Enterprise Edition

Overview

Last update: April 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. See http://en.wikipedia.org/wiki/Regex

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 and the Content Storage Reactor
- The Replay Feature

Reactor Changes

Some of our Reactors have been changed slightly:

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 Storage Reactor**, which stores page or request content into a database, along with a MD5 checksum (to store content only once). This Reactor provides the basis for the Replay Feature. See page 15 for more information about this new Reactor.

The Replay Feature

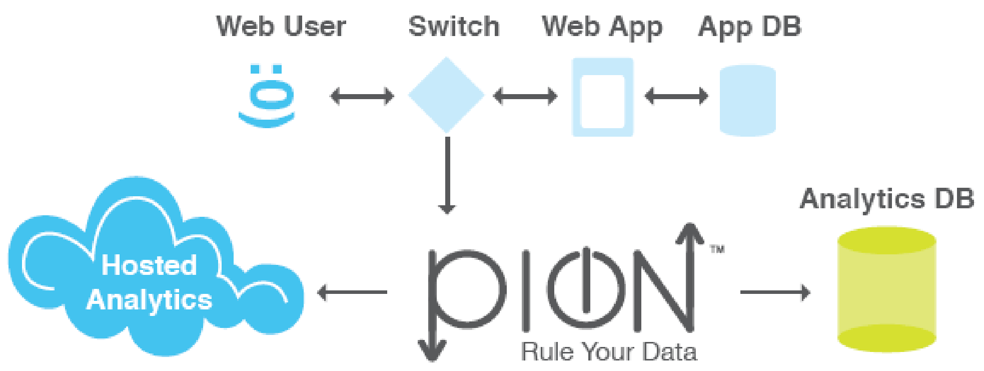
The Replay feature allows you to see what a visitor to your web site saw and “step through” their experience on your web site. The Replay feature captures the page views seen by a user, and **allows you to see the pages the y saw, as the user saw them**! Using Replay, you can also display information 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.

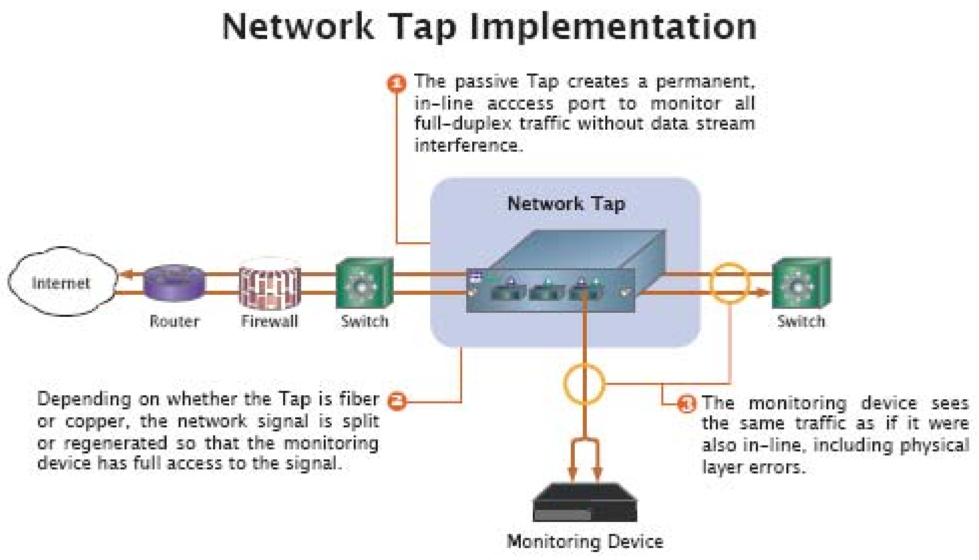
Pion Sniffer Placement

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.

