*Florida International University*

*School of Computing and Information Sciences*

CIS 4911 - Senior Capstone Project

Software Engineering Focus

Final Deliverable

Addigy

IT Intelligence at Scale

Team # X

**Team Members**

Matthew Saunders

Ayme Morrina

**Product Owner**: Jason Dettbarn

**Instructor**: Masoud Sadjadi

All Addigy specific logos and slogan material is copyright Addigy, Inc. All other materials developed for this project are available open source under MIT License:

<http://opensource.org/licenses/MIT>

***Abstract***

The market for IT monitoring tools has been growing in the past years, but there is still a lack of robust IT alerting tools for Mac computers. This study describes and evaluates the updated design of an IT monitoring solution and the visualization of the data it contains. To assist with the visualization, a RESTful API will be developed for processing the data in the cloud. Along with the current state, users will now be able to show historical information and timeline trends of the systems in their network. Data storage will be improved by implementing a NOSQL database in the cloud for data storage instead of json files.

The cloud implementation of storage allows for access from multiple end systems with data storage in a centralized location. A NOSQL database simplifies storage and incorporates a date time stamp for processing historical records and trends. The RESTful API improves transmission costs by processing the data being sent to a user, eliminating unnecessary data from being sent.

**Table of Contents**

Introduction................................................................................................................................... 5

Current System............................................................................................................................. 5

Purpose of New System............................................................................................................... 5

User Stories.................................................................................................................................. 7

Implemented User Stories............................................................................................................ 7

Pending User Stories.................................................................................................................. 19

Project Plan................................................................................................................................ 19

Hardware and Software Resources............................................................................................ 19

Sprints Plan................................................................................................................................ 21

Sprint 1....................................................................................................................................... 21

Sprint 2....................................................................................................................................... 24

Sprint 3....................................................................................................................................... 28

Sprint 4....................................................................................................................................... 31

Sprint 5....................................................................................................................................... 34

System Design........................................................................................................................... 41

Architectural Patterns................................................................................................................. 41

System and Subsystem Decomposition..................................................................................... 41

Deployment Diagram.................................................................................................................. 43

Design Patterns.......................................................................................................................... 43

System Validation....................................................................................................................... 44

Glossary..................................................................................................................................... 51

Appendix..................................................................................................................................... 52

Appendix A - UML Diagrams...................................................................................................... 52

Static UML Diagrams.................................................................................................................. 52

Dynamic UML Diagrams............................................................................................................. 53

Appendix B - User Interface Design........................................................................................... 62

Appendix C - Sprint Review Reports.......................................................................................... 70

Appendix D - Sprint Retrospective Reports................................................................................ 73

References................................................................................................................................. 74

# 

# 

# **Introduction**

The rapid growth in the IT monitoring market imposes the need for robust tools not only to monitor the current state of systems but to show historical information and timeline trends of the systems in their network.

The following sections provide a description of the characteristics and limitations of the current system as well as a description of a proposed system that will address some of these limitations.

## Current System

The current system has a web dashboard that displays all relevant computer information in a given company. There are multiple views available on the page such as the company view which shows an overview of all computers in the company and statistics regarding processor, memory and disk usage. There is also the individual view for each machine which shows the same information limited in reference to that machine.

All report data for the system resides in the cloud using Amazon Web Services (AWS).

Some of the current limitations of the system are:

* Users can only view the latest data sent from a machine, not historical information or previous reports.
* Users cannot view a timeline of data for a system or collection of systems.
* Old data records are removed once new data is received from a machine.
* Report data is stored in json files and not in a database.
* Data storage does not include the date and time at which each piece of datum was received.

## Purpose of New System

Our implementation will keep track of all the collected data that is sent to the server. This information will be stored on a MongoDB, a NOSQL database. The system will query this historical data to analyze and display specific trends or behavior.

The new system will address the current system’s limitations such as:

* storing multiple audit records for a system, not just the latest
* storing records in MongoDB
* creating a RESTful API to query historical data from the database
* graphically displaying system audit data to show historical information

**User Stories**

User stories are part of agile software development methodologies as the basis for defining the functions a system must provide and to facilitate requirements management. User stories captures the 'who', 'what' and 'why' of a requirement in a simple, concise way.

The following sections provide a summary of the user stories implemented for this project as well as the pending user stories to be considered for future development.

## Implemented User Stories

**User Story #266 - User seeing Dev Environment displaying Hello World.**

User would like to see the Dev environment displaying the index.html via your Django instance running

**User Story #267 - User seeing Dev Environment displaying Hello World.**

User would like to see the Dev environment displaying the index.html via your Django instance running

# **User Story #280 - Collect User Hardware Information.**

As an IT Admin, I want user hardware information collected for reporting

**Acceptance Criteria:**

1. The user computer hardware information is parsed and collected into a json file
2. The data is sent to the cloud for storage
3. The collection is scheduled to collect information on a regularly timed interval

# **User Story #281 - Scheduled Audits of User Systems**

# As an IT Admin, I want audits to occur on regular intervals so that I can have consistent and almost real-time data available for reporting

# **Acceptance Criteria:**

# User computer automatically initiates system audits on a time interval

# User computer data is sent to the cloud for storage

# **User Story #282 - Collect user login logout history**

As an IT Admin, I want user login and logout information collected for reporting

### Acceptance Criteria:

1. The username and date and time of user login/logout information is parsed and collected into a json file
2. The json data is received in the server

# **User Story #284 - Setup MongoDB**

As software developer we want to setup and use MongoDB to stored audited data

### Acceptance Criteria:

1. MongoDB is installed.
2. Server connects to MongoDB.

# **User Story #285 - Storing login/logout audit data in MongoDB**

As an IT Admin, I want computer audit information and stored in the database.

