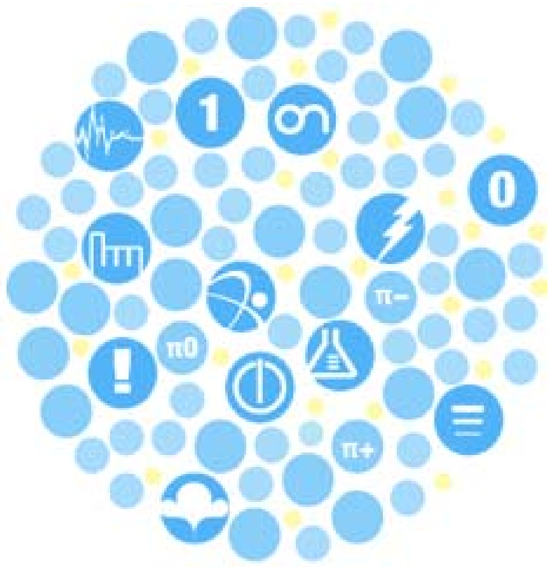


Enterprise Edition

Part I: Overview

Last update: July 2009

Table of Contents

Pion: A Quick Introduction

Pion: What Is It?

Pion is a robust, highly configurable, extremely fast data analytics application, including a server tuned specifically for data acquisition and transformation. The flexibility of Pion's building-blocks architecture allows you to gather data from a wide variety of sources, filter out the parts you don't need, and then keep the useful information. Additionally, Pion allows you to transform the data (or parts of the data) according to rules you specify, and you can store the data for compliance purposes or later review.

Pion has specialized building blocks (called reactors) that you chain together to form a chain that gathers, and transforms data from the sources you choose, and then stores the parts you want to keep.

Pion is passive, in the sense that it does not alter the original data in any way. Pion monitors live data streams or reads log files, then transforms the data as necessary, and finally stores the results or the transformed data, but the original data is not changed in any way.

Pion is standards-based, in that it works and uses with standard protocols, log and data formats, processing libraries, and APIs.

The Pion application is browser-based; and is controlled through a Web Browser. All standard web browsers are supported, but due to implementation differences within the browsers, some Pion features may look different in one browser than they do in another. Pion recommends Mozilla Firefox or Apple Safari for all platforms (note that Google Chrome is not supported).

Pion also provides an XML-based web services API that allows you to do everything the browser-based application does. Documentation for this API is available at our web site:

http://pion.org/projects/pion-platform/documentation/services

To help you understand more about Pion, it will be necessary to define some Pion-specific terms and concepts:

Reactors and Events

A Pion Reactor performs a defined task, and there are many specialized Reactors. Each reactor performs a specific task, and each reactor is specialized for a particular task. A Reactor can be thought of as atomic in the sense in that it is the smallest unit of work. Reactors are the basic building blocks which are chained together to accomplish tasks.

To make organization easy the Reactors are broken down into three basic families by their functions:

- **Collection** Reactors receive some sort of input (either through monitoring a data stream, an application log, or from a script or another Reactor) and create **Events** when defined criteria are met.
- **Processing** Reactors process and/or change Events. Processing Reactors may interact with other systems, for example the SQL Reactor interacts with a SQL Database, such as MySQL.
- **Storage** Reactors handle the output of one or more Processing Reactors. Depending on the Storage Reactor, the processed Event may be stored in a database or log file, or it may be sent to another application (such as Google Analytics or Omniture Genesis).

  Storage Reactors are needed whenever data needs to be persisted for later use for reporting or record keeping. For web analytics to take place, a Pion **Workflow** must always end with a Storage Reactor.

Reactors of different types are chained to form a complete processing path, starting with data acquisition, and ending with some form of output. For example, when reading a web server log (or even when capturing/monitoring a web server's traffic):

a) An Collection Reactor detects an an instance of that server generating a 404 (page not found) error. The Collection Reactor creates an event and sends it to a Processing Reactor.

b) The Processing Reactor receives the Event, and strips out everything but the ID of the server that generated the error, the bad URL, the time, the date, and the session ID of the user. The Processing Reactor then passes this Event to another Processing Reactor and to a logging Storage Reactor.

c) The second Processing Reactor compares the server ID to a list of server ID and email address pairs, and inserts the email address at the beginning of the Event before sending the Event to an alerting Storage Reactor.

d) After receiving the Event from a Processing Reactor, the first Storage Reactor records the information in a log file, and the second Storage Reactor sends an email to the email address in the Event.

In this way, the Reactors receive data, create and process Events, and output the results.

Pion provides an open source platform users or third parties can also create their own Reactors if they need additional functionality without having to modify the core of Pion in any way.

Codec

A Pion Codec defines an input or output format for things like logs, for example CLF (Common Log Format) or ELF (Extended Log Format). A Codec can also define notation such as JSON or XML. Codecs are used by Reactors to 'break' data into chunks that can be recognized or manipulated.

Vocabulary

A Pion vocabulary is a list of terms (and types, such as shortstring, string, longstring, date, uint8, uint16) that form the possible elements of information contained in an event. The most common vocabulary is the Clickstream vocabulary; which is used to describe the Terms in an web page view or HTTP event. Examples of other vocabularies are RSS, Atom, stock prices, and aggregate (see Vocabulary Configuration on page 17, for more information on vocabularies). Users can create new vocabularies, or modify and extend existing vocabularies to meet their needs.

Workspace

A Pion workspace is a graphical presentation of a group of Pion reactors chained together. Pion allows for multiple workspaces, each of which can contain a self-sustained, parallel-running set of reactors. Pion workspaces are browser-based; that is, thee Pion application is controlled through a Web Browser.

All standard web browsers are supported, but due to implementation differences within the browsers, some Pion features may look different in one browser than they do in another. Pion recommends Mozilla Firefox or Apple Safari for all platforms (note that Google Chrome is not supported).

Other terms and concepts that are important to Pion, but are not Pion-specific. Some are specific to other applications (such as Omniture Genesis or Google Analytics). These terms and concepts are used throughout Pion, and you should know what they mean in order to make sure you understand what Pion does and how to use Pion's features and the related technologies.

Regex

A “regex” or “regexp” (REGular Expression) is a common method of describing a text pattern for filtering, matching or transformation purposes.

Queue Size

Generally used to describe optional Queuing (or grouping) of items. For databases, Queue Size defines the maximum number of events to queue before writing them (in bulk) to a database.

Queue Timeout

Defines the amount of idle time before a queue is written (in bulk) to a database.

Pion: What Will It Do For Me?

Pion will provide you with better information for less effort then you are putting in today with traditional page tags, data warehouses and ETL tools. Pion does this through:

- Capturing the complete conversation between your users and the web backend servers.
  - Identify and track individual users across visits.
  - Sessionize clickstreams to establish unique behavioral patters.
  - Extract critical information and events from the conversation (Totals, Products, Promotions, etc)
  - Handle non-browser interactions from RSS and ATOM news services.
  - Capture Mobile Phone browsers.
  - Eliminate the need for most page tags (90%+ reduction).
- Combining real-time and historical information.
  - Collect information from relational databases (Oracle, DB2, MS-SQL, MySQL, Informix, Sybase and more).
  - Pull in customer information from ERP and CRM systems.
  - Connect to back end log files.
  - Provide an merged stream of data from different sources.
- Freeing your data through integration.
  - Feed your existing web analytics solution from Omniture, WebTrends, Google Analytics and others.
  - Store data in your data warehouse (Oracle, DB2, MS-SQL, MySQL, Teradata and others).
  - Feed your existing business intelligence tools (Cognos, Business Objects and others).
  - Provide web services streams to real-time applications.
- Replaying customer interactions.
  - Provide a visual, page by page representation of what the user saw.
  - Expose detailed session information about the user and their environment.
  - Report on the performance of the end user experience.
  - Provide detailed information on the user submission and server response.

Pion: What's New in this Release?

In this release, several changes have been made in Pion.

- Reactor changes
- New Reactors; the Fission Reactor, the Multi Database Reactor, and the Content Hash Reactor
- The Replay Feature

Reactor Changes

Some of our Reactors have been changed slightly (See the Reference for more information).

New Reactors

Two new Reactors provide new functionality in Pion:

- The **Fission Reactor**, which splits input events into sequences of output events. See page 15 for more information about this new 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. See page 15 for more information about this new Reactor.
- The **Multi Database Reactor** enables Pion to store data in several databases, based on the data being stored.

The Replay Feature

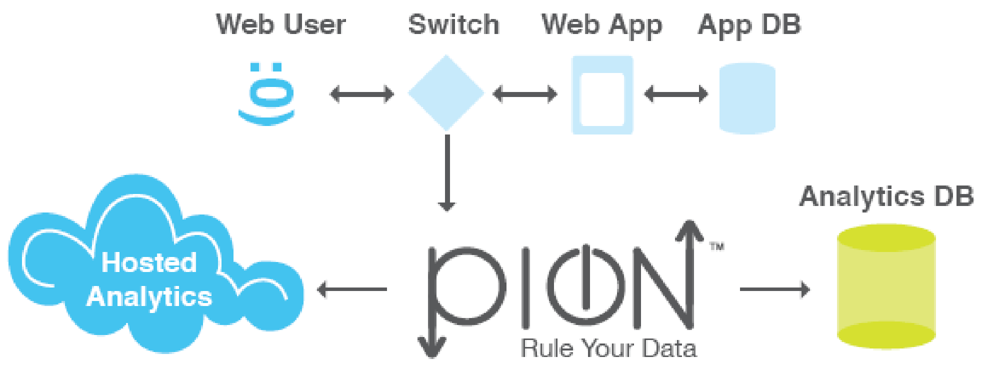
The Replay feature allows you to see what a visitor to your web site saw and "step through" their experience on your web site. The Replay feature captures the page views seen by a user, and **allows you to see the pages the y saw, as the user saw them**! Using Replay, you can also display information about the user's session, such as the client IP address, date and time, number of pages viewed, and more.

- Uses the Content Storage Reactor and a database to store the content (pages).
- User can search sessions from "Sessions," "Pages," or "Requests" using configurable queries.
- Session contains multiple page views, which can be viewed in a browser ("you're seeing what the end-user saw").
- Currently (in 2.0) the only supported database for REPLAY is the embedded SQLite -- in subsequent versions, additional databases will be supported

Pion: Architecture

Pion uses a very straight-forward architecture design based on the ability for it to listen to the conversation between the user's web browsers and the back end servers. When properly placed in the network, Pion is able to both capture the live browser conversation and listen for any active content calls from the browser for purely client-side content (such as videos, music, or files). Capturing data from web log files is easily accomplished from almost any network location with remote file access.

A basic conceptual architectural diagram looks like this:



As you can see in the above diagram, Pion is completely passive and in no way impacts the user's web experience. This is accomplished by using either network tapping or spanning to simply take a copy of the conversation between the web clients and the servers (the data stream). The very nature of a tap/span is a one way conversation; meaning Pion never communicates back with your web site visitors browsers. This ensures that the highest levels of both security and performance are delivered to your web site visitors, because Pion does not interact with the clients browser at all. After the Pion server captures and interprets the user traffic, it communicates with your analytics database. In the case of hosted providers such as Omniture Genesis and Google Analytics, this communication occurs via a web service. For internal analytics packages, this communication is most often performed via a relational database output Reactor or a structured file Reactor.

The critical pieces of this architecture are:

1. Pion Server
2. Pion Sniffer Reactor placement in your network
3. Network communication channels

These, and other topics that affect Pion's operation, are covered in greater detail below.

The Pion Server

The Pion software takes full advantage of the latest commodity hardware available, as well as multiple operating systems. Pion currently supports the following operating systems:

- Linux x86– RedHat EL 4, 5 (32 and 64 bit), CentOS 4,5 (32 and 64 bit)
- Windows – XP, Vista, Windows 2000, Windows 2003 and Windows 2008 (32 bit x86)
- MacOS X – 10.5 Intel (32 bit x86)

Pion will also run on VM Ware, but for product network traffic capture we suggest dedicated hardware. Because VM Ware is known to occasionally drop packets, if you plan on using a Sniffer Reactor to monitor network data, you should not run the Pion server on a VM Ware installation.

Although Pion will run on very minimal hardware it is important to size it appropriately for production applications. The most critical components are CPU and RAM. A good rule of thumb is a 2GHz CPU core and 2GB of RAM are generally able to handle about 100Mbps worth of HTTP(S) traffic with a normal configuration. Because Pion is highly multi-threaded, and it makes very effective use of multiple CPU cores, a single server can scale to handle approximately 1.2 Gbps worth of HTTP(S) traffic.

For sniffing purposes, Pion also needs high quality network interface cards, and we have had great experience with the Intel Pro cards. They aren't an absolute requirement, but we have seen Broadcom and 3com cards dropping packets for no apparent reason from time to time. It is also important to ensure that you have an extra network interface for server management in addition to the number of ports you'll need available for sniffing traffic.

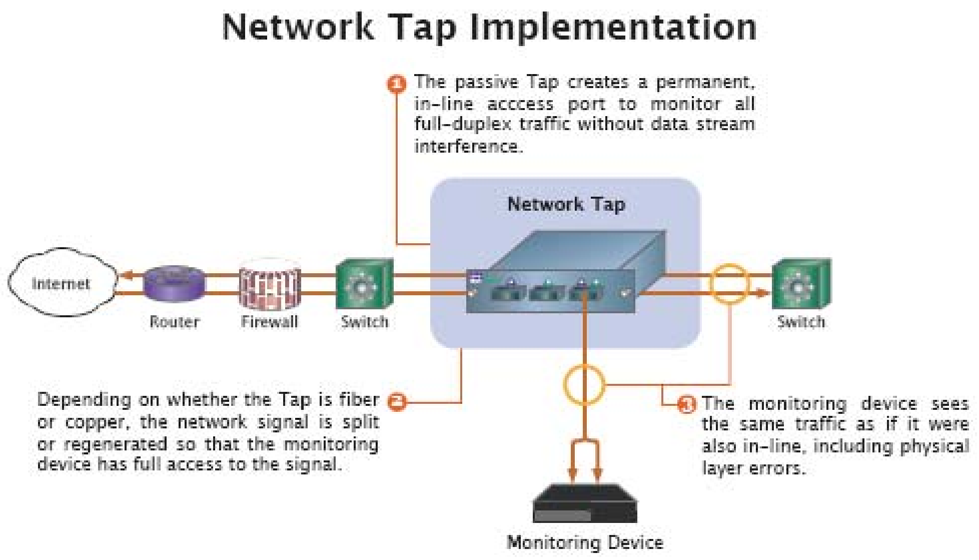
Pion Sniffer Placement

The first step in designing the architecture is to determine where the best place is to capture your network data. Ideally Pion would only capture the traffic you want it to see and nothing else. Therefore, the best place to place to tap/span is often very close to the web servers. This allows us to minimize both the amount of "garbage" traffic we see, and reduce the amount of network listening we need to do. Fortunately, most networks are hierarchical in their design. The hierarchy is usually made up of "server" switches (sometime referred to as "server farms", "farm switches", "edge switches" or similar) and "aggregation switches". As their names suggest "server" switches connect servers to switches while "aggregation" ones connect "server" switches to each other. Because of this it is often possible to monitor even very large data centers with one or two network listening points.

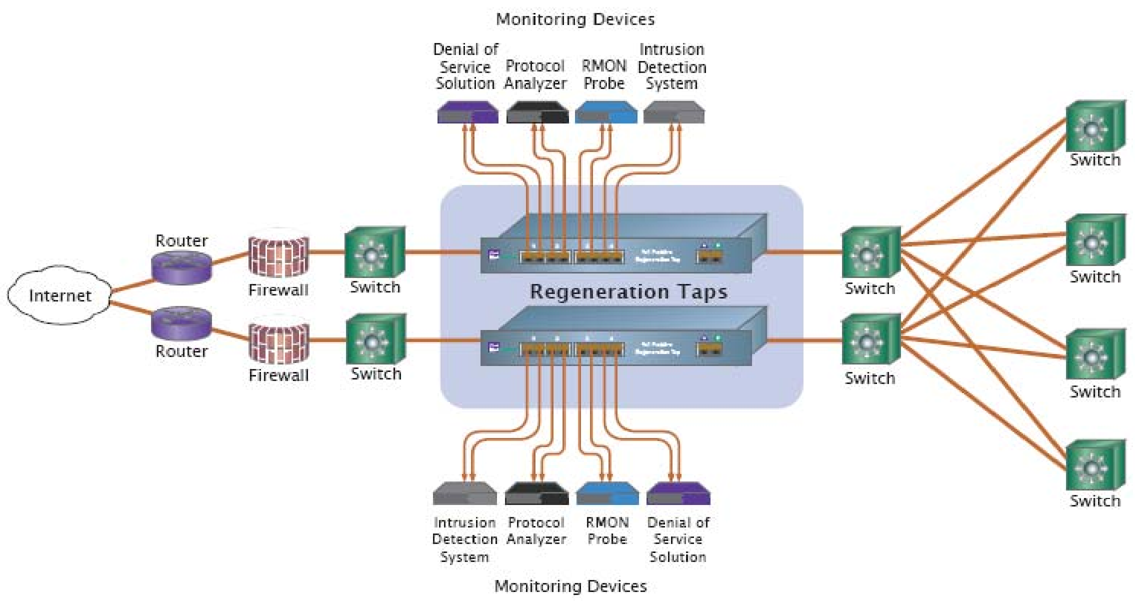
Using Network Taps and Spans

In order for the Pion to passively capture user traffic it must receive a duplicate stream of traffic to analyze. There are 2 methods to implement this:

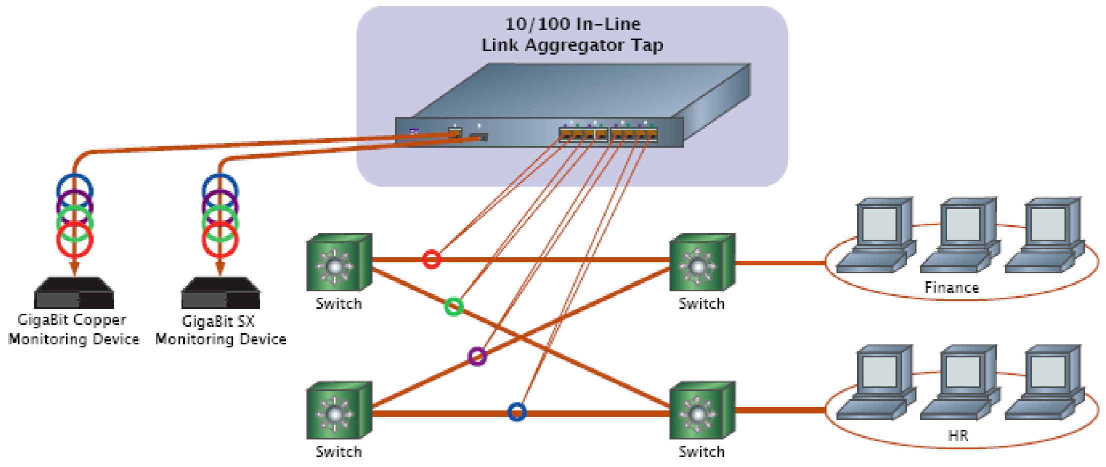
- **Spanning/mirroring**, which is implemented by configuring the switch to duplicate a stream of traffic from one port(s) to another. You connect one of the sniffing NIC(s) on the Pion server to the physical port(s) on the switch to which the duplicate stream is sent.
- **Network taps**, which fall into several categories depending on their configuration.
    - **Basic Network Taps**, which are device with typically one set of inputs (transmit and receive) to one set of outputs (transmit and receive) similar to your standard stereo splitter as shown:


Network Tap Implementation
1 The passive Tap creates a permanent, in-line acccess port to monitor all full-duplex traffic without data stream interference.
Network Tap
Internet
Router
Firewall
Switch
Switch
2 Depending on whether the Tap is fiber or copper, the network signal is split or regenerated so that the monitoring device has full access to the signal.
3 The monitoring device sees the same traffic as if it were also in-line, including physical layer errors.
Monitoring Device

- **Regeneration Taps**, which have a set of inputs (transmit and receive) and are able to send the same output to multiple transmit and receive ports allowing multiple devices to evaluate the same traffic as shown below:



- **Aggregation Taps**, which are able to take in multiple sources of traffic and combine that traffic so that it can be consumed for one monitoring device as illustrated in the picture below:



Spanning/Mirroring vs. Taps Comparison

Factor	Span/Mirror	Tap
Description	Using the software in network device to copy traffic from one or many ports to another	An "in stream" device that passively copies traffic to another source, like a stereo splitter

Type	Software configuration	Hardware device
Cost	None. Implemented in software on the switch	Varies by vendor and type of device but usually low
Scalability	Limited. Each switch has a limited number that can be implemented	Unlimited
Flexibility	May allow you to filter the traffic	None. All traffic is duplicated
Quality	Under high loads switch may drop packets	Always a clear signal and copy
Installation	On-the-fly, no downtime	Requires cabling and may also require switch downtime

Network Communication Channels

Pion must be able to communicate in order to be effective. To manage Pion you'll want to have at least one secure channel to the web server or terminal.

