Webtrends Reactor you will be prompted for the following fields:



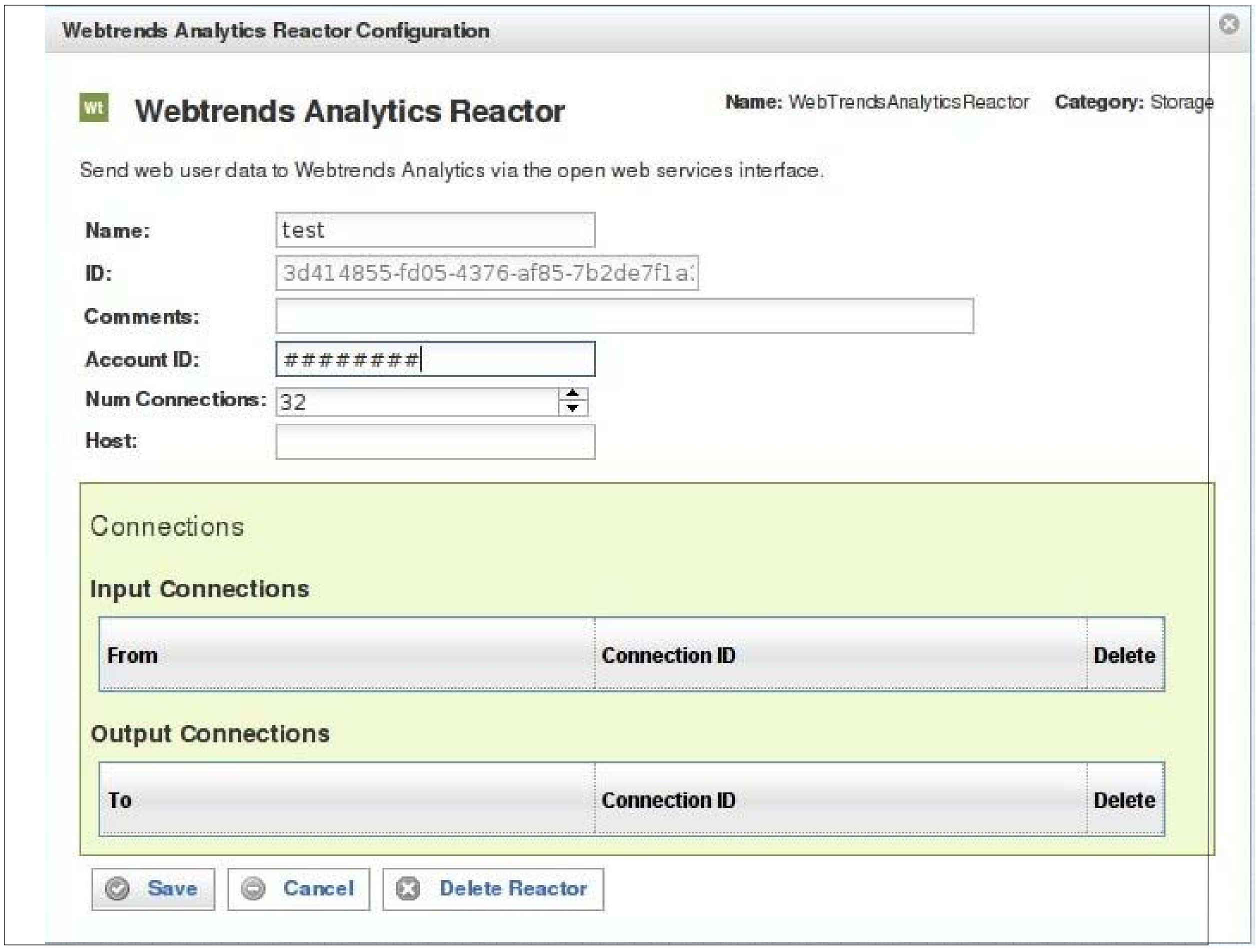
Webtrends Analytics Reactor Initialization

Name:	
Comments:	
Account ID:	
Num Connections:	32
Host:	

Field	Description
Name	Specify the name of this reactor. This can be any name you find descriptive, we recommend something which describes the data this reactor is processing. The name may be any combination of letters and numbers.
Comments	Enter some descriptive comments about this reactor. Any comment you feel will help you in the future, we recommend choosing something that describes the data being stored.
Account ID	Your Webtrends DCSID . This is typically displayed next to each host name for each of your website profiles.