### Acceptance Criteria:

1. The login/logout history data is stored in the database.

# **User Story #288 - Cache mechanism for login/logout history**

As an IT admin (user) I want to have Addigy collect the loging/logout log data of their machine by checking the logs on a regular basis, and deciding if they need to be forwarded up to Addigy, or if there’s little to no change.

### Acceptance Criteria:

1. The collector checks logs on regular basis.
2. The collector checks caches the changes in the log.
3. The collector forwards to Addigy only information that has changed.

# **User Story #309 -Add AngularJS framework to the project**

As an IT admin I would like have AngularJS as a javascript framework for the client-side application.

### Acceptance Criteria:

1. AngularJS framework is integrated.

# **User Story #313 -Show stored audits in front-end**

As an IT admin I would like to access the data stored in the database and show it in the front-end client.

### Acceptance Criteria:

1. The store data on the database is shown in the client.

**User Story #317 - User Manual**

As an IT Admin, I want a user manual so that I know how to use the Addigy system.

### Acceptance Criteria:

1. Includes installation instructions on a client system
2. Includes installation instructions on the server system
3. Includes instructions on how the user can navigate the front-end GUIs

# **User Story #319 -Refactor login history store document format**

As an IT I would like to parse the login history in a format to facilitate later processing of database/functionality of front-end

### Acceptance Criteria:

1. The format of the login history conforms to the new requirements.

# **User Story #326 - Display Facter Data on Front End**

# As an IT Admin, I want to display information collected from client systems so that I can see the current statistics of a client system.

### Acceptance Criteria:

1. Client data is displayed in a web browser
2. The latest client data is displayed from the database

# **User Story #330 -Login History front-end UI**

As an IT Admin I would like to have most graphs in a single screen so I can see the tools the system has and display their details once I select one of them.

### Acceptance Criteria:

1. The main page displays the login graph.
2. When selected the login graph, a modal with the graph and further details and functionality appears.

# **User Story #268- Display a graph of users logging in and logging off**

As an IT admin I would like to see a graph of user login/logout history.

* Display a graph of users logging on to their computers and logging off, in a one day period
* This will be a single graph that aggregates the total number of users Y logged in, at a time of day X

### Acceptance Criteria:

1. A graph of users login/logout history over time is presented.

# **User Story #335- Discovery-Collecting cloud base application information usage**

# As an IT admin, I would like to know if there is a way to keep track of cloud based applications usage by the users so we can do statistical analysis, productivity analysis among others

### Acceptance Criteria:

# Suitable tools are found that allow us to process information regarding cloud based applications and its usage.

# Found tools are integrated/install if needed.

# Enough time to familiarize with the usage of such tools is dedicated.

# **User Story #340- Collect chrome browsing history data**

As an IT admin I would like to collect information regarding users browsing history so analysis on cloud based applications can be done.

### Acceptance Criteria:

1. Information regarding user browsing history is collected and parsed into a json format.
2. The json data is sent to the server
3. The json data is received in the server

# **User Story #346- Store received browsing history data in MongoDB**

As an IT Admin, I want the browsing history audit information stored in the database.

### Acceptance Criteria:

1. The browsing history data is stored in the database.

(06/15/2015 - 06/26/2015)

# **User Story #351- Display top visited domains**

# As an IT admin I would like to know what are the top visited domains within the organization so I can use that information for statistical analysis.

### Acceptance Criteria:

1. The top visited domains are displayed in a graphical way

# **User Story #354- Create filter mechanism for displaying domain usage info**

# As an IT Admin I would like to be able to chose and filter different categories (users,domains,dates) to analyze information regarding the domains usage.

### Acceptance Criteria:

1. A set of filtering mechanism exists that allows to chose among different categories.
2. A graph reflecting the categories filtered is displayed.
3. The graph is automatically updated with the filters changes.

# **User Story #363- Extend domain history collector to include Mozilla Firefox browser history**

# As an IT admin I would like to collect information regarding users browsing history in Mozilla Firefox browser so analysis on cloud based applications can be done.

### Acceptance Criteria:

1. Information regarding Mozilla Firefox user browsing history is collected and parsed into a json format.
2. The json data is sent to the server
3. The json data is received in the server

# 

# **User Story #366- Extend collector to get browsing history of each user of the system**

# As an IT admin I would like to collect the information regarding browsing history for each user of the system so further analysis can be done with the collected data.