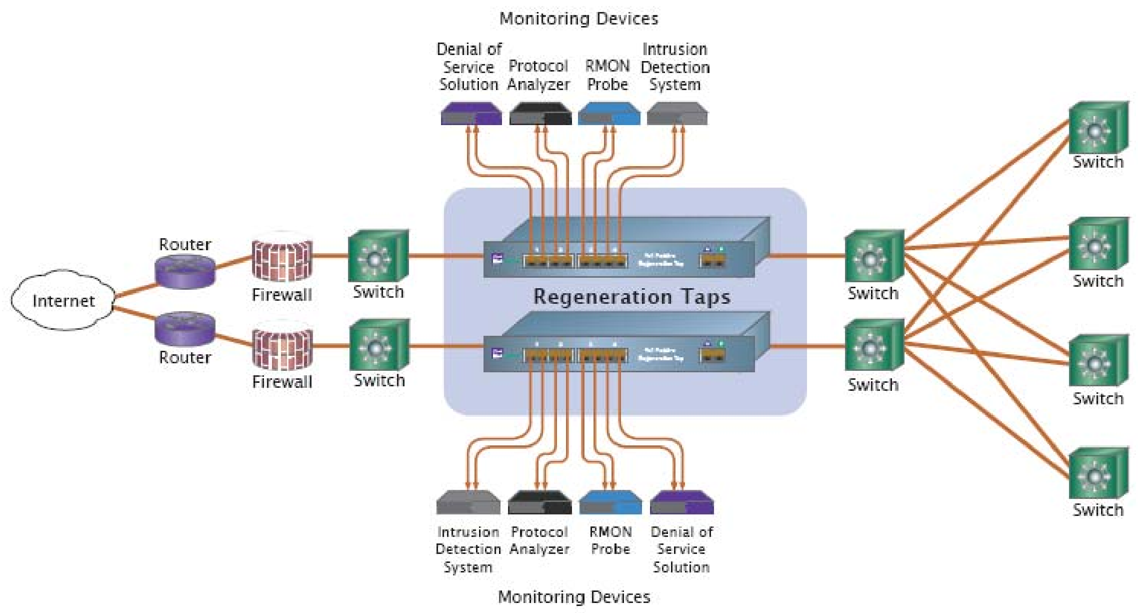
Using Network Taps and Spans

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:

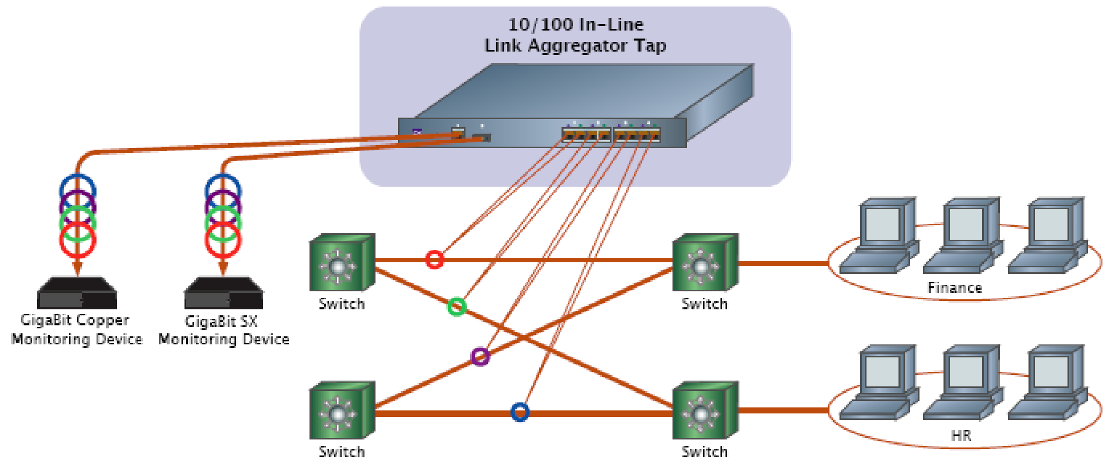
- **Spanning/mirroring**, which is implemented by configuring the switch to duplicate a stream of traffic from one port(s) to another. You connect one of the sniffing NIC(s) on the Pion server to the physical port(s) on the switch to which the duplicate stream is sent.
- **Network taps**, which fall 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
Descript ion	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 446	DB2 CLI	Pion → DB

Data Output Channels	Port	Protocol	Direction
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:

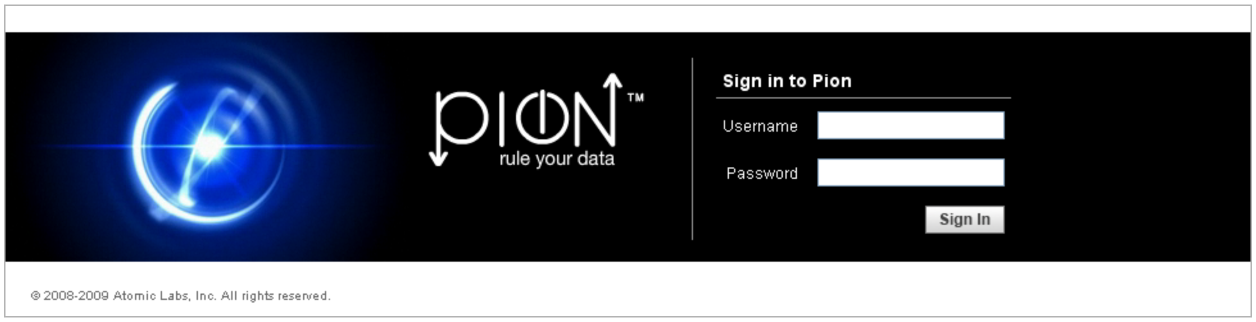
- **One-way communication for the Pion Sniffer Reactor**. The Pion Sniffer Reactor cannot communicate with anything other than the Pion server application under any circumstances. It is completely passive and is only fed one-way traffic. Because of this, Pion is more secure than any of the web servers 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.