Management Channels	Port	Protocol	Direction
Secure command line management	22	SSH	User → Pion
Web administration	Any, default is 8888	HTTP(S)	User → Pion
Windows Terminal Services	Any, default is 3389	RDP	User → Pion

Pion must also have network access to write the data you want to the desired location. This can occur over any port you define. The common ones and their defaults are listed below.

Data Output Channels	Port	Protocol	Direction
Omniture Genesis Web Services	80/443	HTTP(S)	Pion → Omniture
Google Analytics Web Services	80/443	HTTP(S)	Pion → Google
Oracle Database Server	Any, default is 1521	OCI	Pion → DB
Microsoft SQL Server	Any, default is 1433	DB-Lib, OLE-DB	Pion → DB
Sybase	Any	Open Client, ASE or ASA	Pion → DB
DB2	Any, default is 50000 or	DB2 CLI	Pion → DB

Data Output Channels	Port	Protocol	Direction
	446		
MySQL Enterprise	Any, default is 3306	MySQL CAPI	Pion → DB
ODBC	Any	ODBC	Pion → DB
Informix	Any, default is 1526	Informix CLI	Pion → DB
Interbase/Firebird	Any	Native	Pion → DB
Centura/Gupta SQLBase	Any	CAPI	Pion → DB
PostgreSQL	Any	Libpq	Pion → DB
SQLite	Local	Local	Pion → DB

Other Network Considerations

Working with Load Balancers

Load balancers are very common in enterprise environments. In most cases they present a complication when it comes to Sniffer Reactor placement because they perform what is known as Network Address Translation or NAT. When a load balancer receives a request from a user, it determines which server will receive the request, then opens a connection to this server and forwards the request, replacing the user's IP address with its own. This results in a situation where users always see the load balancer as the destination and servers always see the load balancer as the source. Based on whether the sniffer is listening to traffic "before" the load balancer (between the client and the load balancer) or "after" the load balancer (between the load balancer and the servers), Pion may lose its ability to link the client to the specific server if this situation is not addressed. Two easy solutions exist to handle load balancers.

1. Using the X-Forwarded-For (XFF) tag. This solution is very simple, and often requires little or no effort because it is built into almost all modern load balancers and is set to work by default. The X-Forwarded-For (XFF) tag is the unofficial standard for identifying the originating IP address of a client connecting to a web server through an HTTP proxy. Most load balancers have it turned on by default, and those that don't often allow you to enable XFF easily. By default, if the XFF is used, it is interpreted by Pion as the original IP tag. This means that it is very likely that you won't have to do anything. You can read more about XFF on http://en.wikipedia.org/wiki/X-Forwarded-For and http://meta.wikimedia.org/wiki/XFF_project.
2. Having a Pion Reactor extract client session information directly. This solution requires a little more effort, but is more flexible. Since Pion sees all of the traffic, it doesn't have to rely on the network data and Pion can extract the content from the user session. Therefore, Pion can easily identify and track people by user name, host name, session ID, cookies or anything else that is communicated with the server.

Working with Proxy Servers

Proxy servers, while very different in function than load balancers, have similar implications due to the fact that they also implement NAT. There are a few additional issue to contend with when dealing with proxy servers.

1. Unlike load balancers, which only re-write the host address, proxy servers often re-write the URL (or more accurately the URI). If this is the case, Pion may need to be configured to reconstruct the original URI with the Transform Reactor.
2. Proxy servers may implement an additional caching mechanism that doesn’t follow normal browser caching rules. Rather than retrieve all resources from the server for each user request they cache resources (typically static resources such as graphics, entire static pages, rich media, etc.) locally. This means that you may have to place the Pion sniffer before the proxy server to ensure you get all of the calls.

Proxy servers are not traditionally considered part of the network architecture. Therefore, they are not always present on network diagrams. You should find out if they are part of the network environment.

Pion Security Elements

Since Pion captures real user behavior by listening to the conversation that occurs between your web users (customers) and the infrastructure you use to deliver their online experience, it is reasonable to be concerned about what happens to this conversation. Pion takes great pains at every step of the way to ensure the security of the information it touches using a variety of techniques:

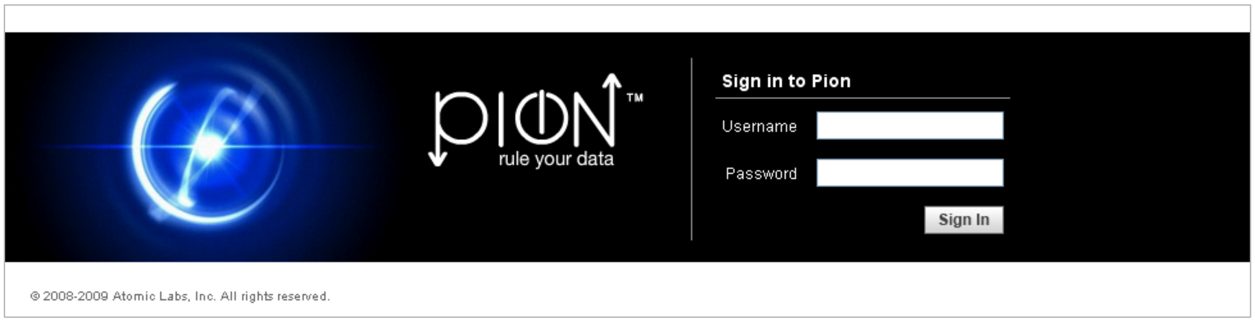
- **One-way communication for the Pion Sniffer Reactor**. The Pion Sniffer Reactor cannot communicate with anything other than the Pion server application under any circumstances. It is completely passive and is only fed one-way traffic. Because of this, Pion is more secure than any of the web servers it is listening to because it cannot communicate.
- **Extra security layers for SSL keys**. Pion often needs to decrypt information on the fly for interpretation and as such requires the critical private SSL keys. These sensitive keys can be kept securely under additional layers of encryption and password protection to ensure safety on top of the network layer security previously mentioned.
- **Filtering/masking of sensitive information**. Once the user and network data is inside of Pion, you can have Pion filter out or translate sensitive fields at the point of collection to ensure privacy and security. For example, account numbers could have the last six digits replaced with “x” characters.
- **Encryption**. Even after data is processed and ready for transmission outside of Pion, it can be encrypted with SSL to ensure secure delivery.
- **Passwords**. Pion itself is password protected and the administration interface supports encryption and VPN technologies to ensure that no one can manipulate Pion itself.
- **Protected by network security**. Pion benefits from the protection offered by the base operating system on which it resides. This means your security people can use their own additional measures to provide security.
- **Out-of-band management interfaces** . Pion supports out-of-band management interfaces, enabling the construction of completely isolated network components that are inaccessible to outside influences.

Pion: The Application

Pion is built to be fast and easy to use without requiring a great amount of technical savvy. The application interface is browser-based, and uses drag-and-drop items and simple menus. The pre-configured Reactors, Codecs, and Vocabularies allow you to be productive quickly, and the ability to customize and create new Reactors, Codecs, and Vocabularies means your Pion system will never become outdated.

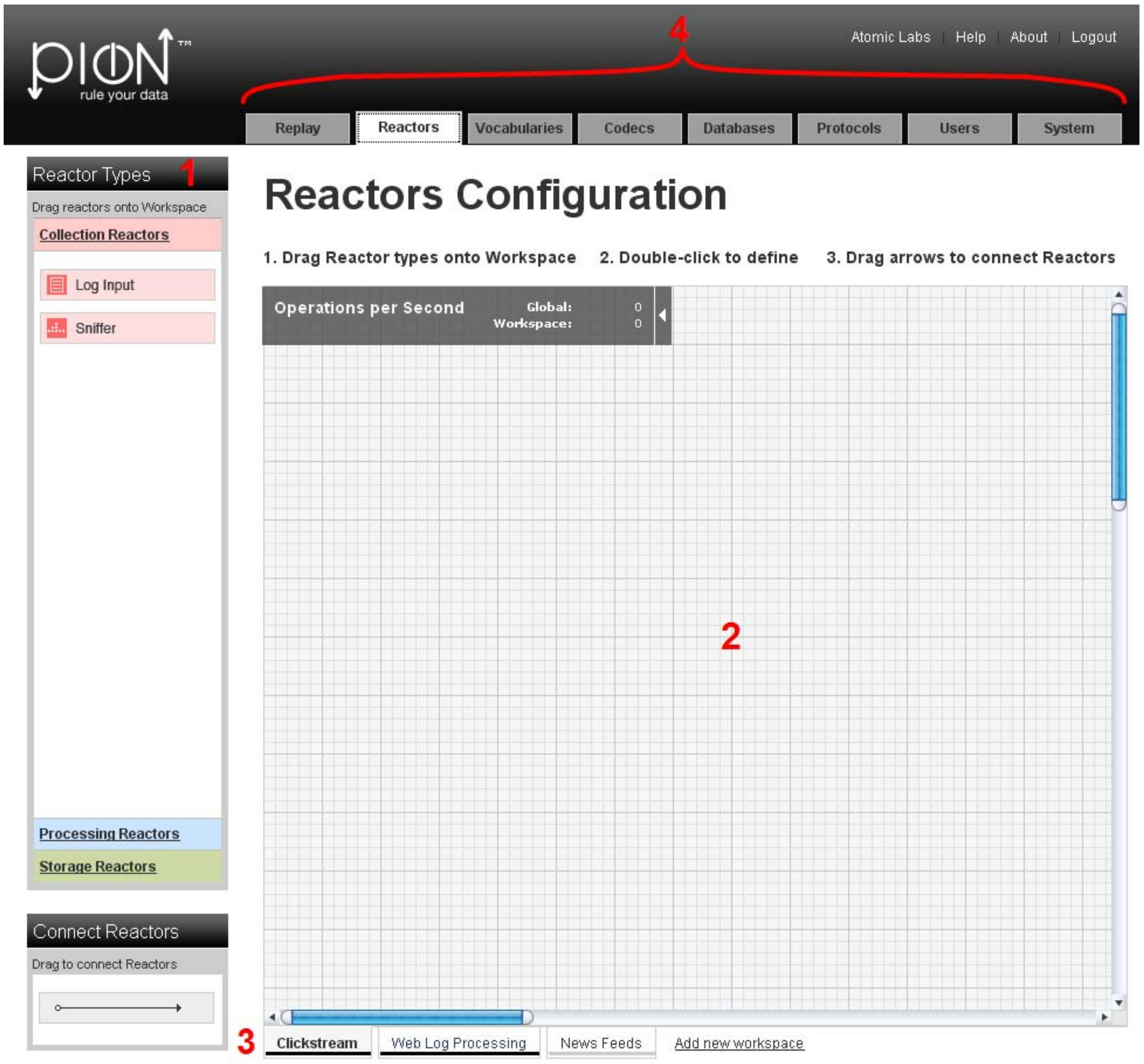
Getting Started with Pion

Getting started is simple, just point your favorite web browser to port 8888 on the server Pion is installed on (for example http://myserver.mycompany.com:8888). The login screen will appear after a few seconds. (Note that, the first time you access a Pion instance from a new client, it will take a few extra seconds longer for the login screen to appear, because some tools are being loaded in background.)



Enter your user name and password. The default username and password are both “pion” and it is strongly suggested that you change the default login information if you are the administrator. For more details on administration please see the chapter on managing users later in this guide.

After logging into Pion you will be presented with the Reactors Configuration screen where you will spend most of your time.



The screen is made up of four main sections (indicated in red):

1. Along the left hand side of the screen you will find the Reactor Types toolbar. This toolbar contains the basic building blocks of every workflow you will design.
2. The configuration workspace where you will draw your chain reactions.
3. Workspace tabs are used to separated workspaces, making organization quick and easy.
4. Navigation tabs for the different sections of Pion.

Reactor Configuration Page

The Pion Reactor Configuration page is where you build your chains. By dragging Reactors of different types to the workspace, and then connecting them, you define the sequence of data capture, processing, and output.

Reactor Types Toolbar

In the Pion application, the “Reactor Types” menu is divided along Reactor family lines. Working with reactors is pretty easy, because reactors can be dragged from the pallet on the left into the workspace area.

When you place the reactor in the workspace, you will be prompted for some basic naming information for the reactor plus any special information for the particular reactor you are creating. For example when you a Sniffer Reactor you are prompted for the general fields for “Name” and ”Comment” as well as two fields specific to the Sniffer Reactor.

Collection Reactors

Collection Reactors are responsible for getting data into Pion and therefore **every Pion workflow needs to 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:

- **Sniffer Reactor**, which captures/monitors network traffic.
- **Log File Input Reactor**, which reads log files.

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:

- **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.
- **Session Filter Reactor**, which enables you 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.
- **Aggregate Reactor**, which is able to take large volumes of raw data and distill critical pieces of information out of them. For example, the Aggregate Reactor could count all of the web users on the site in the last hour, calculate the average page size the 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 of Pion the Aggregate Reactor can do real-time report and reduce the stress on backend networks and systems which is especially critical in high volume applications.
- **Fission Reactor**, which splits input events into sequences of output events. For example, after receiving an input event of a news feed containing multiple RSS and XML items, the Fission Reactor can split the incoming event into an series of events containing only the RSS items.
- **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.

- **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.
- **Filter Reactor**, which filters 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.
- **Clickstream Reactor**, by understanding how browsers normally communicate with backend systems and how session state is maintained, is able take raw web traffic and organize it into logical pages and individual user sessions. The Clickstream Reactor does this in order to determine who is who and gather overall session statistics such as visit length, pages viewed, images viewed, time per page view, and more. Often, you won't have to configure any of the options inside of the Clickstream Reactor for it to work correctly. If necessary though, though the Clickstream Reactor also contains the ability to handle very complex and custom sessionization parameters that are sometimes employed by very large applications to maintain session state across globally distributed application instances.
- **Script Reactor**, which enables you to leverage existing command line scripts to quickly add capabilities to Pion. Using 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.

Storage Reactors

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

The main Storage Reactors include:

- **Database Output Reactor**, which 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.
- **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. 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.
- **Omniture Genesis Reactor**, which is able to take web user behavior information and send it through the Omniture Genesis integration system to SiteCatalyst via its native web services interface (in real-time) or the XML loading interface. All you need to supply is the relevant account information and site name to connect Pion to Omniture..
- **Log Output Reactor**, which 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.

- **Multi Database Reactor** enables Pion to store data in several databases, based on the data being stored.

Once Reactors have been dragged into the workspace and connected, you can:

1. Double-click on a Reactor to display its configuration dialog. You can then change the Reactor's configuration or connection characteristics.
2. Right-click on a Reactor to:
   - Display the Reactor's configuration dialog. You can then change the Reactor's configuration or connection characteristics.
   - Display its configuration in XML format in a new page (note that this is read-only).
   - Delete the Reactor.

The Vocabulary Configuration Page

This page allows you to manage and specify (define) the vocabularies used to describe events. Vocabularies allow you to define the terms and the format of the elements (data structures) that make up an event. There are five (5) built-in vocabularies:

- Clickstream
- RSS Channels
- Atom Feeds
- Stock Prices
- Aggregate

If necessary, you can alter the built-in vocabularies, or even create your own specialized vocabulary to describe another kind of event.

The Codec Configuration Page

A Pion Codec defines an input or output format for a log or a data stream (feed). Once the codec is defined, the log or data stream can be read and events can be created from the data in those logs/data streams. Codecs can also be used by Storage Reactors when writing events or results.

There are 18 built-in codecs in Pion:

- Common Log Format
- Combined Log Format
- Extended Log Format
- Page View Log Format
- Visitor Session Log Format
- WebTrends Log
- 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 Codec
- Atom Entry Extraction Codec
- Atom Entry Log Codec
- Stock Price Log
- JSON Log Format
- XML Log Format

If necessary, you can alter the built-in codecs, or create your own specialized codec to describe another kind of input or output format.

The Database Configuration Page

The Database Configuration Page is used to configure a connection with a database. Connecting to a database means:

- Identifying the type of database (for exampleMySQL, SQLite or Oracle).

  Note: In some cases, you will also have to identify the sub-type (engine) of the database being used. For example MySQL-MyISAM or MySQL-InnoDB; there are differences in the engines, and Pion makes effective use of those differences to optimize performance by changing the SQL syntax and/or the Isolation level.

- Specifying the database connection information:
  - The Username/Password for the Pion user account (the Pion login) on the database.
  - The host/machine/instance of the database to connect to (host:port, or similar).
  - The individual database instance (the database name space).
  - The "connect strings" Pion uses to connect to the database (note that the connect string will be different for each kind/type of database, and may be different for individual databases of the same type/kind). For more information about connect strings, see http://sqlapi.com/OnLineDoc/Connection_Connect.html

SQLite is Pion's embedded database, meaning it is always available, and by default will be used by Reactors, the analytics interfaces (for both Omniture Genesis and Google Analytics), and the Replay feature.

Note: SQLite is not a client/server database, though it is a RDBMS (relational database management system). Instead of a username and password pair, SQLite only has a filename.

The Protocol Configuration Page

The Protocol Configuration Page is used to specify and configure the protocols Pion monitors. Currently there are two (2) variations of the HTML protocol built-in to Pion:

- HTTP (No Content), which is used when Pion is monitoring visitor sessions to your web site(s), but not collecting data to be used by the Replay feature..
- HTTP (Full Content), which is used to capture the data required by the Replay feature. See “Using the Replay Feature“ on page 20 for more information.

Because of Pion's extensible architecture, adding support for additional protocols is simply a matter of altering or creating codecs and/or vocabularies.

Managing and Configuring Pion

User Configuration

The User Configuration page is used to manage Pion user accounts. You can add, delete, or modify user accounts.

System Configuration

The System Configuration page is used to view the configuration of the Pion application, including configuration file location, services, configuration paths, and plugin paths. You cannot make changes from this page, but you can find out where the configuration files are, and then edit them separately, if necessary. Note that all configuration files are in XML format.

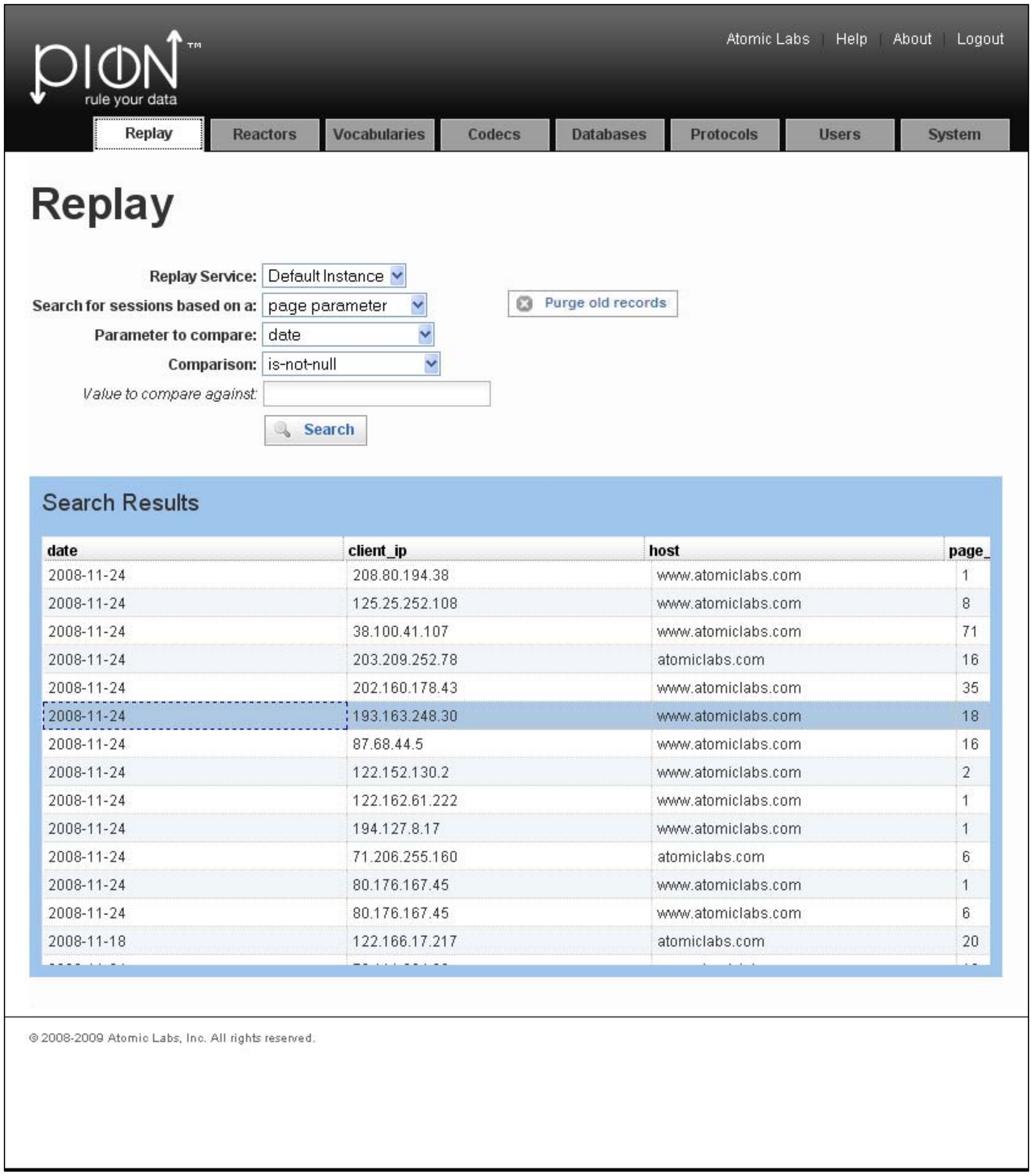
The System Configuration page is also where you can add a Reactor, Connection, codec, or database by importing the XML definition of the Reactor, Connection, codec, or database.