Field	**Description**
Num Connections	Specify the maximum number of concurrent threaded connections that will be established to deliver traffic to the Webtrends servers. One connection is sufficient for most websites, although sites with a large volume of traffic should increase this number accordingly.
Host	This can be used to override the Pion host name string delivered to the Webtrends servers. Normally left blank.

This creates the Webtrends Reactor in the workspace. Double-clicking on the reactor brings you to the configuration screen:


Webtrends Analytics Reactor Configuration

Webtrends Analytics Reactor

Name: WebTrendsAnalyticsReactor Category: Storage

Send web user data to Webtrends Analytics via the open web services interface.

Name:	test
ID:	3d414855-fd05-4376-af85-7b2de7f1a
Comments:	
Account ID:	########
Num Connections:	32
Host:	

Connections

Input Connections

From	Connection ID	Delete

Output Connections

To	Connection ID	Delete

In addition to the fields you saw when adding the reactor initially (described above), you will see one additional field and the *Connections* section. The additional field is:

Field	Description
ID	The reactor ID, a unique ID generated by the Pion server. This ID identifies a particular reactor.

The Connections Section

The Connections section describes input and output connections for this Webtrends Reactor. See “Reactor Input and Output Connections“ on page 6 for more information.

Appendices:

Appendix A: Specifics about Databases

MySQL

MySQL Database is somewhat different from other databases, in that MySQL has multiple engines (referring to MySQL 5.1.x and most other versions). Each engine has specific features, so MySQL is like 11 different databases in one. Pion supports all of the engines, with the two most common: MyISAM and InnoDB configured readily. The four engines most useful for Pion, and their notable features are outlined below.

- **MyISAM** is the fastest, and the simplest engine, and is the recommended engine to use with Pion. MyISAM does not support transactions, but when Pion is inserting data, transactions are not needed -- so you can gain substantial speed (which is crucial if Pion is inserting large amounts of data). MyISAM supports advanced features like bulk insert, and delayed inserts, speeding operations even further.

- **InnoDB** is a comprehensive, transactional database. You pay a heavy penalty for the transactions. Use this engine ONLY if you are interfacing with an application that requires the InnoDB engine to be used.

- **Archive** is an append-only database, it does not support DELETE or UPDATE. This engine is good for logging (immutable), it compresses data automatically, and provides fast queries.

- **MEMORY** is an in-memory database, and is very fast. This engine is very good for lookups (as may be done using the SQLReactor). When using this engine always load MEMORY based tables at MySQL startup, and be sure to allocate sufficient memory for the memory tables. The engine is defined in the CREATE TABLE syntax, at the end with "ENGINE=MYISAM" -- see dbengines.xml for examples.

Getting Started

Create a database named piondatabase using the "mysql" command line tool, or the graphical administration tool on the database server:

```
CREATE DATABASE piondatabase;
```

For a remote MySLQ installation create a user "pion" with a password, and grant rights to Pion to create tables:

```
GRANT USAGE ON piondatabase.* TO 'pion'@'pionserverip' IDENTIFIED BY
'pionpassword';
where:
*Pionserverip is the IP address of the Pion server
*pionpassword is the password for the user 'pion'
```

For a local MySQL installation, create a user “pion” with a password, and grant rights to the Pion to create tables:

```
GRANT USAGE ON piondatabase.* TO 'pion'@'localhost' IDENTIFIED BY
'pionpassword';
```

where *pionpassword* is the password for the user 'pion'.

Creating a Database Connection from Pion to MySQL:

- Use "@" to connects to a local server (very fast connection)
- Use "@databasename" to connect to database “databasename” on a local server
- Use "servername@databasename" to connect to database “databasename” on a remote server
- The “servername” is the DNS name of the server
- If other than port 3306 is being used, specify “servername:port”

Pion server must have the MySQL client library installed, such as mysqclient14 or later, and often this is already installed. On RedHat/CentOS systems, install "MySQL-client-community" or "mysqlclient14" using yum.

Useful Extensions

Note, that MySQL supports certain extensions to SQL, and these extensions particularly useful with Pion.

- REPLACE (instead of INSERT), this is effectively an INSERT or UPDATE depending on whether a record already exists
- INSERT DELAYED, which bulks inserts together, and gives priority to queries
- INSERT IGNORE, which attempts to INSERT a record, but fails silently if the record is already present (this method is faster than testing for record, and then inserting the record)

For more examples, and for an understanding on how to change behavior or add new engines, see the dbengines.xml configuration file in your config directory.