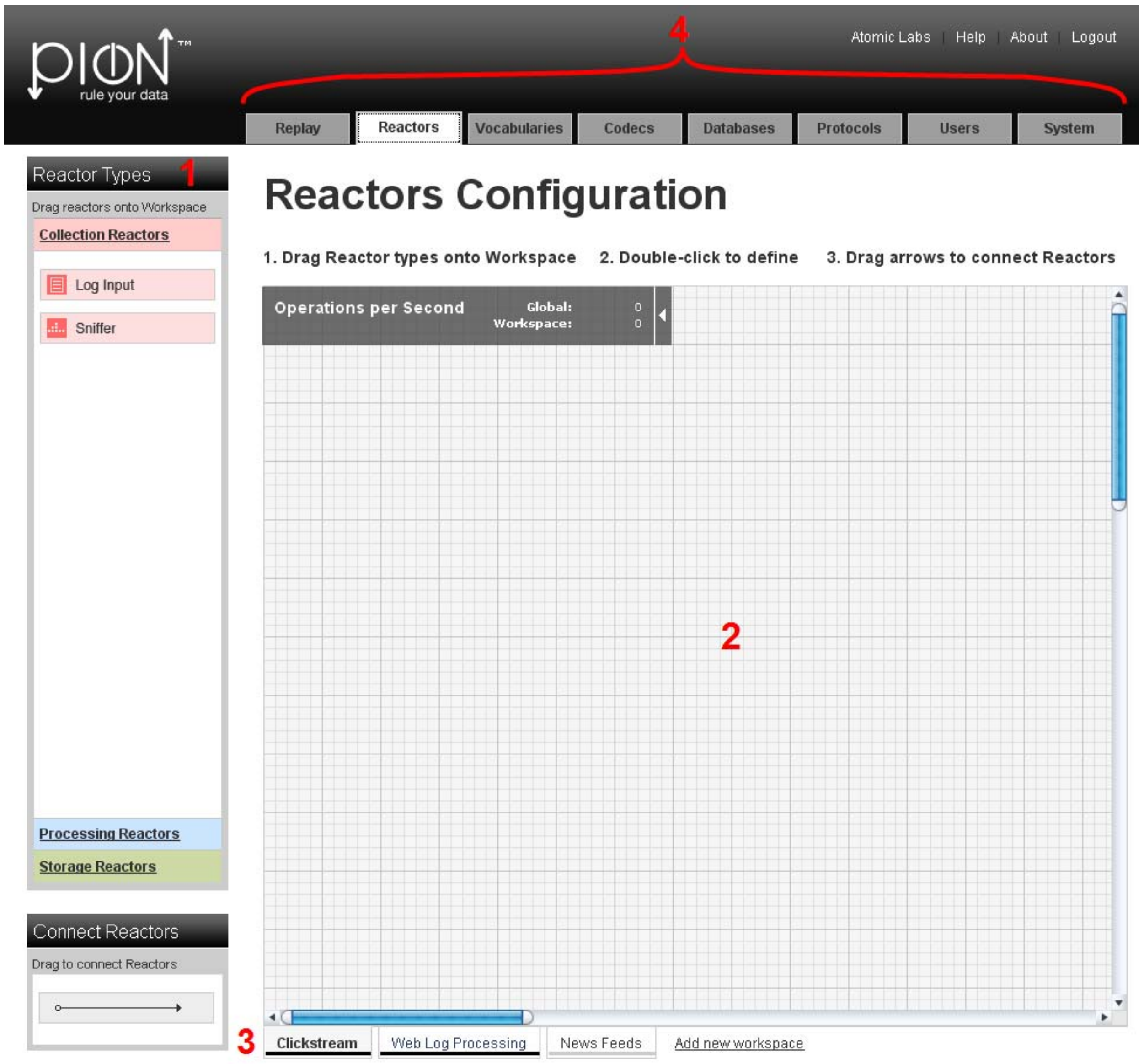
Getting Started with Pion

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 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 Storage Reactor**, which stores page or request content into a database, along with a MD5 checksum (to store content only once). The Replay feature uses the Content Storage Reactor and the database to retrieve and display what users saw. The Replay feature can retrieve graphics and other content directly from the live site, while getting only the page content from the Content Storage Reactor, or the system can be configured to store every request content (be careful about how big the CSR database can get).

- **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.

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.

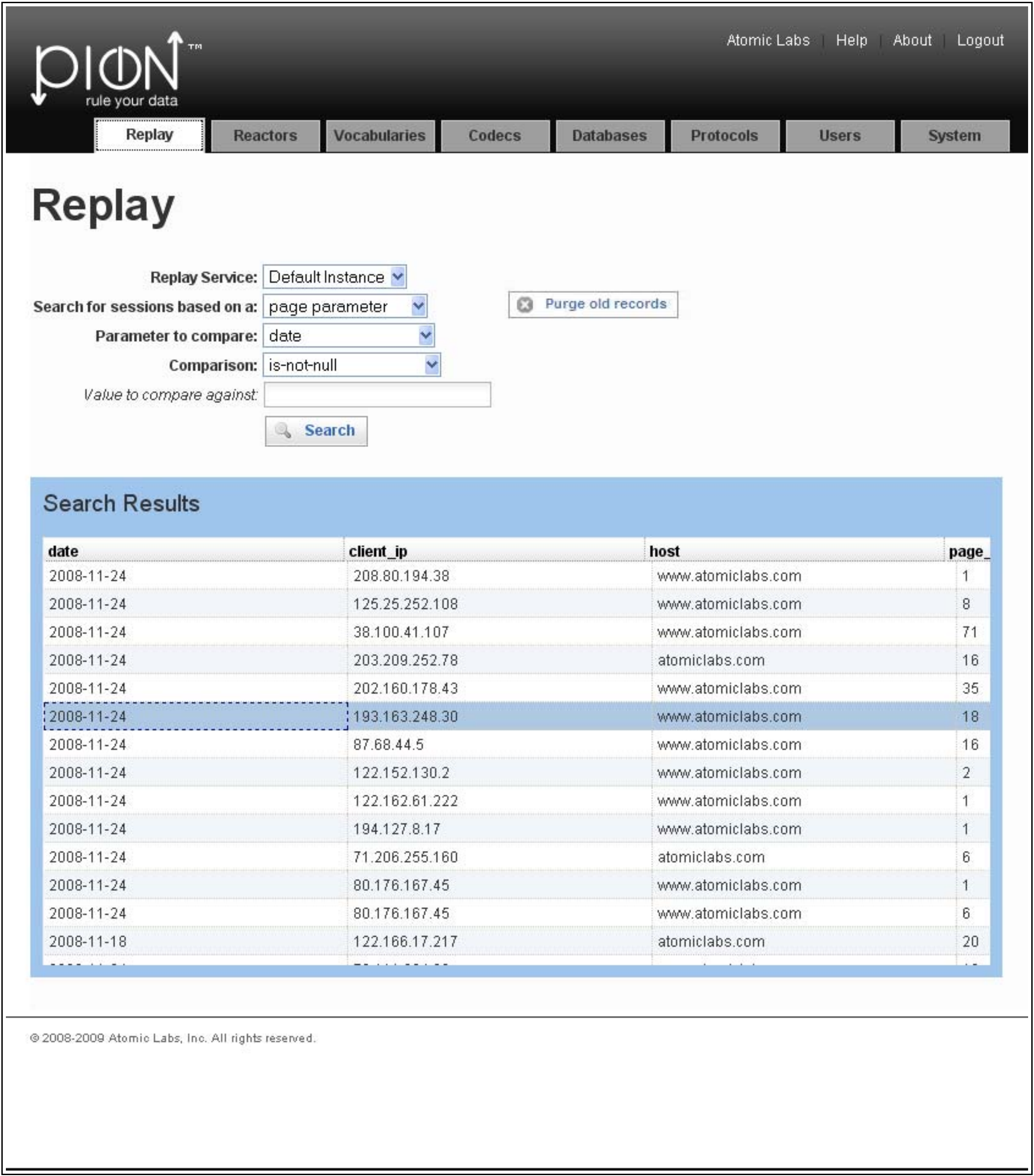
The System Configuration page is also where you can add a Reactor, Connection, codec, or database by importing the XML definition of the Reactor, Connection, codec, or database.