Using the Replay Feature

The Replay feature allows you to see what a visitor to your web site saw and “step through” their experience on your web site.

To use Replay, Pion must be configured to monitor the web server, and the Content Storage Reactor and the Database Output Reactors must be properly configured.

Note: Currently, the only supported database for Replay is the embedded SQLite database. In future releases, additional databases may be supported.



Replay

Replay Service: Default Instance
Search for sessions based on a: page parameter
Parameter to compare: date
Comparison: is-not-null
Value to compare against:

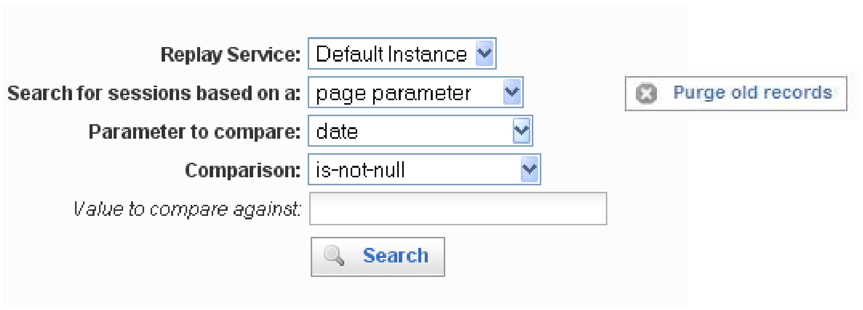
Search Results

date	client_ip	host	page_
2008-11-24	208.80.194.38	www.atomiclabs.com	1
2008-11-24	125.25.252.108	www.atomiclabs.com	8
2008-11-24	38.100.41.107	www.atomiclabs.com	71
2008-11-24	203.209.252.78	atomiclabs.com	16
2008-11-24	202.160.178.43	www.atomiclabs.com	35
2008-11-24	193.163.248.30	www.atomiclabs.com	18
2008-11-24	87.68.44.5	www.atomiclabs.com	16
2008-11-24	122.152.130.2	www.atomiclabs.com	2
2008-11-24	122.162.61.222	www.atomiclabs.com	1
2008-11-24	194.127.8.17	www.atomiclabs.com	1
2008-11-24	71.206.255.160	atomiclabs.com	6
2008-11-24	80.176.167.45	www.atomiclabs.com	1
2008-11-24	80.176.167.45	www.atomiclabs.com	6
2008-11-18	122.166.17.217	atomiclabs.com	20

© 2008-2009 Atomic Labs, Inc. All rights reserved.

To replay a visitor's session:

1. Go to the Replay page of the Pion application.
2. Select the Replay Service (the Pion instance that is monitoring the web server that hosted the sessions you want to review).
3. Using the drop-down list boxes, specify the search criteria for the session(s) you want to review:


Replay Service: Default Instance

Search for sessions based on a: page parameter

Parameter to compare: date

Comparison: is-not-null

Value to compare against:

4. Select the **Search for session based on** criteria. You can select one of the following:

    - **Session parameters**, which include session-based information, such as server IP address, client IP address, user agent (the visitor's browser), referrer, cookie ID, and other factors relating to the session itself.

      Because Pion 'sessionizes' web visits and stores that information in a database, using the **Session parameters** will be the fastest way to find particular sessions. Finding a particular session using **Session parameters** is more efficient than the other search criteria parameters because the database storing session information is smaller and will therefore be processed faster than the others. However, because there is a built-in delay to allow sessions to time-out (1800 seconds by default), you may want to use page or request parameters to locate the session.

      There are two reasons to use the page or request parameters:

        - To see sessions that have not yet timed out (a way to handle the sessionizing timeout).
        - To be able to search based on parameters that are only available in page or request metadata. For example: status (to find sessions that had page-not-found requests) or page_title (to find sessions that hit a particular page title).

    - **Page parameters**, which include page-based data, such as server IP address, host, page number, page status, URI query (the name of the called resource), and other parameters related to the page the visitor(s) viewed.
    - Finding a session using page parameters involves grouping all page views together into a Session view, which slows searches.
    - **Request parameters**, which include details about the incoming client request, such as date, time, session ID, client IP, and other client-related details.

      Finding a session using request parameters involves searching through all the individual requests, grouping all requests together, and other processing steps, making this the slowest/most resource intensive search type.

5. Select the **Parameter to compare** criteria.

   Depending on the selection you made in the **Search for session based on** criteria, the list of possible parameters to compare against changes. Select the parameter that will identify the user session(s) in which you have an interest. (You can select only one parameter at a time.)

   Select the **Comparison** to make:

    - **is-not-null**, which means there must be a value (any type of value) to compare against. This will result in a list of requests/pages/sessions that have a non-empty parameter of the type you specified in the **Parameter to compare** criteria.
    - **is-null**, which means there must **not** be a value to compare against. This will result in a list of requests/pages/sessions that have no value specified for the parameter of the type you specified in the **Parameter to compare** criteria.

- **exact-match**, which means an exact match to the string you will specify in the next step. This will result in a list of requests/pages/sessions whose **Parameter to compare** values exactly match the string you will specify in the next step.
- **not-exact-match**, which means “anything except an exact match” to the string you will specify in the next step. This will result in a list of requests/pages/sessions whose **Parameter to compare** values **are anything other than** the string you will specify in the next step.
- **like**, which means you will specify a character string, which will provide the pattern to compare against. Note that wildcards, such as "%" (a multi-character wild-card) and "_" (a single-character wildcard). This will result in a list of requests/pages/sessions whose **Parameter to compare** values match the pattern you will specify in the next step.
- **not-like**, which means you will specify a character string, which will provide the pattern to compare against. Note that wildcards, such as "%" (a multi-character wild-card) and "_" (a single-character wildcard). This will result in a list of requests/pages/sessions whose **Parameter to compare** values **do not** match the pattern you will specify in the next step.
- **ordered-before**, which means you will specify a string containing some form of sequence identifier, which will be compared against (a date, a numerical or alphabetic value). This will result in a list of requests/pages/sessions whose **Parameter to compare** values come before (are lower or earlier) in the sequence than the string you will specify in the next step.
- **not-ordered-before**, which means you will specify a string containing some form of sequence identifier, which will be compared against (a date, a numerical or alphabetic value). This will result in a list of requests/pages/sessions whose **Parameter to compare** values come after (are higher or later) in the sequence than the string you will specify in the next step.
- **ordered-after**, which means you will specify a string containing some form of sequence identifier, which will be compared against (a date, a numerical or alphabetic value). This will result in a list of requests/pages/sessions whose **Parameter to compare** values come after (are higher or later) in the sequence than the string you will specify in the next step.
- **not-ordered-after**, which means you will specify a string containing some form of sequence identifier, which will be compared against (a date, a numerical or alphabetic value). This will result in a list of requests/pages/sessions whose **Parameter to compare** values come before (are lower or earlier) in the sequence than the string you will specify in the next step.
- **starts-with**, which means you will specify a string, which will be compared against the beginning of the stored data. This will result in a list of requests/pages/sessions whose **Parameter to compare** values start with the string you will specify in the next step. Note that this comparison is not case sensitive.
- **ends-with**, which means you will specify a string, which will be compared against the final characters of the values in the stored data. This will result in a list of requests/pages/sessions whose **Parameter to compare** values end with the string you will specify in the next step. Note that this comparison is not case sensitive.
- **contains**, which means you will specify a string, which will be compared against the values in the stored data. This will result in a list of requests/pages/sessions whose **Parameter to compare** values include (contain) the string you will specify in the next step. Note that this comparison is not case sensitive.

Note: Comparisons depend on the specific database engine used to store the meta-data. Currently, Pion supports only the embedded SQLite database engine.

6. Using the **Value to compare against** field, specify the search criteria for the session(s) you want to review.

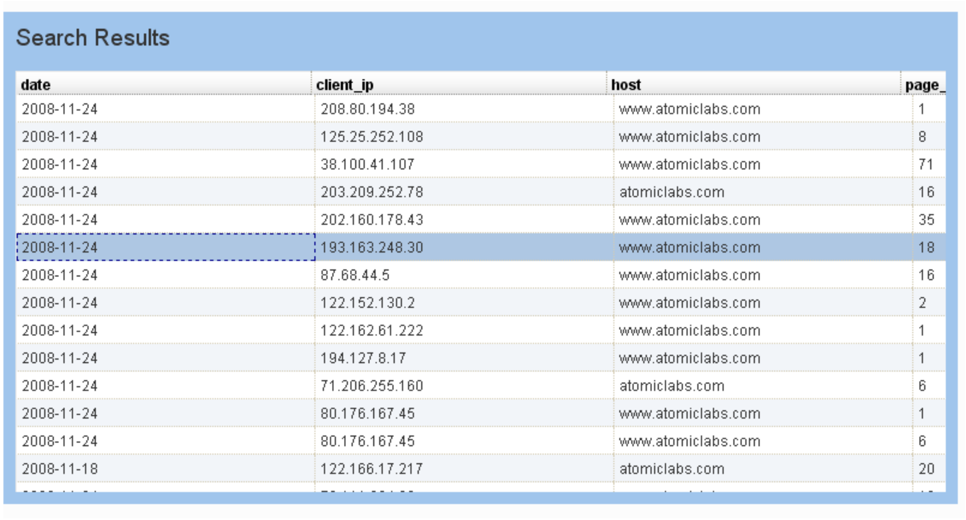
Note: The **Value to compare against** field is disabled for comparisons for which it is not applicable, such as **is-null** and **is-not-null**.

Depending on the selection you made in the search criteria above, enter the value/regular expression/sequence identifier for comparison.

7. Click **Search**.

   Note: If the search time exceeds 10 seconds, the query will time out. If you experience this situation, consider adding indexes or pruning the database tables (pruning can be done by clicking the Purge button, or with a pruning script).

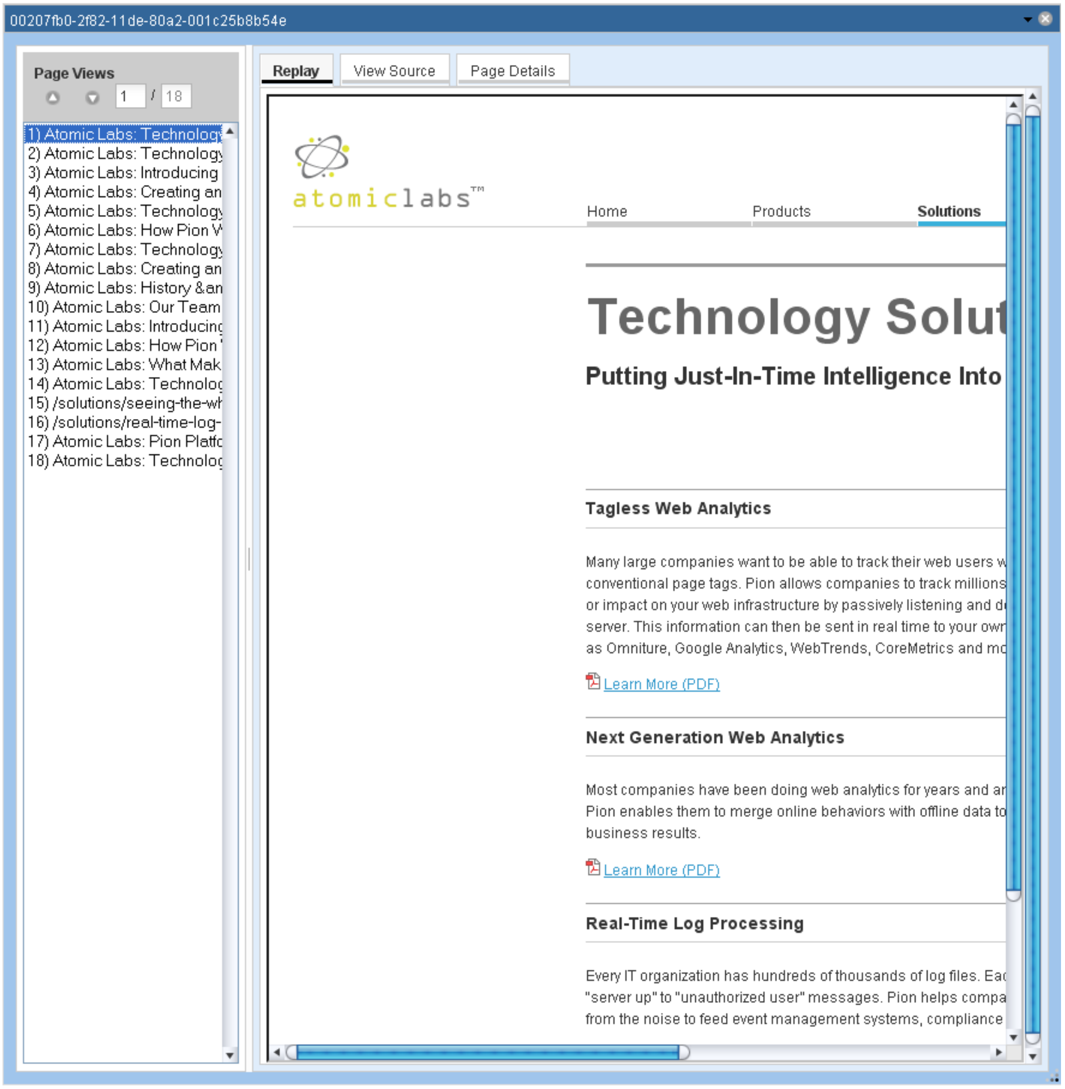
   The search results, if any, are displayed below the search criteria area.



Search Results

date	client_ip	host	page_
2008-11-24	208.80.194.38	www.atomiclabs.com	1
2008-11-24	125.25.252.108	www.atomiclabs.com	8
2008-11-24	38.100.41.107	www.atomiclabs.com	71
2008-11-24	203.209.252.78	atomiclabs.com	16
2008-11-24	202.160.178.43	www.atomiclabs.com	35
2008-11-24	193.163.248.30	www.atomiclabs.com	18
2008-11-24	87.68.44.5	www.atomiclabs.com	16
2008-11-24	122.152.130.2	www.atomiclabs.com	2
2008-11-24	122.162.61.222	www.atomiclabs.com	1
2008-11-24	194.127.8.17	www.atomiclabs.com	1
2008-11-24	71.206.255.160	atomiclabs.com	6
2008-11-24	80.176.167.45	www.atomiclabs.com	1
2008-11-24	80.176.167.45	www.atomiclabs.com	6
2008-11-18	122.166.17.217	atomiclabs.com	20

8. Double-click on the session you want to view. The Replay window appears:


00207fb0-2f82-11de-80a2-001c25b8b54e

Page Views

1 / 18

1) Atomic Labs: Technolog
2) Atomic Labs: Technology
3) Atomic Labs: Introducing
4) Atomic Labs: Creating an
5) Atomic Labs: Technology
6) Atomic Labs: How Pion W
7) Atomic Labs: Technology
8) Atomic Labs: Creating an
9) Atomic Labs: History &an
10) Atomic Labs: Our Team
11) Atomic Labs: Introducing
12) Atomic Labs: How Pion
13) Atomic Labs: What Mak
14) Atomic Labs: Technolog
15) /solutions/seeing-the-wh
16) /solutions/real-time-log-
17) Atomic Labs: Pion Platfc
18) Atomic Labs: Technolog

Technology Solut

Putting Just-In-Time Intelligence Into

Tagless Web Analytics

Many large companies want to be able to track their web users w
conventional page tags. Pion allows companies to track millions
or impact on your web infrastructure by passively listening and d
server. This information can then be sent in real time to your own
as Omniture, Google Analytics, WebTrends, CoreMetrics and mo

Learn More (PDF)

Next Generation Web Analytics

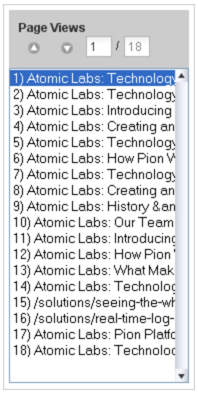
Most companies have been doing web analytics for years and ar
Pion enables them to merge online behaviors with offline data to
business results.

Learn More (PDF)

Real-Time Log Processing

Every IT organization has hundreds of thousands of log files. Eac
"server up" to "unauthorized user" messages. Pion helps compa
from the noise to feed event management systems, compliance

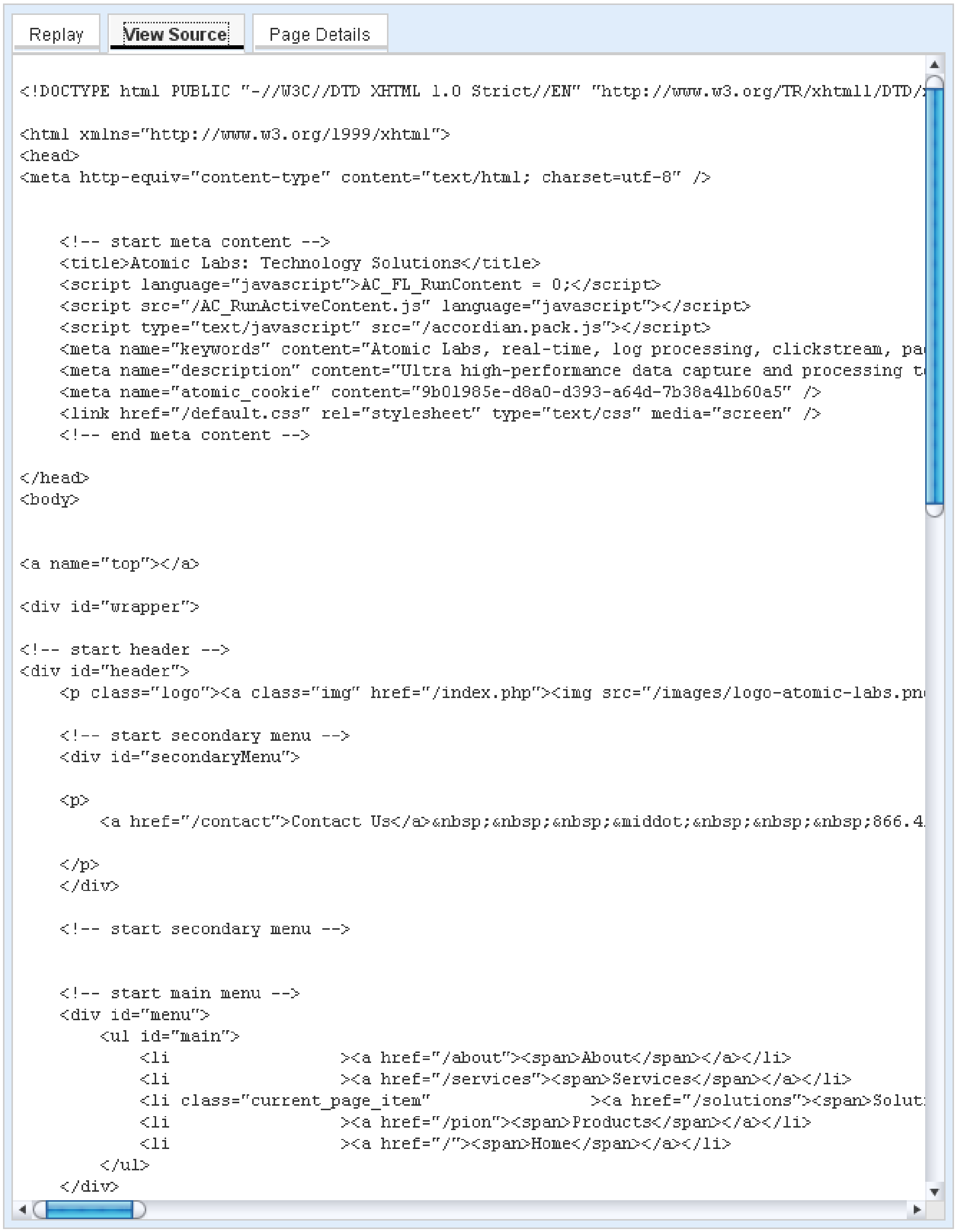
- **To see the pages that the web site visitor saw**, go to the **Page Views** list and click on the page you want to view.