Oracle

Here you will find information on how to use Oracle with the Database Output Reactor and the SQL Reactor. SQL Lite is embedded in Pion, but Oracle is not, this means that you must have your own instance of Oracle. Your Oracle database must be fully installed and you should be able to run PL/SQL from a client.

Limits

Oracle XE edition has a **database size of limit of 2GB**

Pion supports **Blobs** with **Oracle 10g**. Pion does not support Blobs in versions of Oracle prior to 10g.

Since Oracle has a length **limitation of 30 characters** for any identifier (table, column, INDEX), and since Pion tries to make unique identifiers (using tablename_columnname_idx), this limits the combined Oracle table name and column name length to a maximum of 25 characters. For example in a table named "my_long_name" (12 characters) no column should exceed 13 characters (25 - 12 = 13).

If you cannot reduce the table name length, and you are NOT using any indexed columns, you may add "ORA-00972" to the dbengines.xml file, under the "Oracle" section, in "CreateStat" and "CreateLog" sections, causing Pion to ignore the "identifier is too long" errors.

Getting Started

Create an account on Oracle, for example use: "pion" password: "pion" This account must have permissions to CREATE and DROP tables and indexes.

Installing the client on the Pion server

Install the Oracle client or instant client software on the Pion server. You should have received it with the database. Or, you can download it from Oracle's web site. Go to http://www.oracle.com/technology/index.htm search for "instant client".

Note that your client library version must match your database version.

For example, install **oracle-xe-client_10.2.0.1-1.0_i386.deb** if using a **32 bit client on Debian (Ubuntu) client**.

On an **RHEL, 64 bit** environment you should install: **oracle-instantclient-basic-10.2.0.4-1.x86_64.rpm**

Linking the Driver

- The Oracle installation seems to leave out linking the driver, you may have to edit your:
  - **/etc/ld.so.conf** file to include the path, where Oracle installed the client library, and run ldconfig
- In the Ubuntu example, this path:
  - **/usr/lib/oracle/xe/app/oracle/product/10.2.0/client/lib** must be added to ld.so.conf
- In the RHEL/64:
  - **/usr/lib/oracle/xe/app/oracle/product/10.2.0/client/lib** needs to be added to ld.so.conf

Configuring Pion to use an Oracle database connection

You must first create a "new database" in Pion. When creating the database, choose the “Enterprise Database.”

For the connection string you would specify: hostname: 1521/XEXDB

Where:

- hostname is the IP number or the hostname of your Oracle server
- 1521 is the port, required only if not using the default port (1521)
- XEXDB is the service name, in Oracle EXpress it is "XEXDB" (for normal Oracle installations, it s ofter”orcl”
- Check with your DBA for the proper service name you can find the service name in TNSNAMES.ORA file. Once a database is configured, you can use it in a reactor, such as DatabaseOutputReactor or SQLReactor

When Pion starts the Database Output Reactor, it creates the table with appropriate schema, adding indexes as configured.

Pion generates and removes indexes automatically from tables that it controls (via Database Output Reactor or Multiple Database Reactor).

Subsequent starts of Database Output Reactor will not create the table (if it already exists), but every time Database Output Reactor starts, it will verify, and possibly create/drop the indexes as configured.

Notice on table and column names, that Oracle imposes certain limitations. Consult with your DBA.

Note that if you add columns, after the table has been created, the new columns will **not** be automatically added to the table.

In order to avoid loss of data, Pion does not drop or alter tables. You must either shut down Pion or stop the reactor using the database and then, using Oracle, drop or rename the table. This allows Pion to re-created the table on the next start. If necessary, use the ALTER TABLE syntax in Oracle to change the table schema (not recommended).

Errors

If you get an error message about `“libclntsh.so missing”` it means that Pion can not find the Oracle client library; perhaps its location is not specified in /etc/ld.so.conf, or ldconfig has not been run.

If you get: `ORA-12541 "TNS Listener does not exist",` it means, that the Service name (described above with XEXDB and orcl examples) is not configured correctly. Contact your DBA, and find out the proper service name to use.

- **Pion is configured to ignore "**`ORA-00955" and "ORA-01418" errors` **when creating/dropping tables and indexes. You can configure additional constraints in the Oracle section of the dbengines.xml file.**