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:	

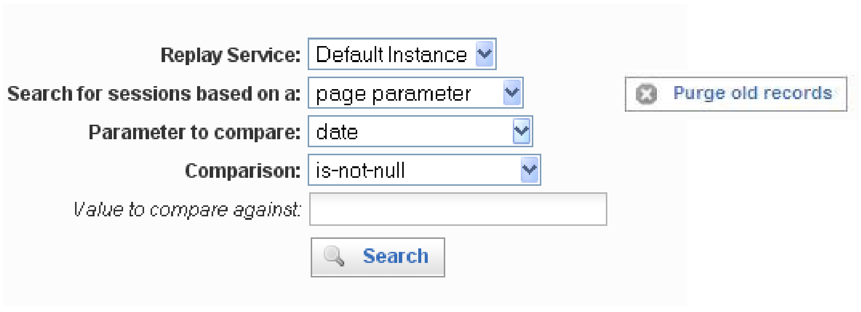
Search Results

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:

a) 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.

b) 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.)

c) 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.

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

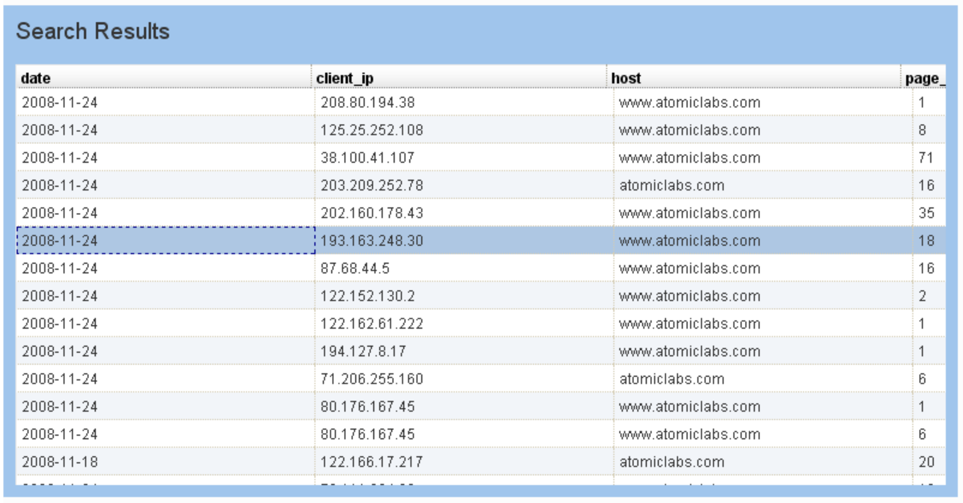
   Note: The **Value to compare against** field is disabled for comparisons for which it is not applicable, such as **is-null** and **is-not-null**.

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

5. 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).