Page Views

1 / 18

1) Atomic Labs: Technology
2) Atomic Labs: Technology
3) Atomic Labs: Introducing
4) Atomic Labs: Creating an
5) Atomic Labs: Technology
6) Atomic Labs: How Pion W
7) Atomic Labs: Technology
8) Atomic Labs: Creating an
9) Atomic Labs: History &an
10) Atomic Labs: Our Team
11) Atomic Labs: Introducing
12) Atomic Labs: How Pion
13) Atomic Labs: What Mak
14) Atomic Labs: Technolog
15) /solutions/seeing-the-wh
16) /solutions/real-time-log-
17) Atomic Labs: Pion Platfc
18) Atomic Labs: Technoloc

- **To see the page source**, select the **View Source** tab.



```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />


    <!-- start meta content -->
    <title>Atomic Labs: Technology Solutions</title>
    <script language="javascript">AC_FL_RunContent = 0;</script>
    <script src="/AC_RunActiveContent.js" language="javascript"></script>
    <script type="text/javascript" src="/accordian.pack.js"></script>
    <meta name="keywords" content="Atomic Labs, real-time, log processing, clickstream, pa
    <meta name="description" content="Ultra high-performance data capture and processing t
    <meta name="atomic_cookie" content="9b01985e-d8a0-d393-a64d-7b38a41b60a5" />
    <link href="/default.css" rel="stylesheet" type="text/css" media="screen" />
    <!-- end meta content -->

</head>
<body>


<a name="top"></a>

<div id="wrapper">

<!-- start header -->
<div id="header">
    <p class="logo"><a class="img" href="/index.php"><img src="/images/logo-atomic-labs.pn

    <!-- start secondary menu -->
    <div id="secondaryMenu">

    <p>
        <a href="/contact">Contact Us</a>&nbsp;&nbsp;&nbsp;&middot;&nbsp;&nbsp;&nbsp;866.4

    </p>
    </div>

    <!-- start secondary menu -->


    <!-- start main menu -->
    <div id="menu">
        <ul id="main">
            <li                 ><a href="/about"><span>About</span></a></li>
            <li                 ><a href="/services"><span>Services</span></a></li>
            <li class="current_page_item"               ><a href="/solutions"><span>Solut
            <li                 ><a href="/pion"><span>Products</span></a></li>
            <li                 ><a href="/"><span>Home</span></a></li>
        </ul>
    </div>
```

- **To see the page details for the currently selected page view**, select the **Page Details** tab.

Enterprise Edition

Part II: Reactor Reference

Last update: July 2009

Table of Contents

About this Document

This document contains notes and useful information on Pion reactors.

Reactor Type	**Reactor Name**	**See Page**
Collection	Log File Input Reactor	8
	Sniffer Reactor	11
Processing/Transform	Aggregate Reactor	15
	Clickstream Reactor	25
	Content Hash Reactor	32
	Filter Reactor	34
	Fission Reactor	38
	Script Reactor	40
	Session Filter Reactor	40
	SQL Reactor	42
	Transform Reactor	48
Storage	Database Output Reactor	54
	Google Analytics Reactor	58
	Log Output Reactor	60
	Multi Database Reactor	62
	Omniture Analytics Reactor	65

Important Notes

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 n names that are the ‘lowest common denominator,’ any name that is valid in Pion they should work in any database.

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

- Start with a letter (a-z or A-Z)
- Contain only letters (a-z or A-Z), numbers (0-9) and the underscore character (_)
- 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** (checkbox), 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. However, in order to delete a connection, you must 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

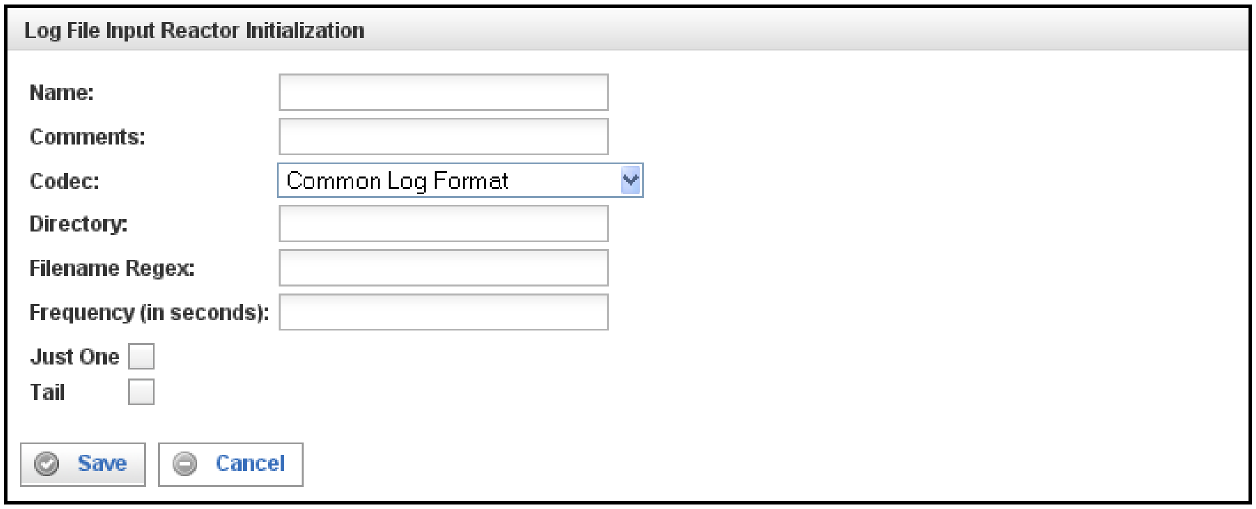
The Log File Input Reactor can capture data from any structured text file so long as that file's format is defined in a "Codec." A Pion Codec defines an input or output format for things like logs. A Codec can also define notation such as JSON or XML. Codecs are used to 'break' data into chunks that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

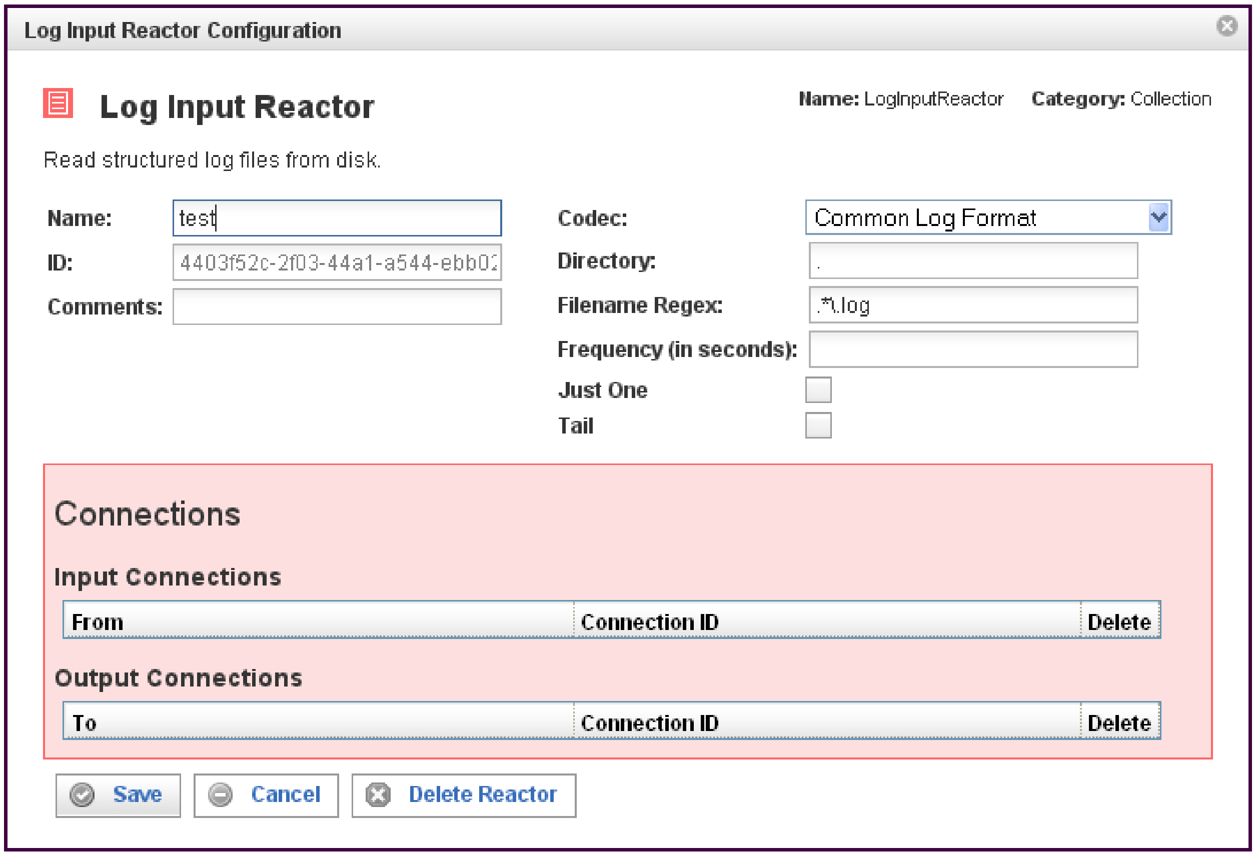

Log File Input Reactor Initialization

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 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 Log • Response Content Log • RSS Channel Extraction Codec • RSS Channel Log Format • RSS Item Extraction Codec • RSS Item Log Format • Atom Feed • Atom Feed • Atom Entry • Atom Entry • 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.
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.

Field	Description
Just One	Fill this check box to read only the first record of a log file, converts it to an event, and repeatedly duplicate and output that event until the reactor is stopped. (This is useful for testing.)
Tail	When this checkbox 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
ID:	4403f52c-2f03-44a1-a544-ebb02
Comments:	
Codec:	Common Log Format
Directory:	.
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.

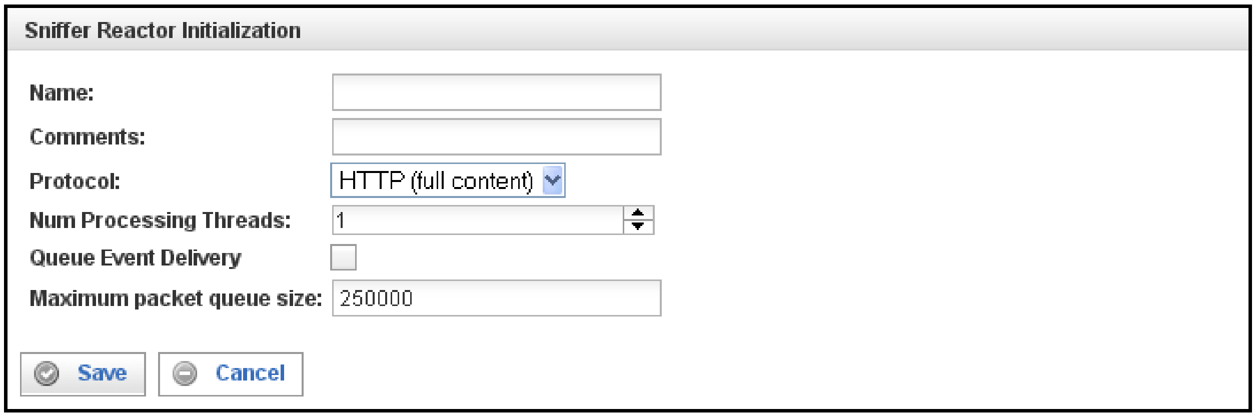
The Connections Section

The Connections section describes input and output connections for this Log Input Reactor. See Reactor Input and Output Connections on page 7 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:

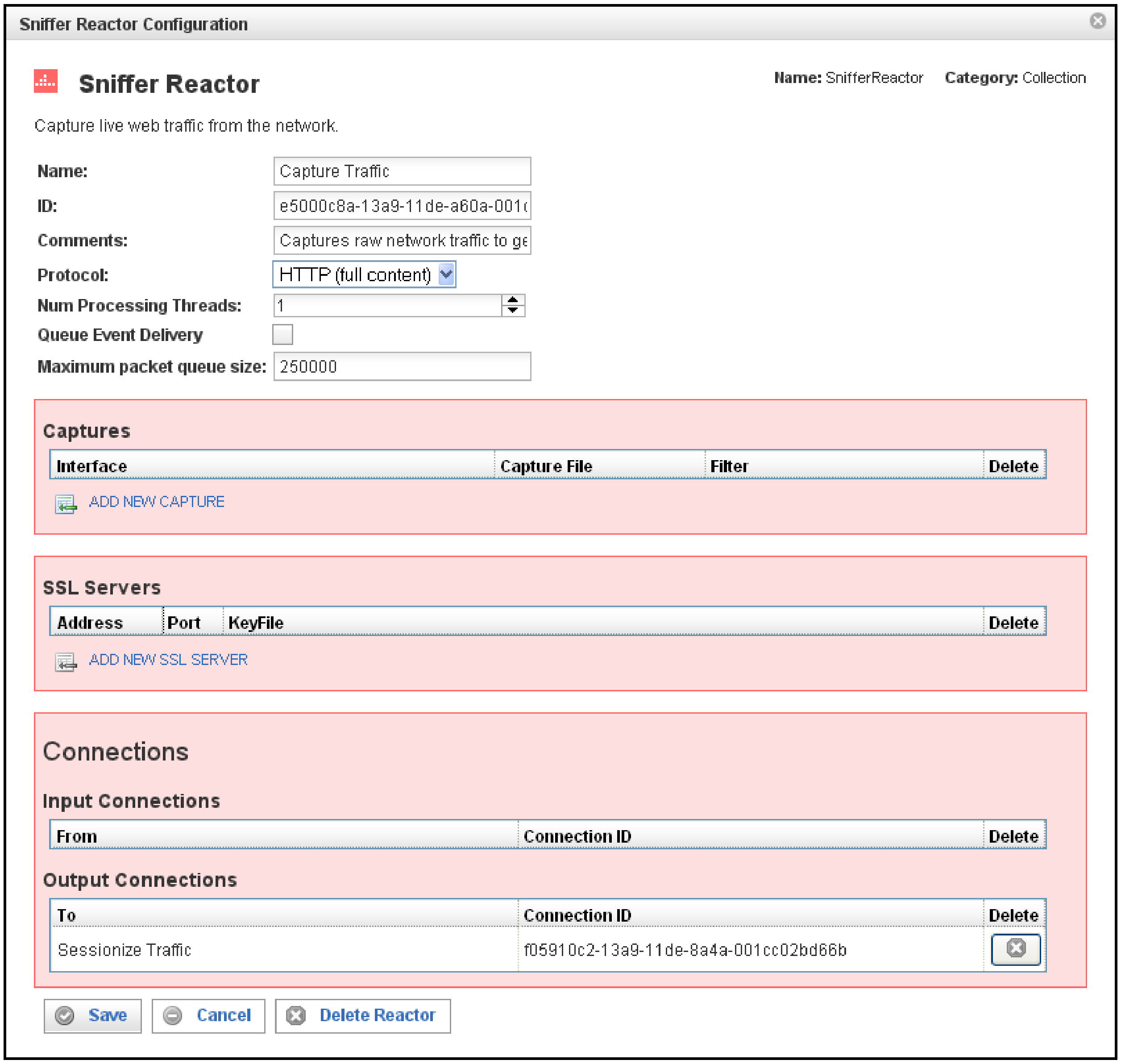

Sniffer Reactor Initialization

Name:	
Comments:	
Protocol:	HTTP (full content)
Num Processing Threads:	1
Queue Event Delivery	☐
Maximum packet queue size:	250000

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: • *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 (from the request and the 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. The default is 512k for *sc-content*. The client request is saved into a special field in the clickstream term called *cs-content*. 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* captures the critical fields from SMTP-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.

Field	**Description**
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.
Maximum packet queue size	Specify the maximum number of packets that can be in the event queue per processing thread. We recommend that the total (this value * number of processing threads) be between 200,000 and 300,000. 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 Configuration

Sniffer Reactor

Name: SnifferReactor **Category:** Collection

Capture live web traffic from the network.

Name:	Capture Traffic
ID:	e5000c8a-13a9-11de-a60a-001(
Comments:	Captures raw network traffic to ge
Protocol:	HTTP (full content)
Num Processing Threads:	1
Queue Event Delivery	☐
Maximum packet queue size:	250000

Captures

Interface	Capture File	Filter	Delete

ADD NEW CAPTURE

SSL Servers

Address	Port	KeyFile	Delete

ADD NEW SSL SERVER

Connections

Input Connections

From	Connection ID	Delete

Output Connections

To	Connection ID	Delete
Sessionize Traffic	f05910c2-13a9-11de-8a4a-001cc02bd66b	☒

Save Cancel Delete Reactor

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

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. Select the cell in the *Interface* column to display an automatically populated 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 listen to. Interfaces include: • eth0 / eth1 / eth*X* (Ethernet interfaces) • wlan0 (or similar, wireless interfaces) • lo / lo0 (or similar, often "loopback" devices) **Note**: If you are running VMware, there will be many other virtual interfaces, between the VM and the host. 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, run the command "`ifconfig -a`".
Capture File	Specifies a file that contains saved traffic. This content of the file will be used instead of live traffic from an interface. 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).
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 part 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 if you only wanted normal HTTP traffic you would use “tcp port 80” as your filter string to remove all non HTTP 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 SSL Servers Section

This section is used only for sites that use SSL encrypted communications such as those common in retail, defense, and financial applications. Pion is able to decrypt data in real-time provided it has the proper keys.

To specify an SSL server, simply click the *ADD NEW SSL SERVER* icon and specify the following:

Field	Description
Address	The IP Address or DNS name of the SSL server.
Port	The value specified is used in conjunction with the Address (above) to determine if the traffic is encrypted using the provided keyfile (below).
Keyfile	The path to the SSL key file. This path must be relative to the Pion server to ensure proper access. The key can be any standard PEM formatted file.

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

The Connections Section

The Connections section describes input and output connections for this Sniffer Reactor. See Reactor Input and Output Connections on page 7 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:

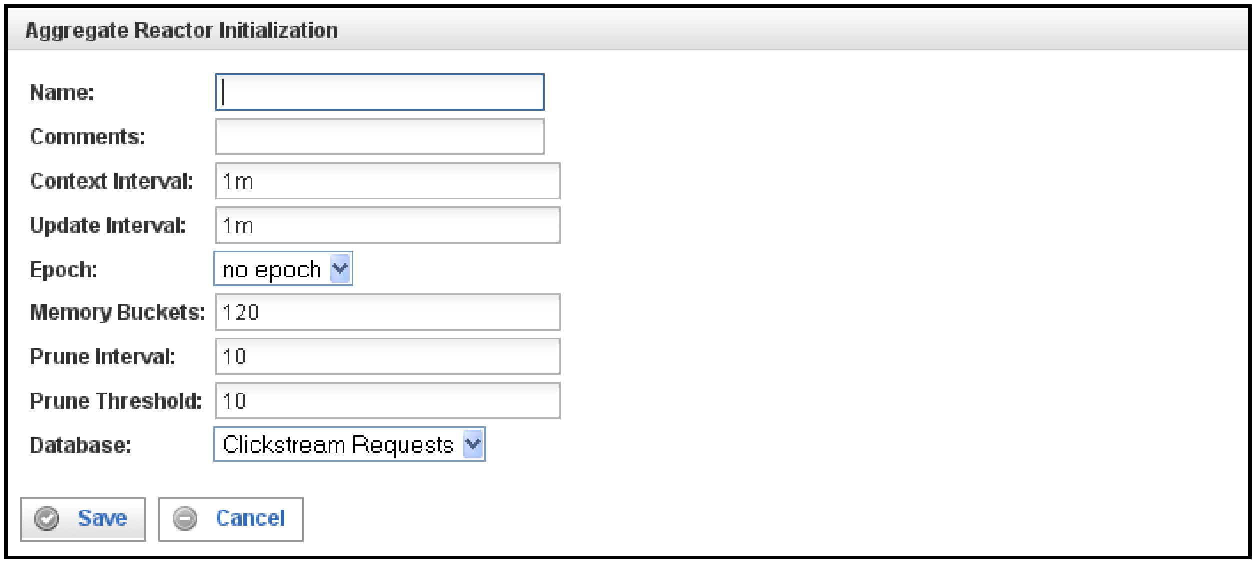
- **Aggregate Reactor**, which is able to take large volumes of raw data and distill critical pieces of information out of them.
- **Clickstream Reactor**, which is able take raw web traffic and organize it into logical pages and individual user sessions.
- **Content Hash Reactor**, which creates and 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.
- **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:



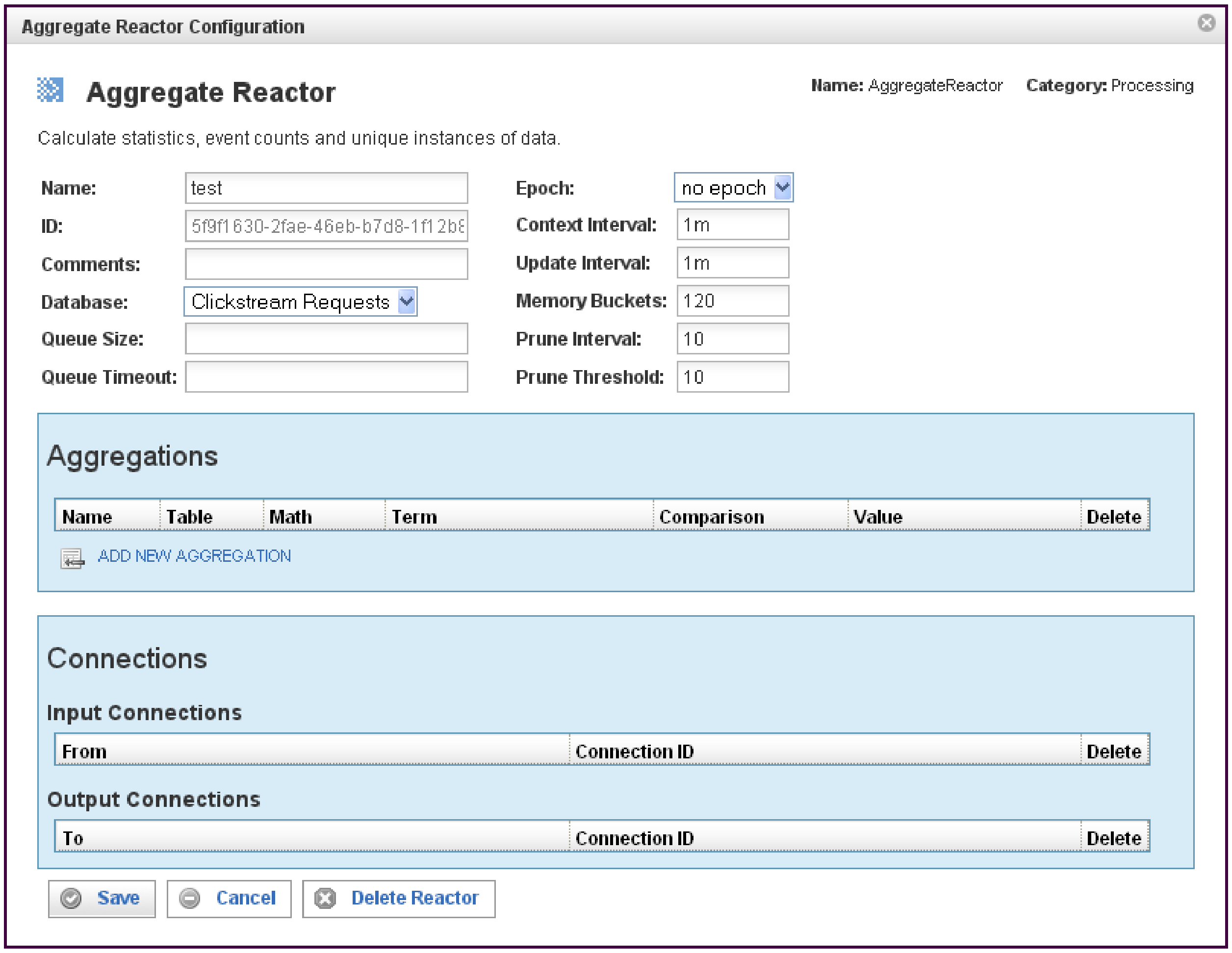
Name:	
Comments:	
Context Interval:	1m
Update Interval:	1m
Epoch:	no epoch
Memory Buckets:	120
Prune Interval:	10
Prune Threshold:	10
Database:	Clickstream Requests

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 must be unique (within Aggregate Reactors) and is not case sensitive. The name may contain any combination of letters and numbers.
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 “The Connections section describes input and output connections for this Aggregate Reactor. See Reactor Input and Output Connections on page 5 for more information. • Using the Query Service with the Aggregate Reactor” on page 23).
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).
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).

Field	**Description**
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

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.
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 must be unique (within aggregations) and is not case sensitive. 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 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:: • **SUM** (add the values together). • **MIN** (find the lowest value). • **MAX** (find the highest value). • **AVG** (calculate the average value). • **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: • Any unique value creates a new counter (also known as a “slot" or an "accumulator"). • Once a counter is created, the number of unique values is counted. • 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 7 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:

1. date, time, date-time, epoch-time = timestamp for first http event associated with the page
2. page-num = sequence number for the page view, starting with 1
3. robot = 1 if the session is considered to be a "robot"
4. bytes = total bytes for all requests and responses
5. cs-bytes = total bytes for all requests
6. sc-bytes = total bytes for all responses
7. page-title = title extracted from HTML content
8. page-load = total time to load the page, from start of first request to end of last response
9. page-load-redirect = total time for all redirected (302) page requests
10. page-load-base = time-taken for the base HTML file
11. page-hits = total number of HTTP requests associated with the page
12. page-load-content = total time for all page object (not base HTML or redirected) requests
13. page-dwell = total time from the last response to the first request for the next page view
14. referer = copied from first http event associated with the page

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.

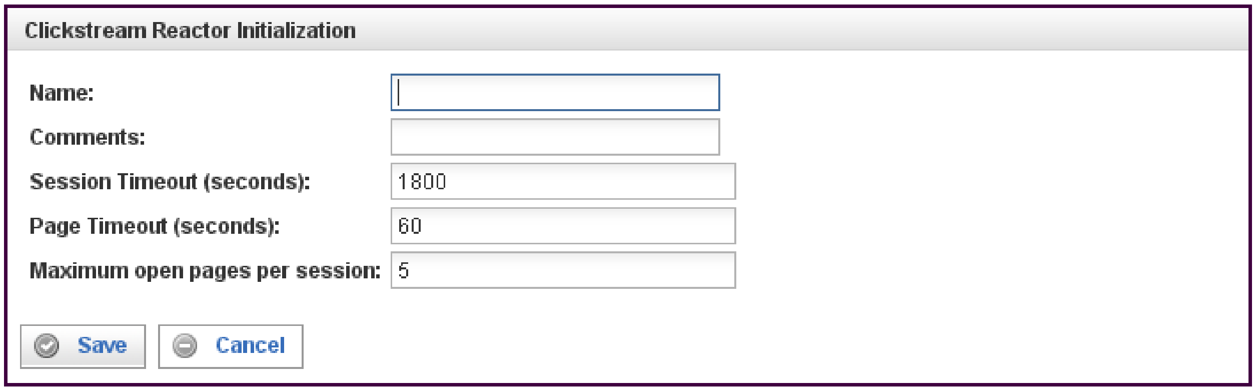
7. Requests are assigned to sessions using the cookies configured in the Clickstream Reactor, and if no cookies are found, a combination of the user agent and the IP address are used for matching.
8. When a session closes, a "session event" is generated that represents all the requests associated with a particular visitor session.

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

1. date, time, date-time, epoch-time = timestamp for first http event associated with the session
2. session-id, session-group, date, time, date-time, referer
3. session-pages = number of pages in the session
4. session-requests = number of http requests in the session
5. session-dwell = total dwell time for all page views
6. session-length = total length of the session
7. robot = 1 if the session is considered to be a "robot"
8. bytes = total bytes for all requests and responses
9. cs-bytes = total bytes for all requests
10. sc-bytes = total bytes for all responses
11. 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.

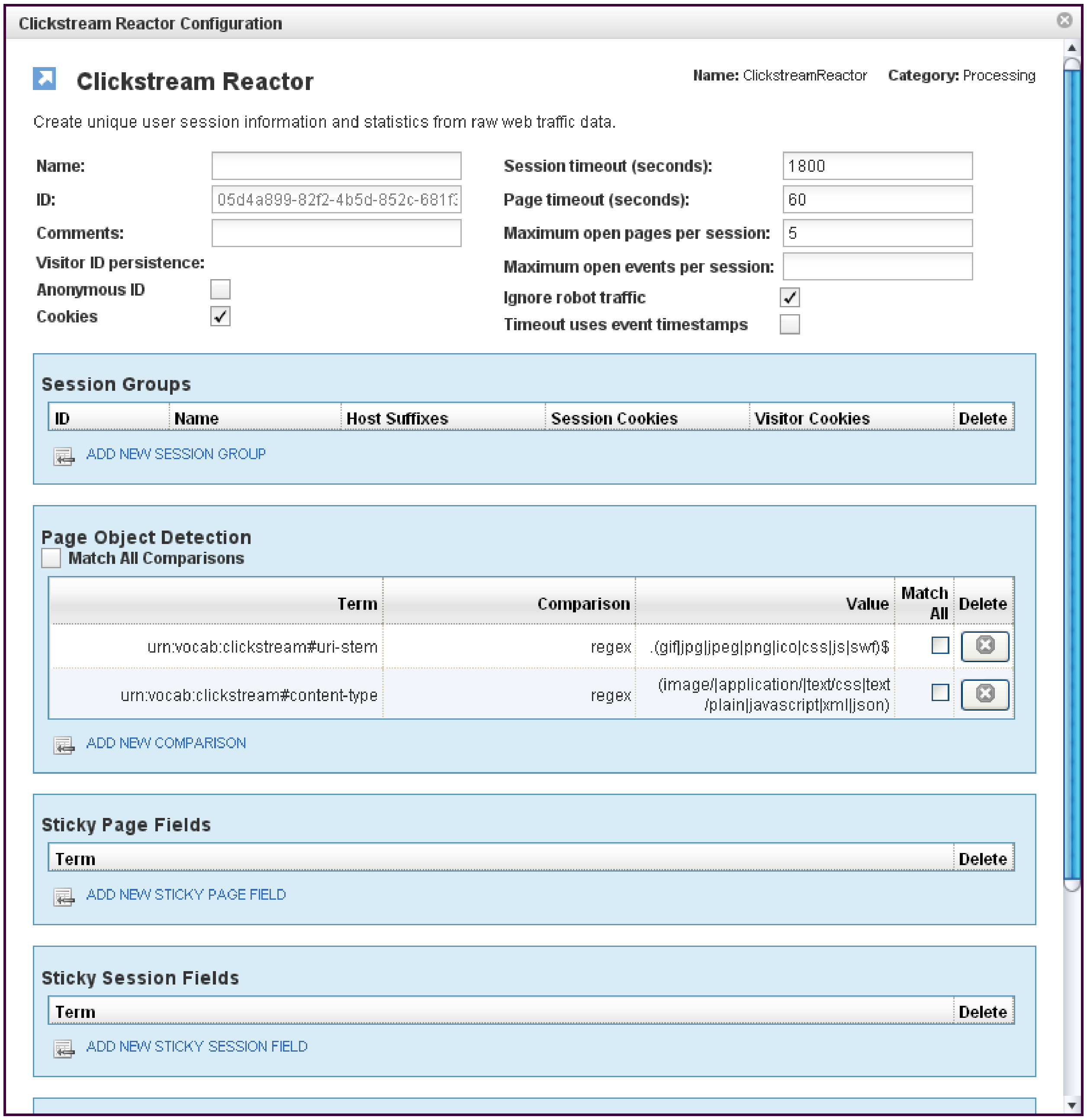
When you add a Clickstream Reactor you will be prompted for the following fields:



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 must be unique (within Clickstream Reactors) and is not case sensitive. 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.
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 checkbox 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 checkbox 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.
Ignore robot traffic (checkbox)	When this checkbox is filled, all sessions detected as "robots" or "spiders" are discarded by the Clickstream Reactor. When this checkbox 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 (checkbox)	When this checkbox 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 checkbox should always be filled. When using live network interfaces, this checkbox 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.
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 6 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 (checkbox)	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 (checkbox)	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.

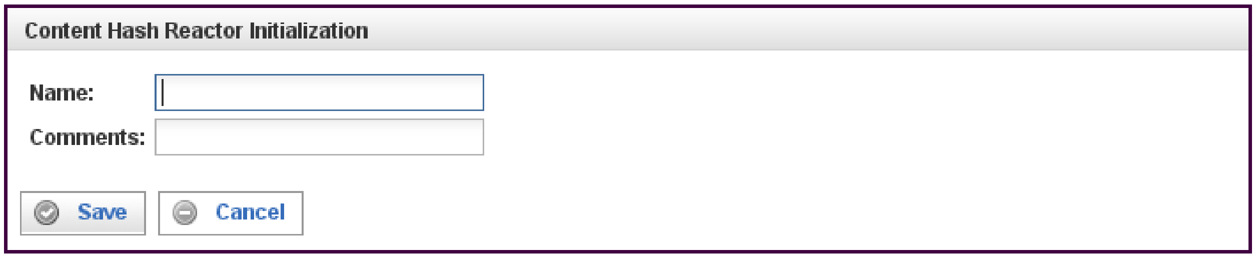
The Connections Section

The Connections section describes input and output connections for this Clickstream Reactor. See Reactor Input and Output Connections on page 7 for more information.

Content Hash Reactor

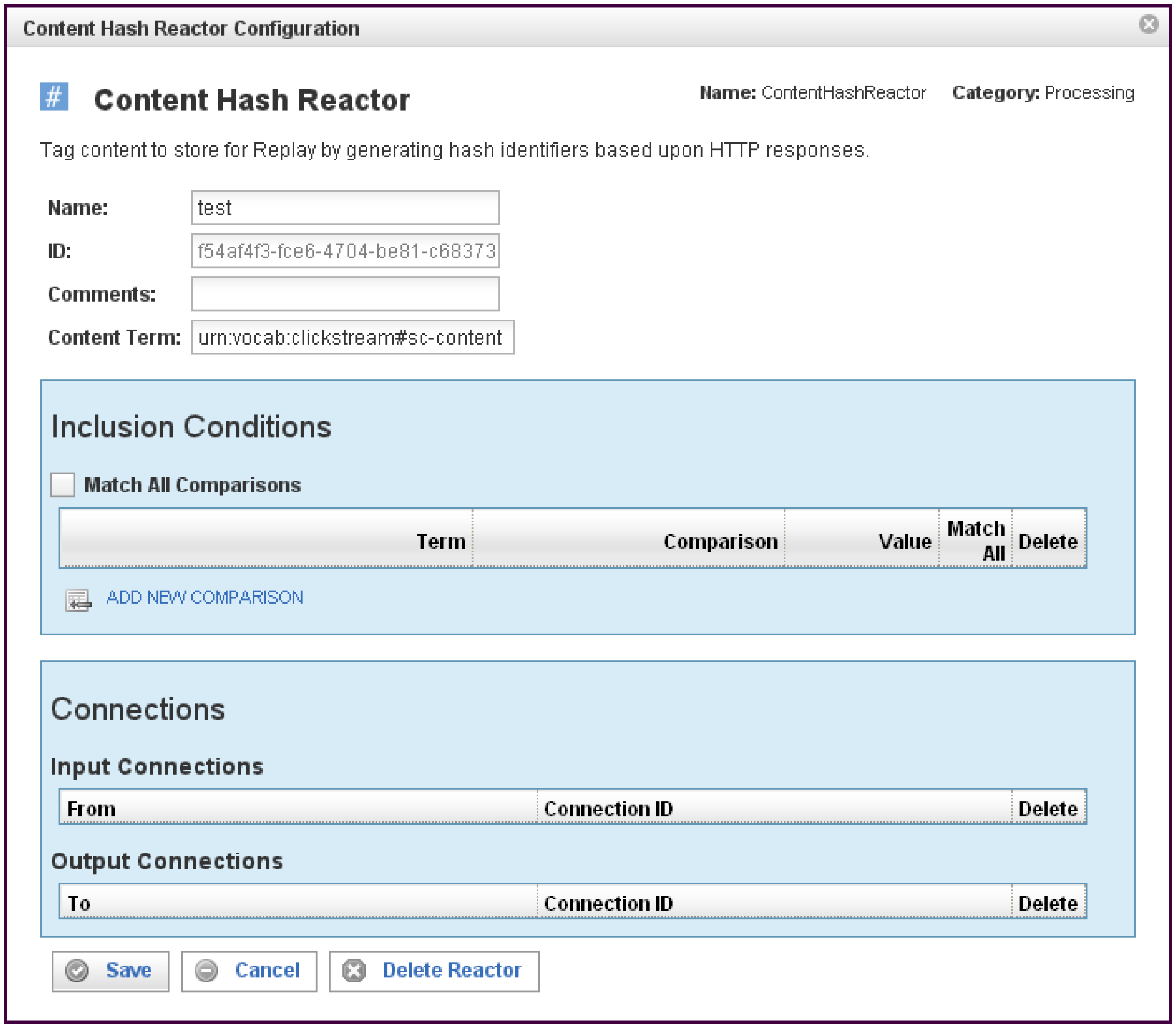
The Content Hash Reactor creates and then stores an md5 hash as an identifier (a content_id) for a specified content term in an HTTP response. This identifier is stored in a database, which is then used by the Replay feature when reviewing the user experience.

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 must be unique (within Content Hash Reactors) and is not case sensitive. 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 6 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 (checkbox)	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 (checkbox)	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.

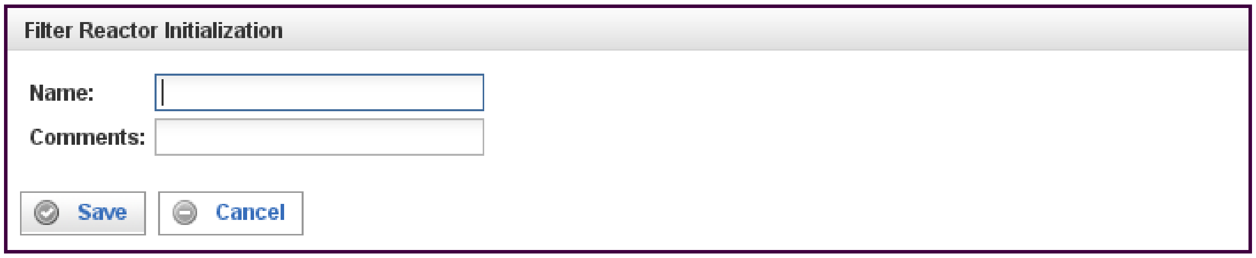
The Connections Section

The Connections section describes input and output connections for this Content Hash Reactor. See Reactor Input and Output Connections on page 7 for more information.