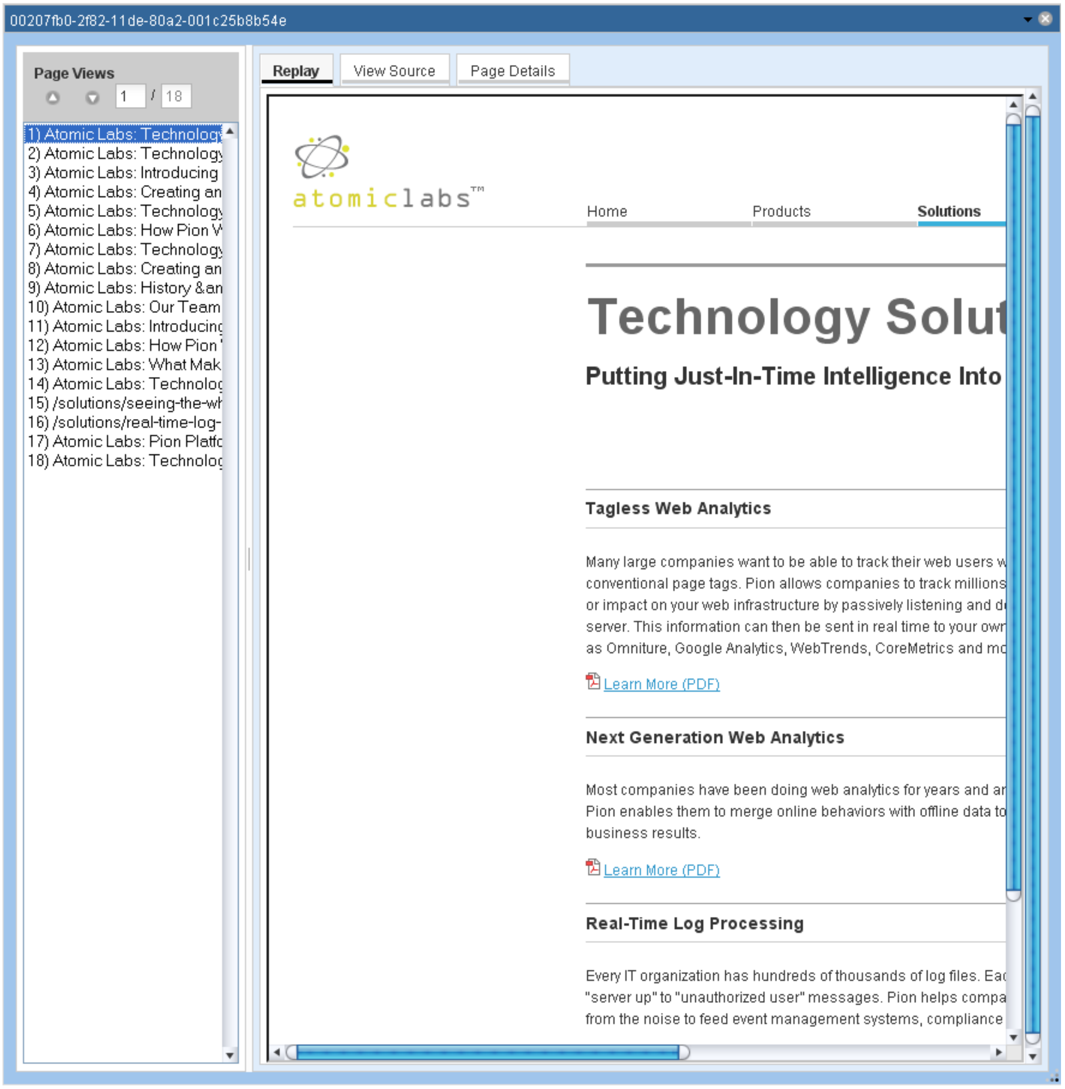
   The search results, if any, are displayed below the search criteria area.



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

6. 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 Platfo
18) Atomic Labs: Technolog

Replay | View Source | Page Details

atomiclabs™

Home | Products | Solutions

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

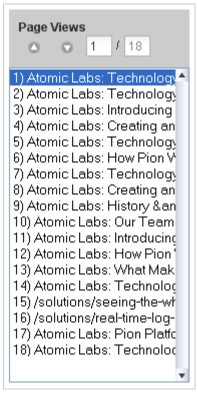
Most companies have been doing web analytics for years and ar
Pion enables them to merge online behaviors with offline data to
business results.