Filter Reactor

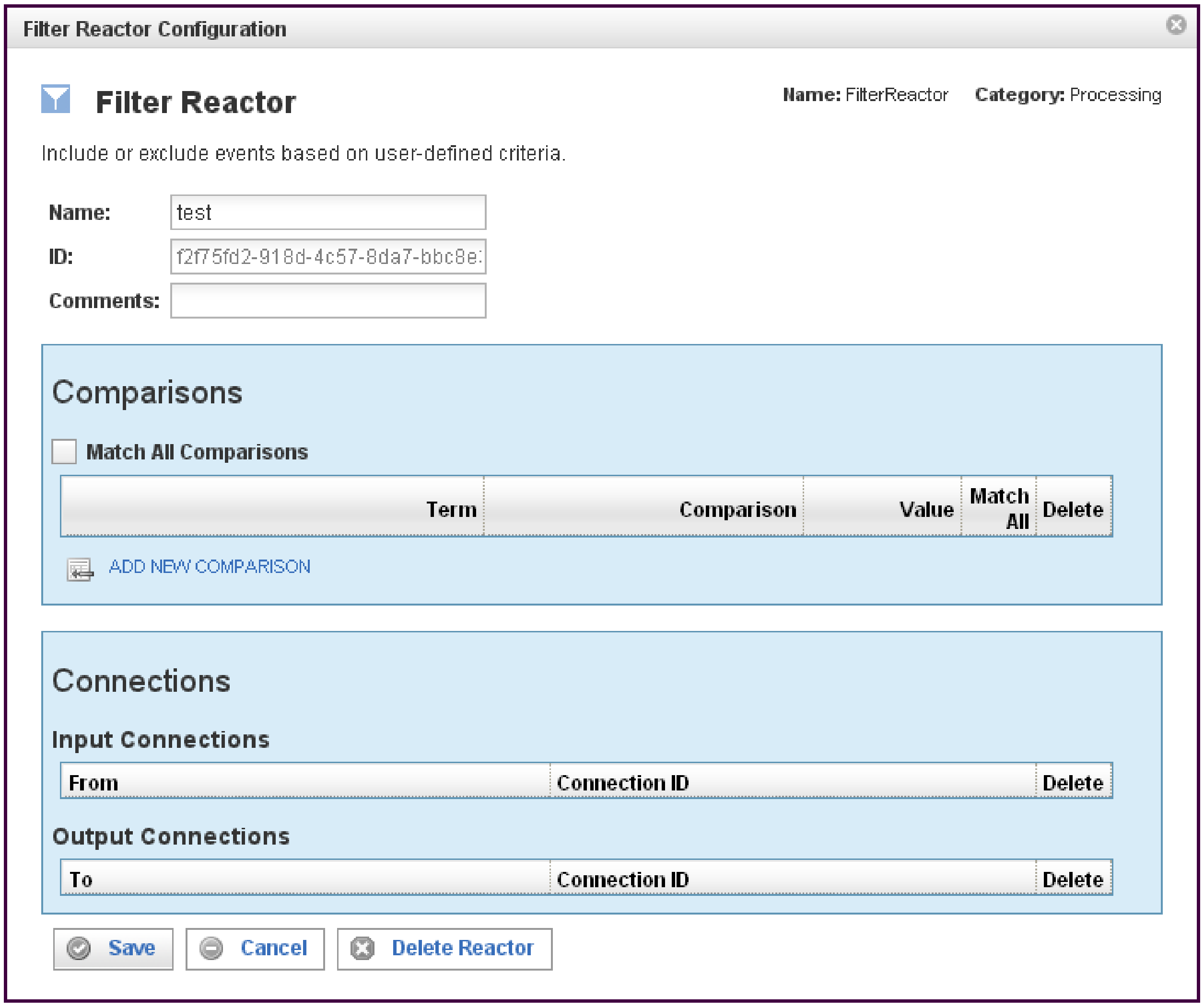
The Filter Reactor can filter in or out any information based on any number of criteria you might set. For example, you could do something simple like only looking at a particular set of users of web servers. Or you could do something much more complicated such as only tracking certain individuals who 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 must be unique (within Filter Reactors) and is not case sensitive. 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 6 for more information.

Field	Description
Match All (checkbox)	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 (checkbox)	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.

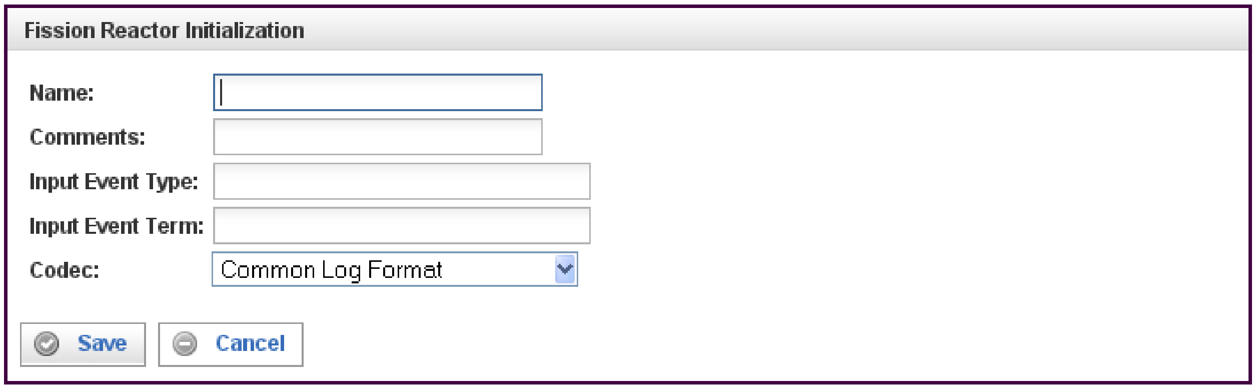
The Connections Section

The Connections section describes input and output connections for this Filter Reactor. See Reactor Input and Output Connections on page 7 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:

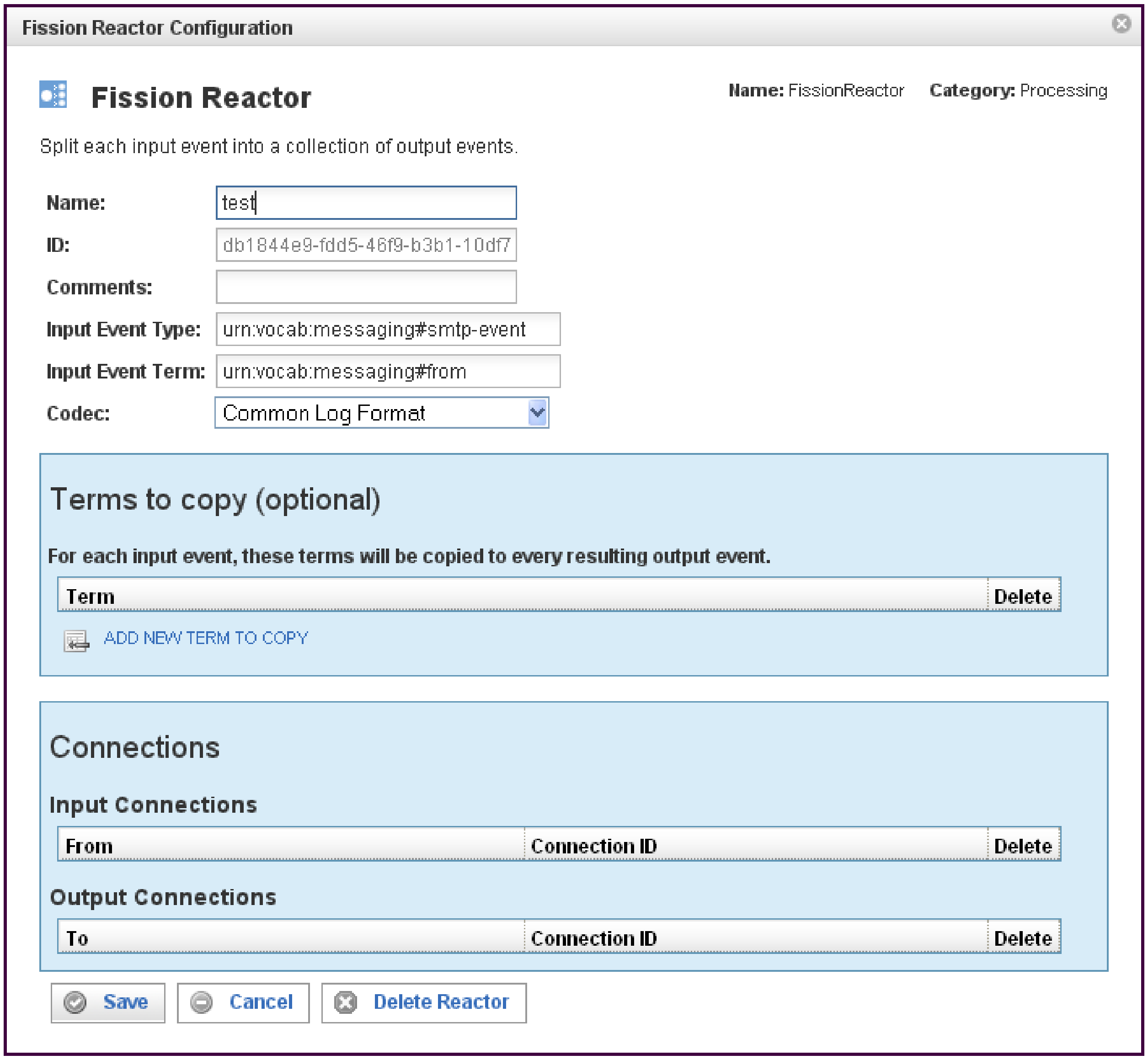


Fission Reactor Initialization

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 must be unique (within Fission Reactors) and is not case sensitive. 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.
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.

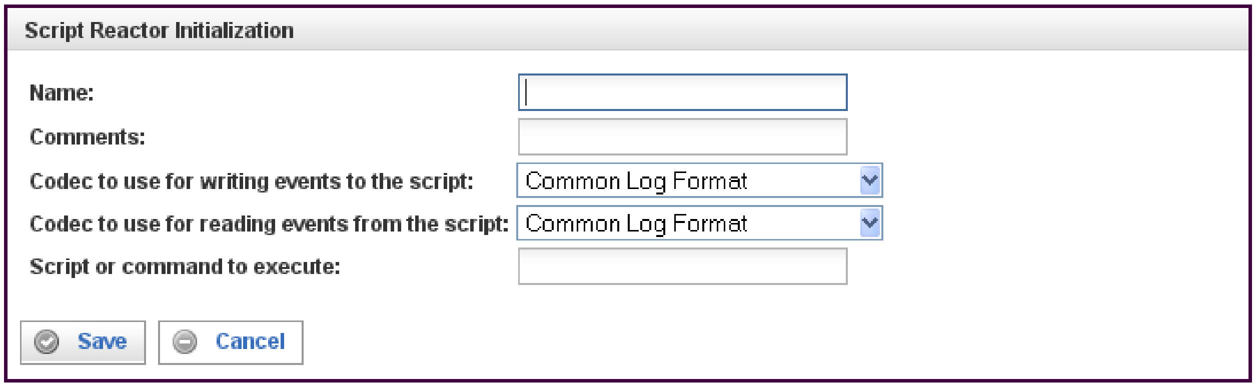
The Connections Section

The Connections section describes input and output connections for this Fission Reactor. See Reactor Input and Output Connections on page 7 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.

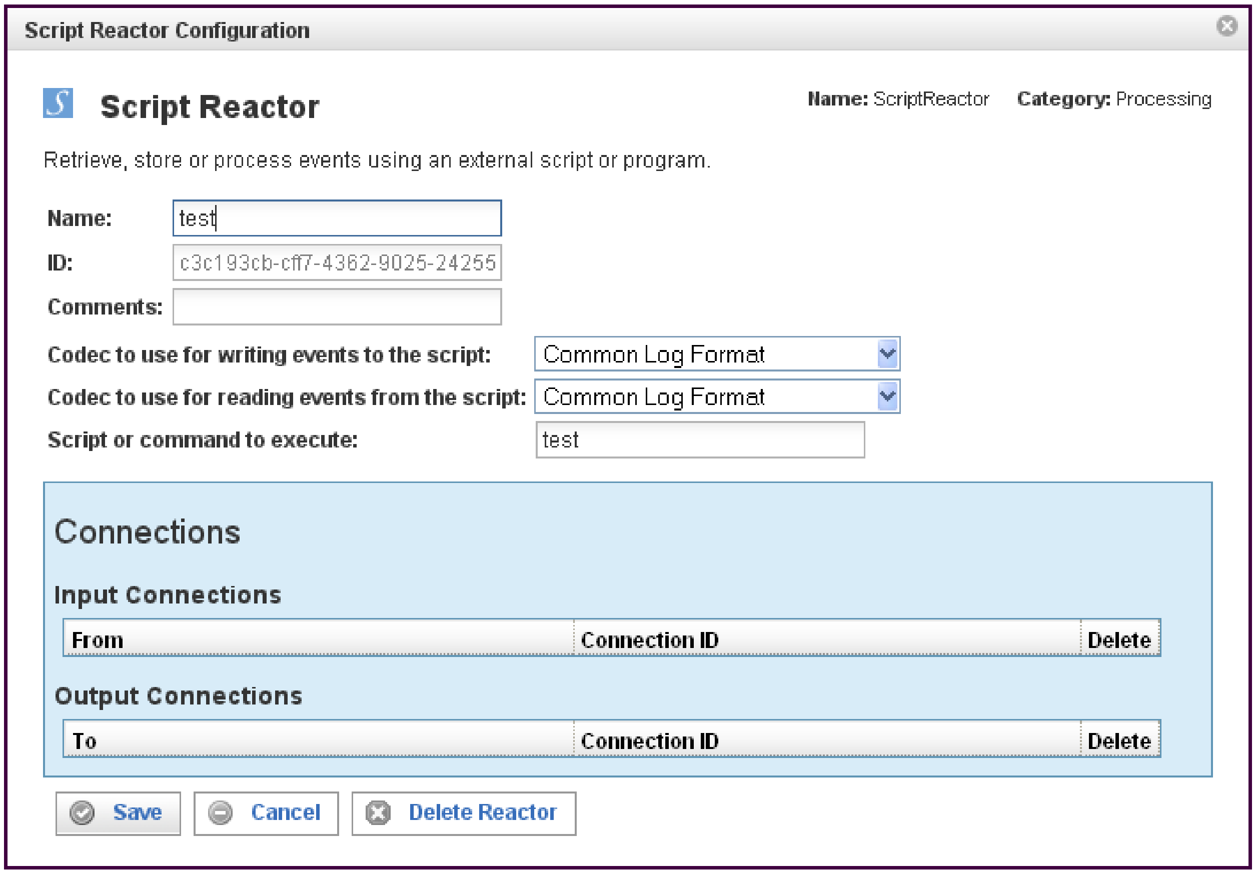
When you add a Script Reactor you will be prompted for the following fields:



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 must be unique (within Script Reactors) and is not case sensitive. 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.
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.

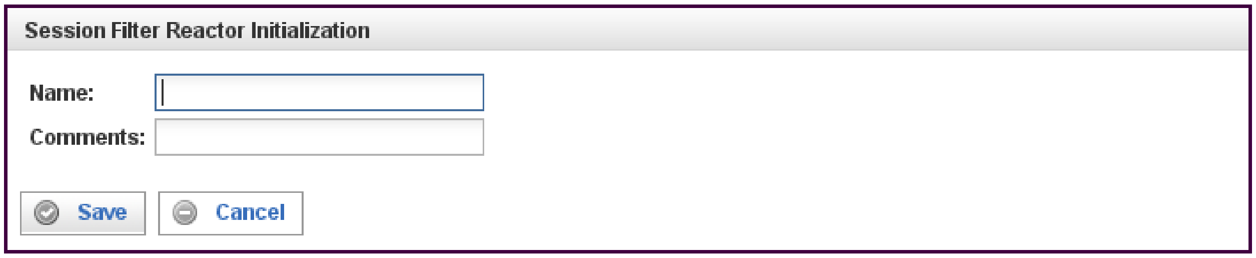
The Connections Section

The Connections section describes input and output connections for this Script Reactor. See Reactor Input and Output Connections on page 7 for more information.

Session Filter Reactor

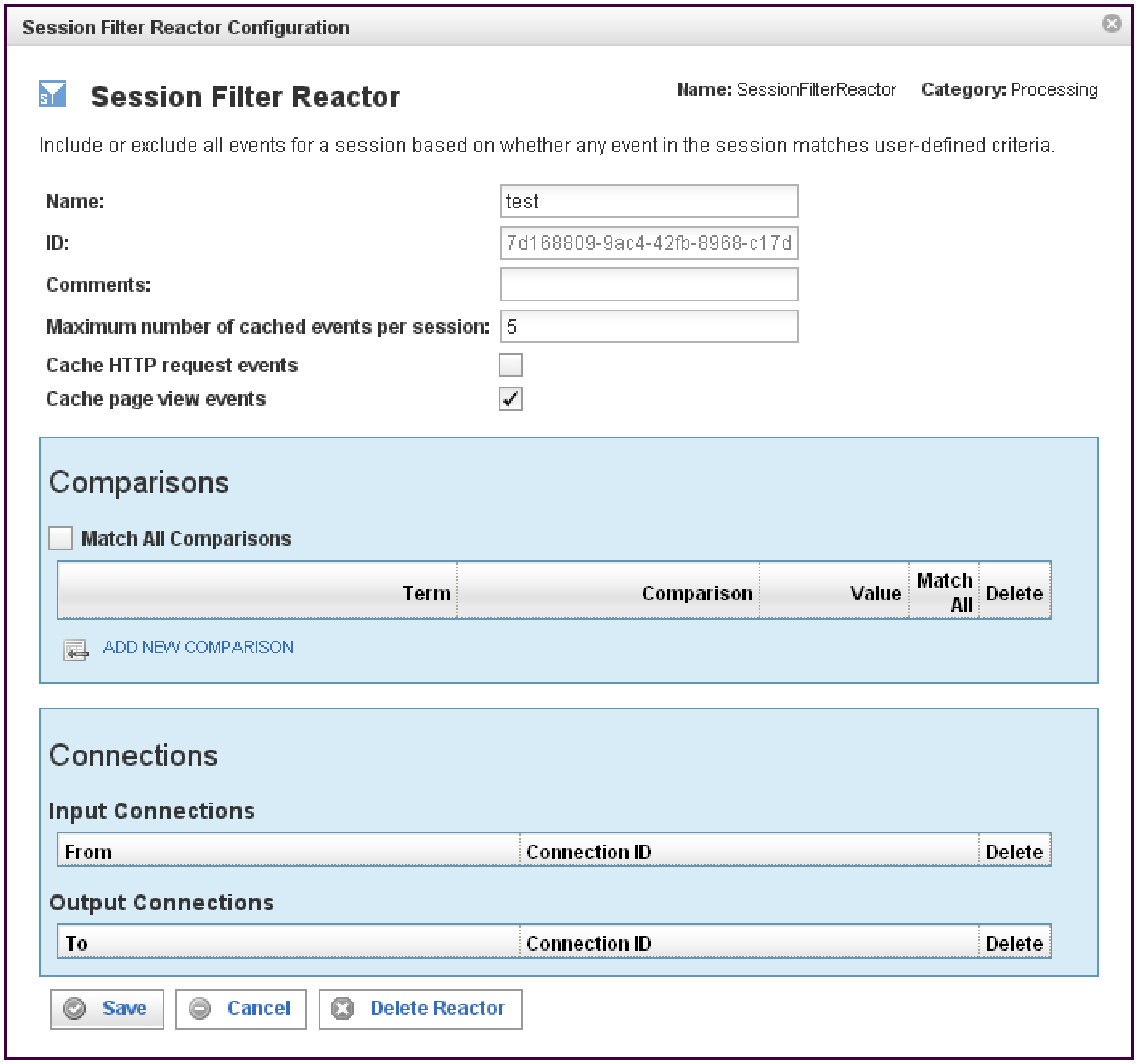
The Session Filter Reactor enables you to hold entire sessions until they meet certain criteria. The Session Filter Reactor must always be used after a Clickstream Reactor to ensure that the sessions are created. Using the Session Filter Reactor, you can do things such as saving full page view detail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



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 must be unique (within Session Filter Reactors) and is not case sensitive. 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.
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 (checkbox)	When the checkbox 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 (checkbox)	When the checkbox 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 6 for more information.

Field	Description
Match All (checkbox)	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 (checkbox)	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.

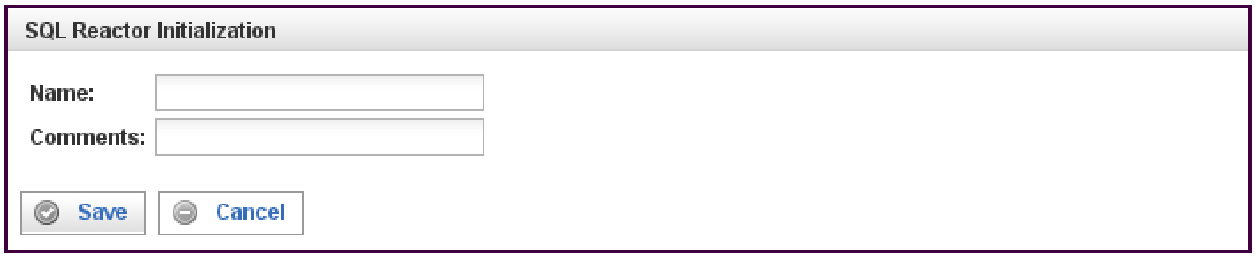
The Connections Section

The Connections section describes input and output connections for this Session Filter Reactor. See Reactor Input and Output Connections on page 7 for more information.

SQL Reactor

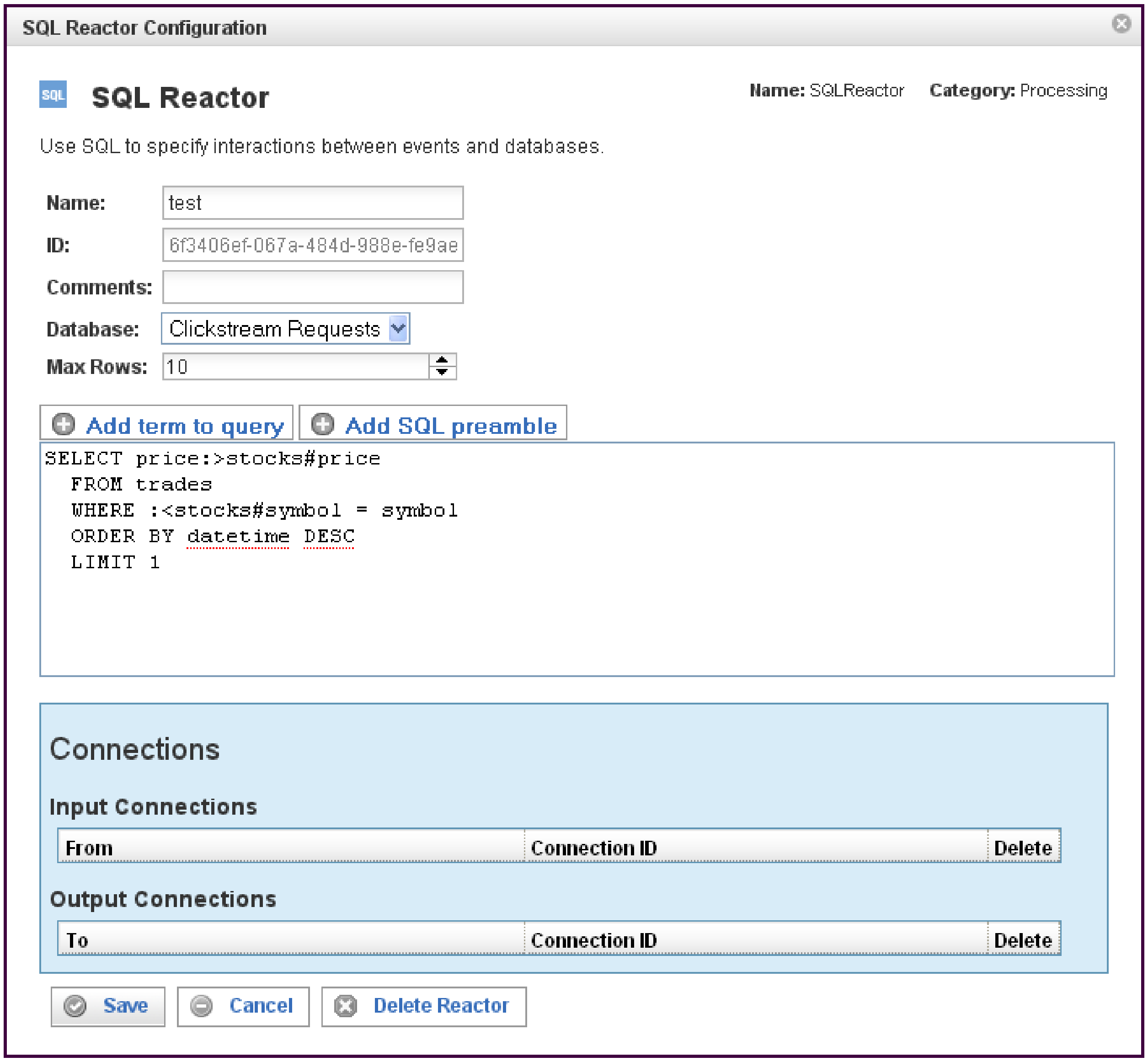
SQL Reactor, which lets Pion interact with any relational database based on real time information. This means that Pion can gather additional information on sessions based on past activity. For example, you can see how much a customer spent during the last 2 years to add context to their session and 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 must be unique (within SQL Reactors) and is not case sensitive. 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.
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`
	`   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 (":<"

Field	Description
	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: `"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

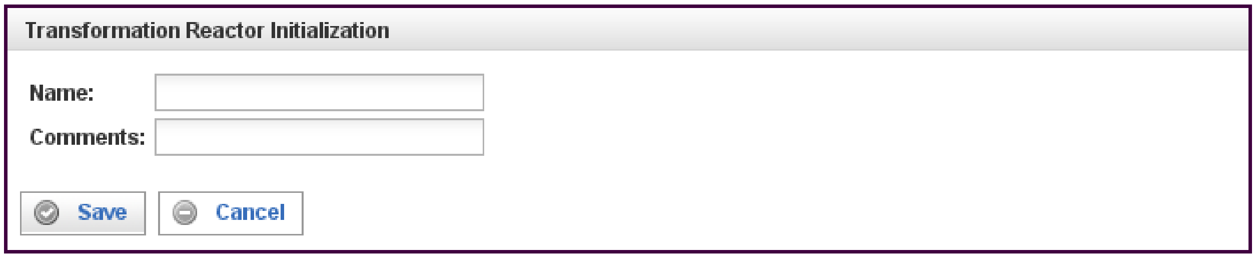
The Connections section describes input and output connections for this SQL Reactor. See Reactor Input and Output Connections on page 7 for more information.

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.

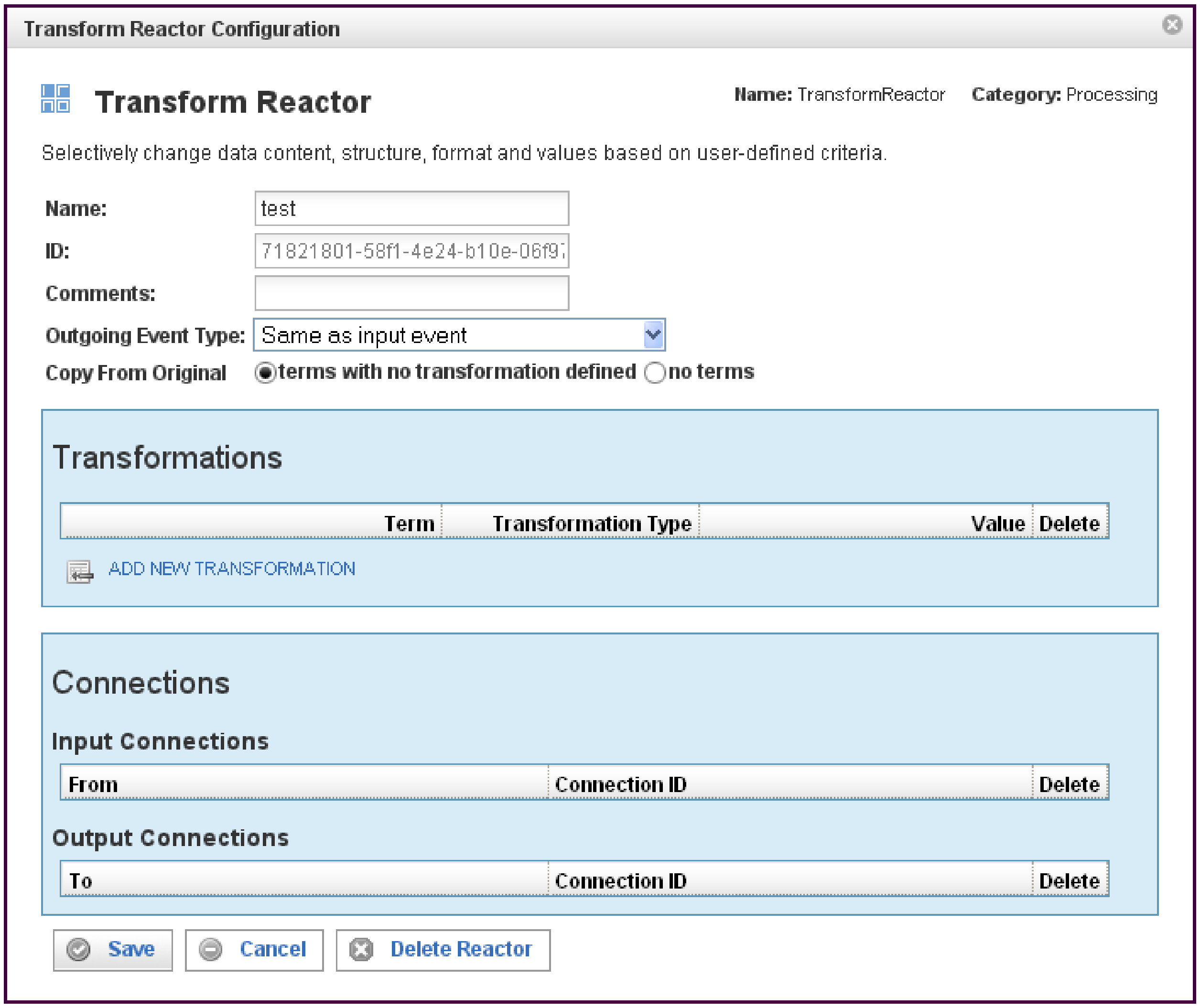
Note: An event processed by the Transform Reactor is transformed. The original (unmodified) event is not passed through. The Transform Reactor's only output is transformed events. If the original (unmodified) event is desired, make an additional connection from the reactor preceding the Transform Reactor 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 must be unique (within Transform Reactors) and is not case sensitive. 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



Transform Reactor Configuration

Transform Reactor

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

Transformations

Term	Transformation Type	Value	Delete

ADD NEW TRANSFORMATION

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 *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.
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). • 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, and SetValue defines the output format. **Note**: Not-RegExp assigns SetValue as a literal, since the regex failed.

Field	Description
	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. **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 7 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:

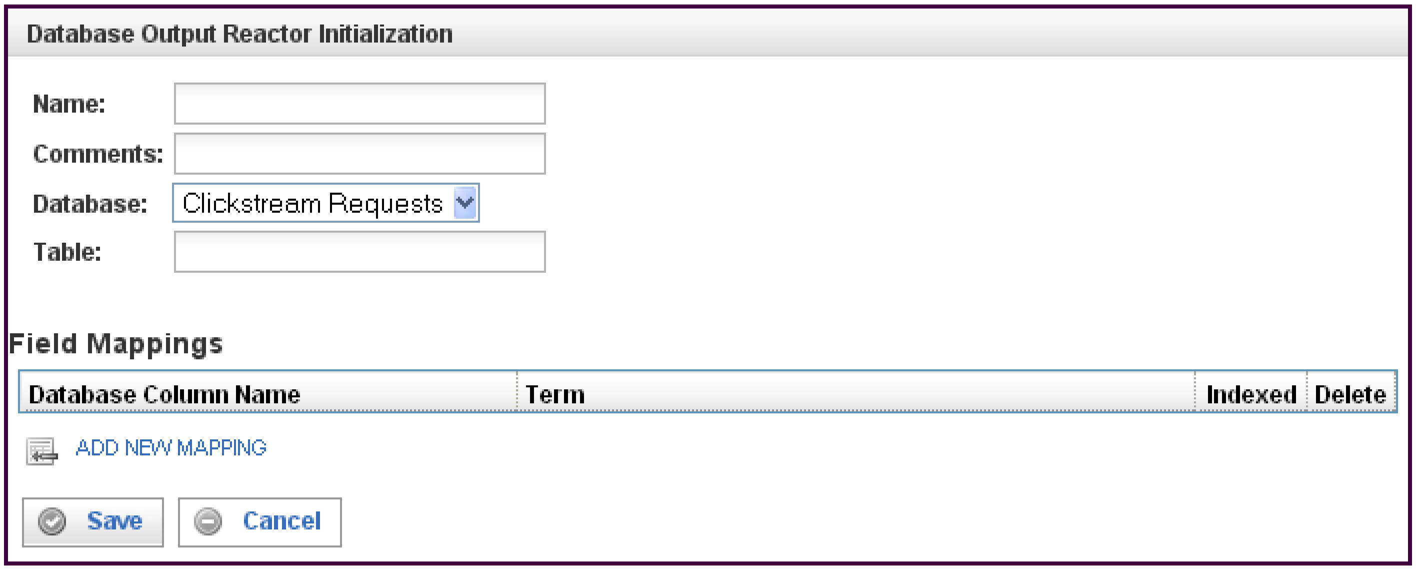
- **Database Output Reactor**, which enables Pion to store information in almost any database.
- **Google Analytics Reactor**, which is able to take web user behavior information and send it to Google Analytics servers via its native web services interface in real-time.
- **Log Output Reactor**, which 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

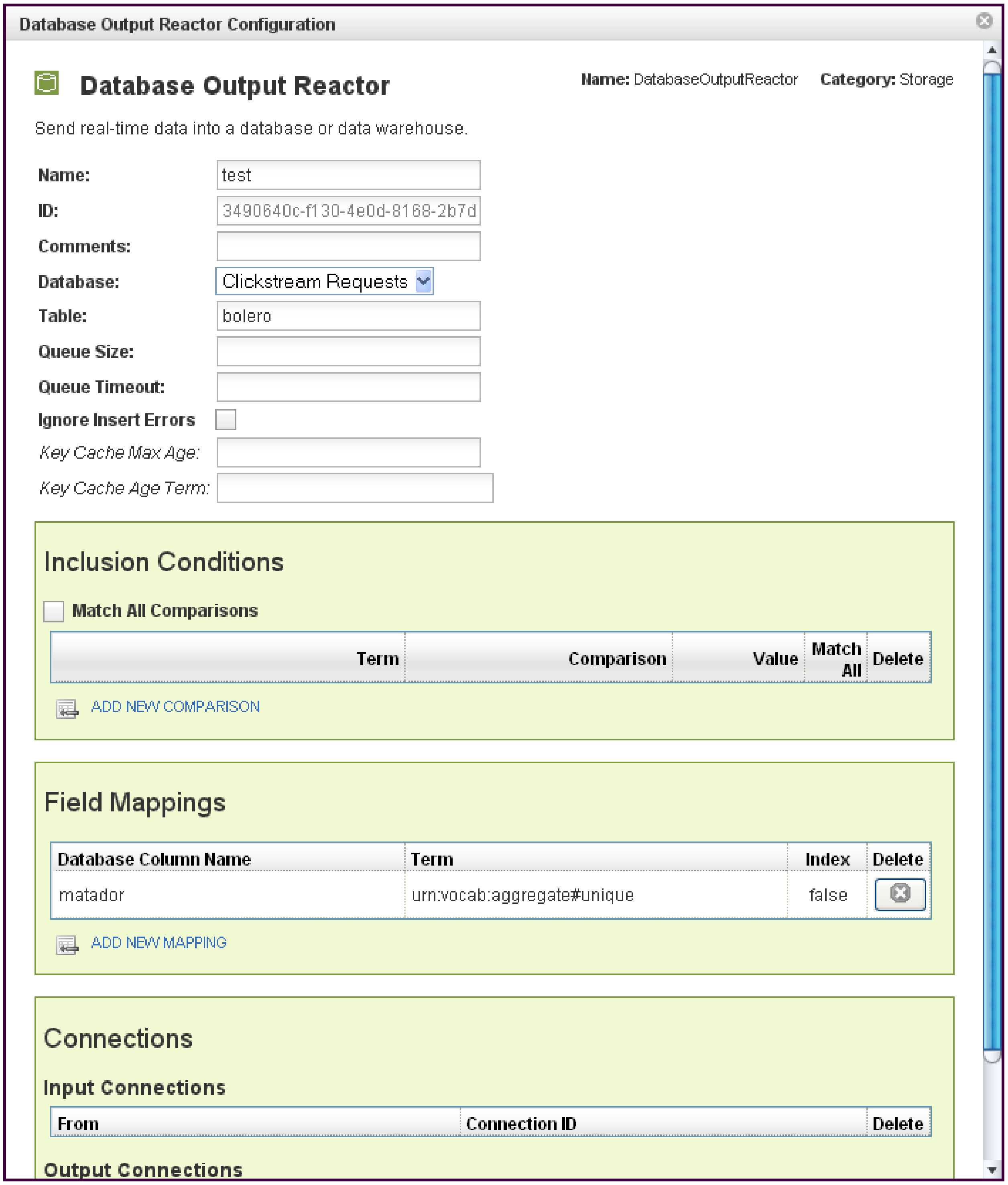
Name:	
Comments:	
Database:	Clickstream Requests
Table:	

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 must be unique (within Database Output Reactors) and is not case sensitive. 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.
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.

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:	3490640c-f130-4e0d-8168-2b7d
Comments:	
Database:	Clickstream Requests
Table:	bolero
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
matador	urn:vocab:aggregate#unique	false	☒

ADD NEW MAPPING

Connections

Input Connections

From	Connection ID	Delete

Output Connections

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 (checkbox)	Inserts may fail if records already exist within the table which have the same unique index values. Fill this checkbox 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 (checkbox)	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 (checkbox)	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.

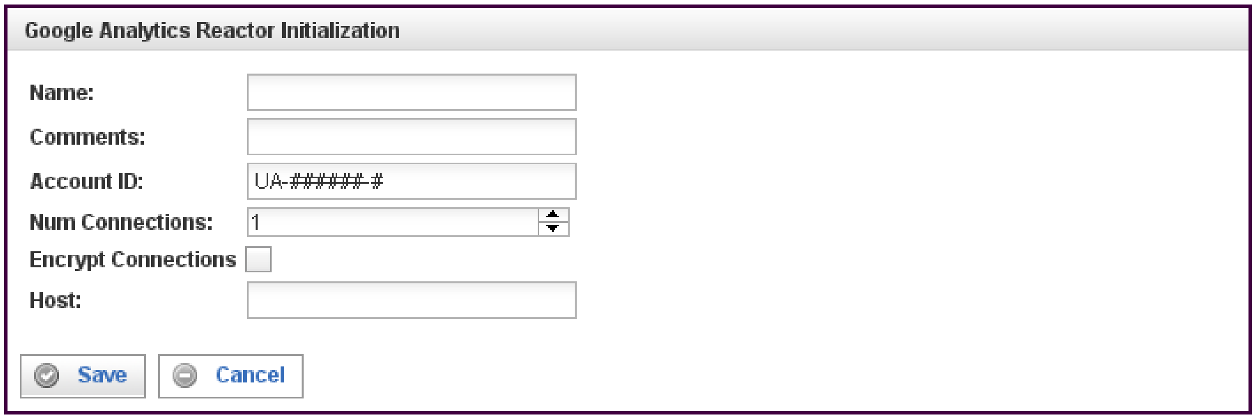
The Connections Section

The Connections section describes input and output connections for this Database Output Reactor. See Reactor Input and Output Connections on page 7 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.