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.

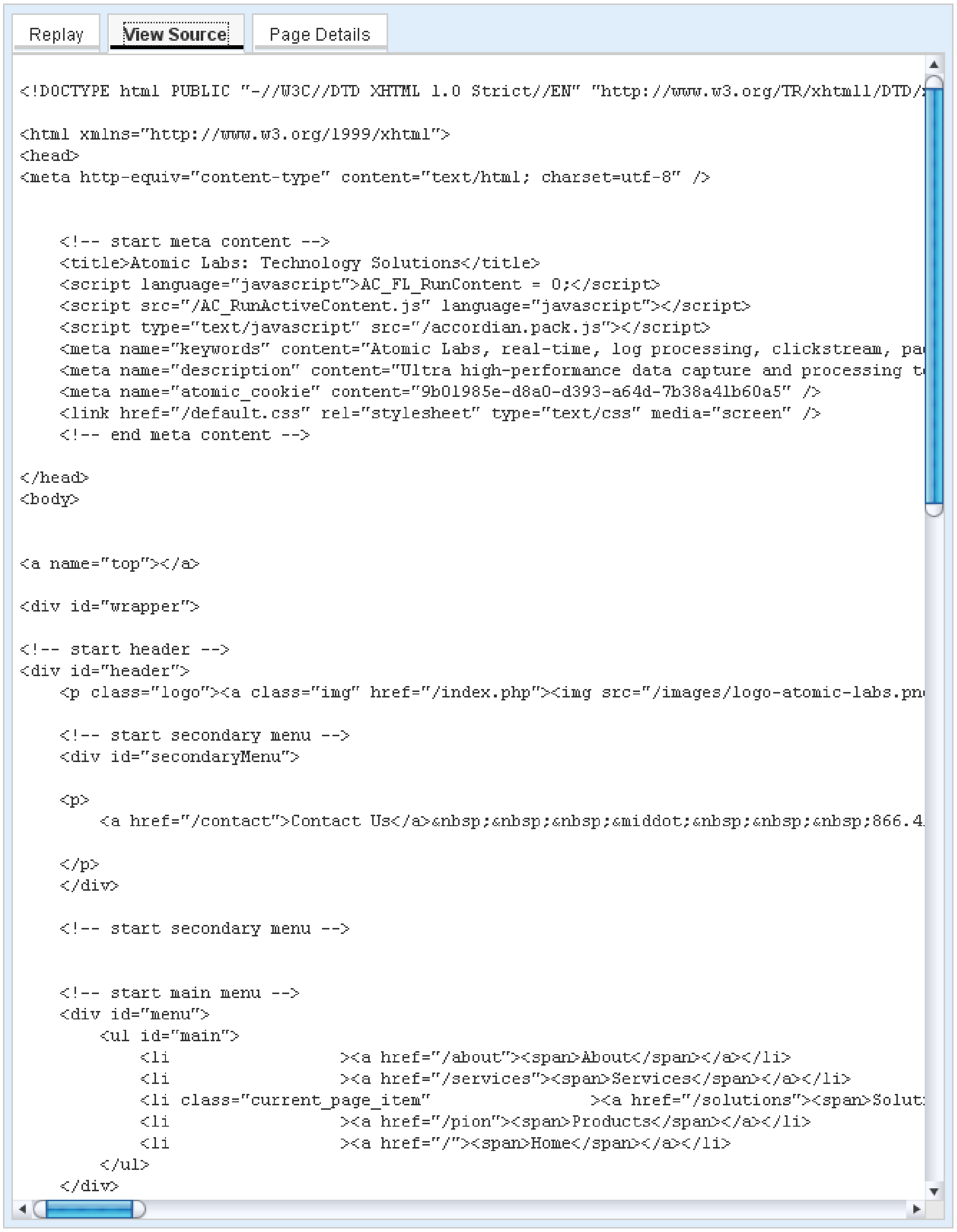


Page Views

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 Platfo
18) Atomic Labs: Technolog

- **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.