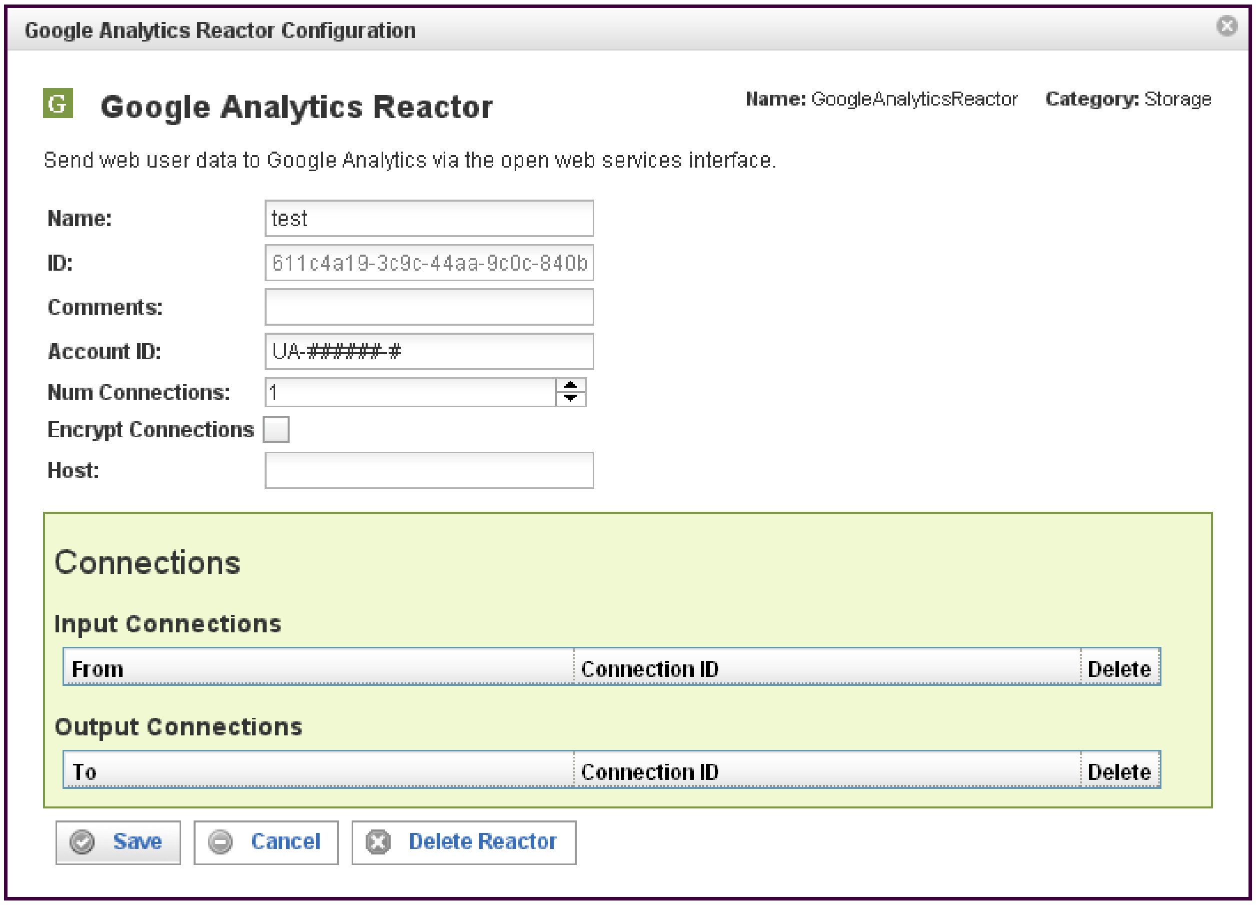
When you add a Google Analytics Reactor you will be prompted for the following fields:


Google Analytics Reactor Initialization

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 must be unique (within Google Analytics Reactors) and is not case sensitive. 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-"`
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 (checkbox)	Fill this checkbox to encrypt connections between Pion and the Google Analytics servers. Normally this checkbox 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.

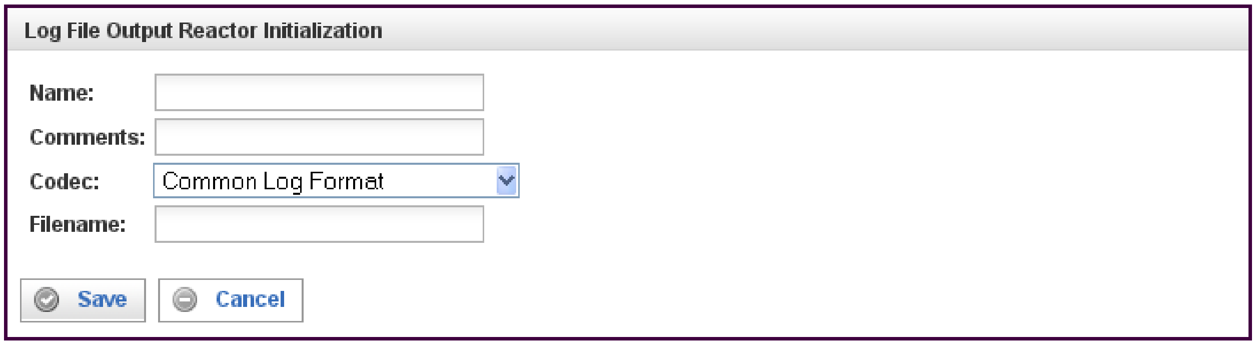
The Connections Section

The Connections section describes input and output connections for this Google Analytics Reactor. See Reactor Input and Output Connections on page 7 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:

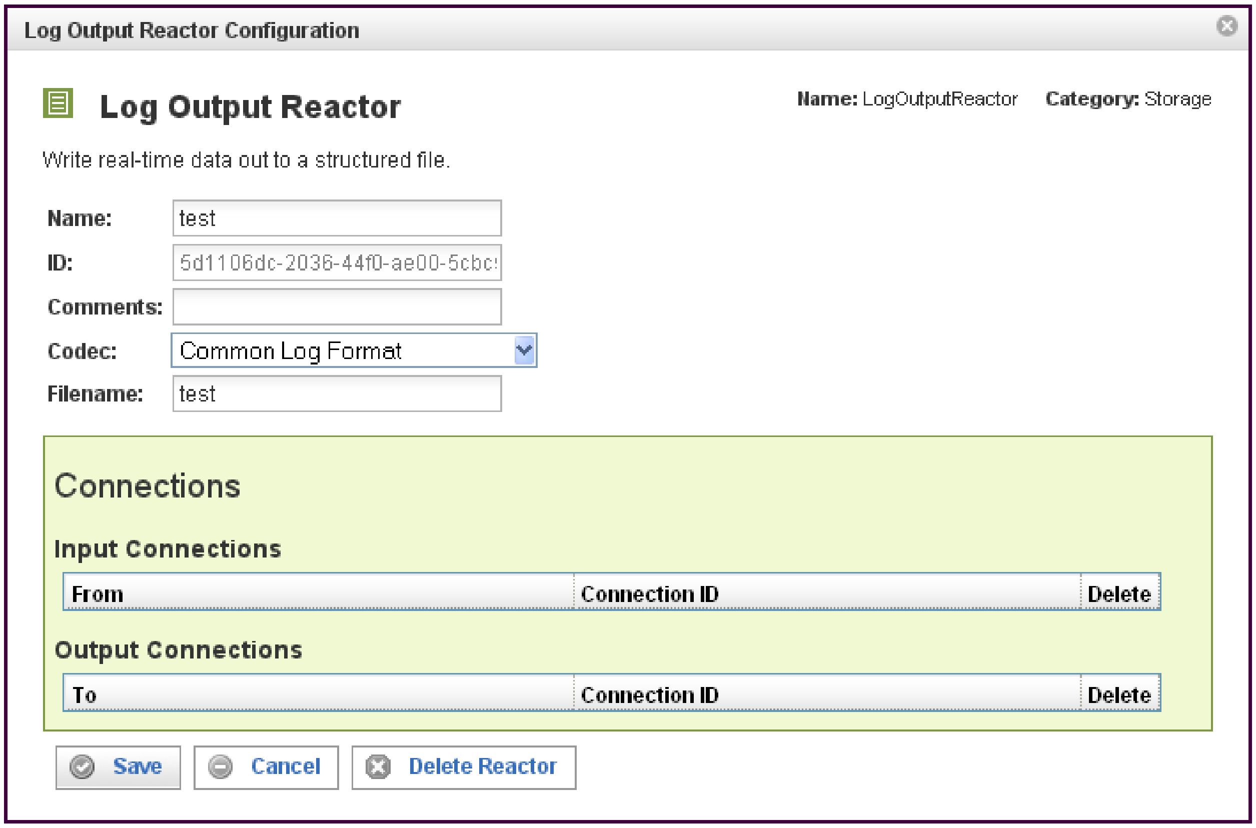
Comments:

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 must be unique (within Log File Output Reactors) and is not case sensitive. 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.
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.

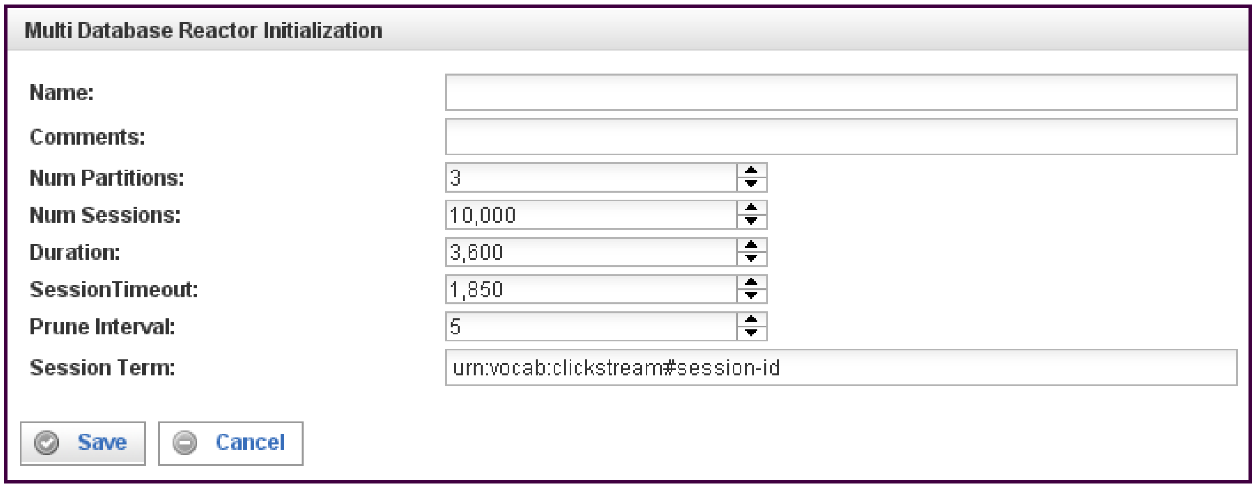
The Connections Section

The Connections section describes input and output connections for this Log Output Reactor. See Reactor Input and Output Connections on page 7 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.

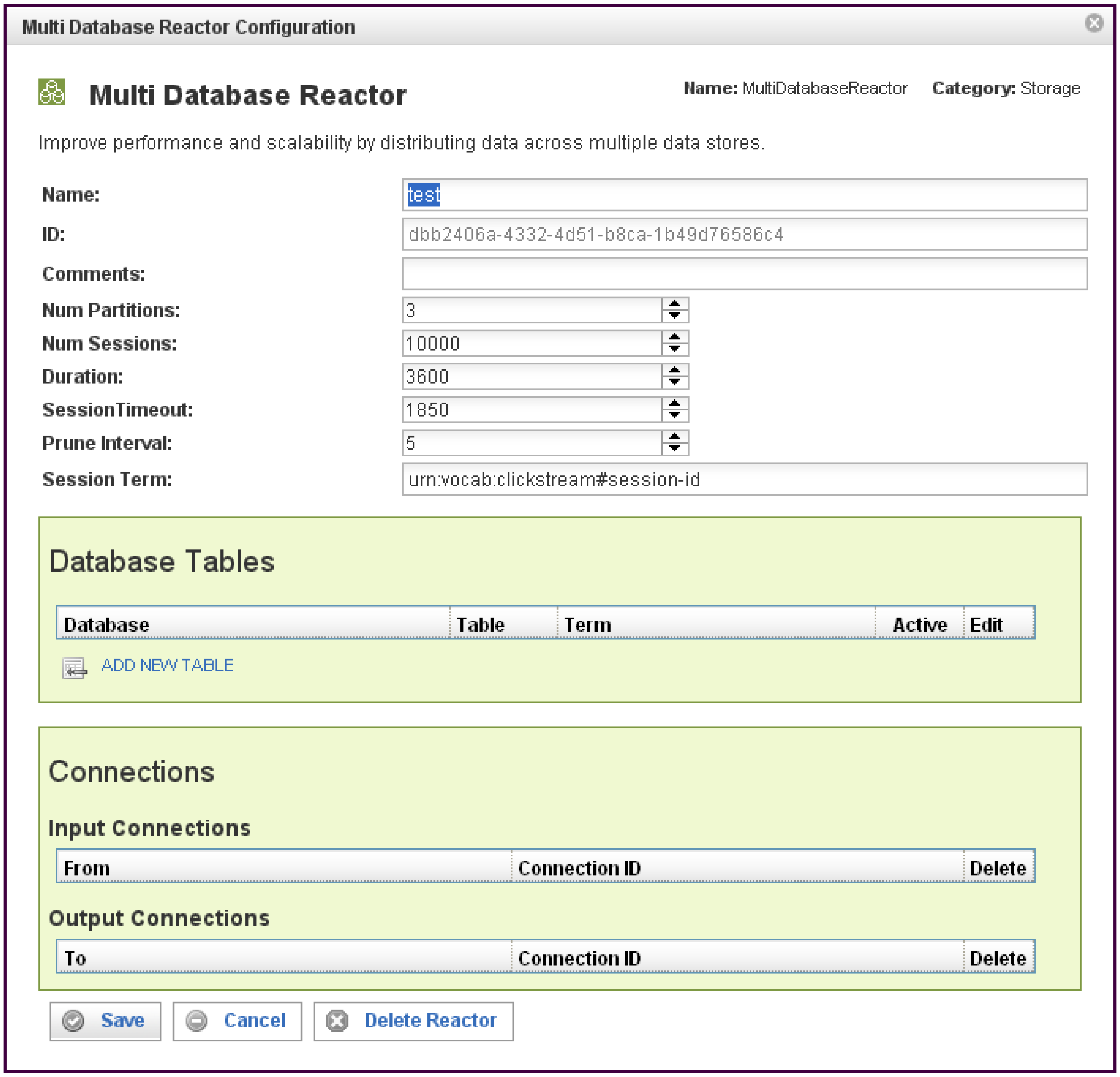
When you add a Multi Database Reactor you will be prompted for the following fields:



Name:	
Comments:	
Num Partitions:	3
Num Sessions:	10,000
Duration:	3,600
SessionTimeout:	1,850
Prune Interval:	5
Session Term:	urn:vocab:clickstream#session-id

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 must be unique (within Multi Database Reactors) and is not case sensitive. 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.
Num Partitions	Specifies the number of partitions to be used. The Multi Database Reactor uses rotating database partitions to guarantee performance for high-volume websites. The number of partitions defined here corresponds with the total amount of data that it will store (the more partitions, the longer period of time Replay will cover, and the more disk storage it will use). You must use a minimum of 3 partitions.
Num Sessions	The total number of sessions after which the Multi Database Reactor will rotate partitions.
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 Filter 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:	dbb2406a-4332-4d51-b8ca-1b49d76586c4
Comments:	
Num Partitions:	3
Num Sessions:	10000
Duration:	3600
SessionTimeout:	1850
Prune Interval:	5
Session Term:	urn:vocab:clickstream#session-id

Database Tables
Database | Table | Term | Active | Edit
ADD NEW TABLE
Connections
Input Connections
From | Connection ID | Delete
Output Connections
To | Connection ID | Delete
Save
Cancel
Delete Reactor

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.

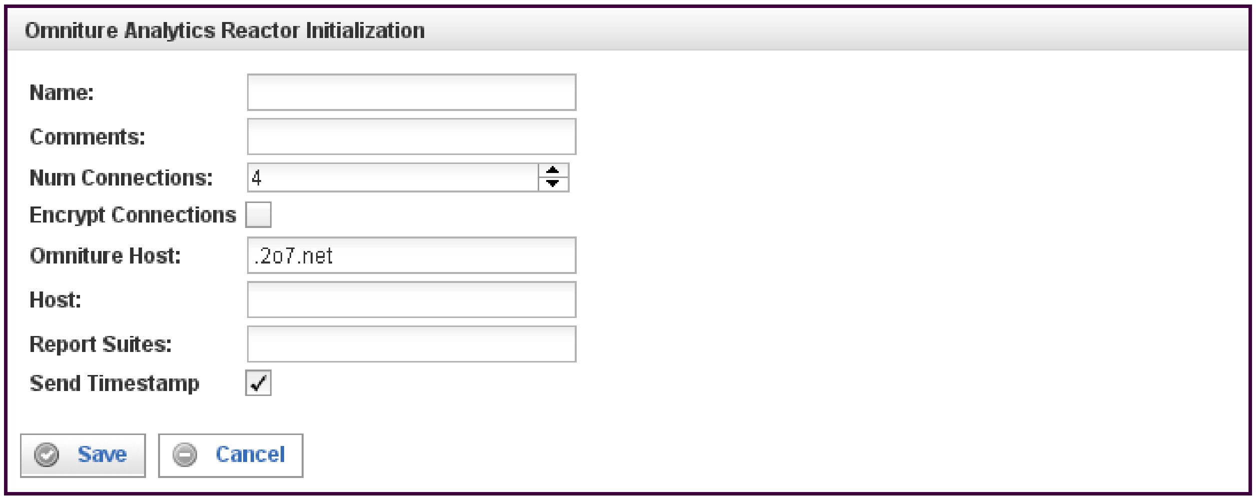
The Connections Section

The Connections section describes input and output connections for this Multi Database Reactor. See Reactor Input and Output Connections on page 7 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:



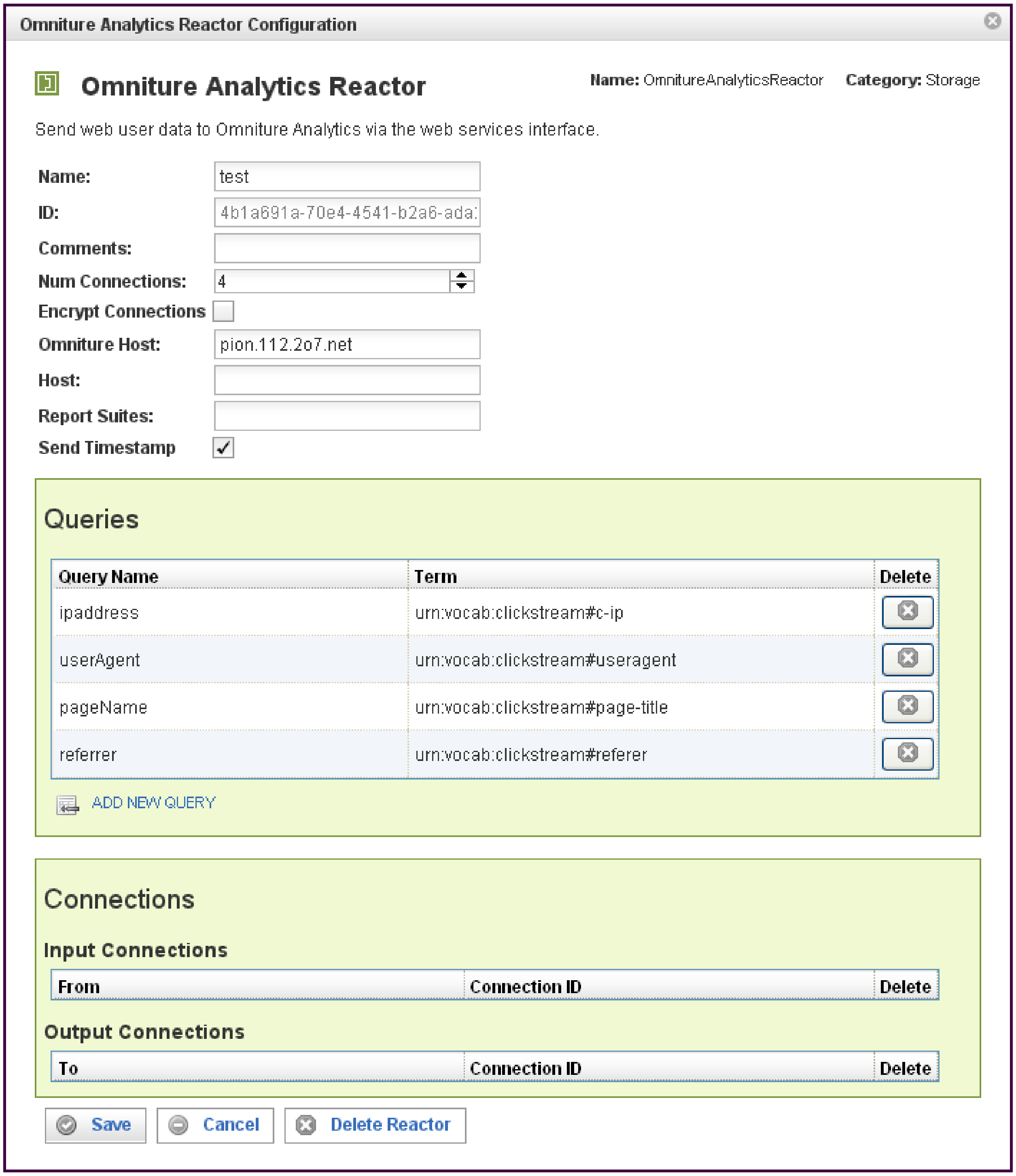
Omniture Analytics Reactor Initialization

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 must be unique (within Omniture Analytics Reactors) and is not case sensitive. 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.
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 (checkbox)	Fill this checkbox 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 (checkbox)	For Omniture accounts used with Pion, you must enable time stamps (this checkbox 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 checkbox 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	⊗

ADD NEW QUERY

Connections

Input Connections

From	Connection ID	Delete

Output Connections

To	Connection ID	Delete

Save Cancel Delete Reactor

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.
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 7 for more information.