### Acceptance Criteria:

1. Browsing history for each user of the system is collected

# **User Story #374- Setup fiu server environment for demos**

# As a Addigy Team Member I would like to set up Addigy Team assigned server so the team can demo the features done for the project.

### Acceptance Criteria:

1. MongoDB is installed and configured.
2. Django is installed and configured.
3. The project is running on the server

# **User Story #380- Display timeline of events when non-volatile characteristic change**

# As an IT Admin, I want to see a timeline of events of non-volatile system changes so that I can better track when changes occur on a system in my organizations

# Front-end design of graphing and displaying per Org / Tag (customers) / machineID

### Acceptance Criteria:

# Timeline displays events from a set time until present

# The events are categorized by color and/or timeline icon as a hardware or software change

# Timeline does not include volatile measurements (i.e. free memory, uptime, etc.)

# **User Story #381- Allow user to select which computer to display from organization**

# As an IT Admin, I want to choose which computer’s information is being displayed so that I can see the data for each computer in my organization

# 

### Acceptance Criteria:

# A dropdown of computer in the user’s organization appears

# The graph plots the data for the selected computer

# The plotted data automatically updates when the user selects a new machine

# **User Story #383- Extend domain history collector to include Safari browser history**

# As an IT admin I would like to collect information regarding users browsing history in Safari browser so analysis on cloud based applications can be done.

### Acceptance Criteria:

1. Information regarding Safari browsing history is collected and parsed into a json format.
2. The json data is sent to the server
3. The json data is received in the server

# 

# **User Story #386- Extend collector to incorporate system applications**

# As an IT admin I would like to see system applications usage so it can be used for statistical analysis.

### Acceptance Criteria:

1. System Applications are collected stored and showed in the dashboard.

# **User Story #400- Query Mongo for Volatile Data over time**

# As an IT Admin, I want to have a table of volatile data so that I can better understand my business

### Acceptance Criteria:

1. Data displayed does not include volatile data
2. Data displayed contains records for a given time period
3. Design a table to index the volatile data, in the fastest query possible. Index might be “connectorID & timestamp”
4. Query the table for all volatile data for Org / Tag (customer) / MachineID
5. Display in raw format

# **User Story #401- Extend collector to incorporate system applications**

# Discuss and build the proper MVC structure for the page of graphs

# Schedule time with Javier on Monday

### Acceptance Criteria:

1. Any necessary updates/refactoring is made to the application.

# 

# ***User Story #403- Add Software Updates Collector***

# As an IT I would like to see information regarding available software updates at the machines so updates actions can be taken.

### Acceptance Criteria:

# Information regarding available software updates is collected, parsed and stored.

# 

# 

# ***User Story #409- Display graph/animation of available updates count and machine count***

# As an IT admin, I would like to see in the main dashboard a graph/animation with information about how many available updates are in my organization and how many machines needs to be updated so decisions can be made on whether updating action should be taken.

### Acceptance Criteria:

# Information regarding available software updates count and machines needing update count is shown on the main dashboard in a graphical way.

# 

# ***User Story #427- Display tree structure of information of available updates in the system***

# Display tree structure of information of available updates in the system

### Acceptance Criteria:

# A tree structure displaying information about available updates in the system is shown

# 

# ***User Story #443- Collect real information from different device***

# As an IT admin, I would like to collect real information from a device in a real company environment so I can test the application behavior in a different environment

### Acceptance Criteria:

# Collect data from a company device

# 

# ***User Story #431- Refactor Software Updates database structure***

# As an IT admin I would like collect software updates information in a format to facilitate later processing of database/functionality of front-end

### Acceptance Criteria:

# Software updates database structure is refactored.

# ***User Story #440- Add updates/application usage graph***

# As an IT admin I would like to see a graph of applications related to the available updates showing usage of this applications so decisions regarding the updates can be taken

### Acceptance Criteria:

# A graph displaying the usage of applications related to the available updates is shown.

# 

# 

# **User Story #453- Update User manual with new functionality**

# Add new functionality to the user manual

### Acceptance Criteria:

1. Section created for Selecting Tenants
2. Section created for Displaying Non-Volatile Event Timeline

# **User Story #456- Login History Testing**

# As an IT Admin I would like to make sure that the login history feature work as expected so accurate information can be extracted from it.

### Acceptance Criteria:

1. The login History feature is correctly working.

# **User Story #462- Fix login History times bug**

# As an IT admin I would like that the login History shows the information in the most accurate way.

### Acceptance Criteria:

1. The login graph displays the user’s activity login/out times in an accurate way having in consideration different login/out day.

# 

# **User Story #463- Add software metering last day filter**

# As an IT admin, I would like to be able to see the last day software activity in the software metering tool so an analysis in shorter period of time can be made.

### Acceptance Criteria:

1. The software metering module allows for last day filtering.

# **User Story #470- Update user manual with Updates module instructions**

# As an IT Admin, I would like to have a instructive section in the user manual related to the Software updates so I can refer to it to use the tool.

### Acceptance Criteria:

1. Add Software Updates section to the user manual.

# 

# **User Story #470- Modify collector to take a list of parameters**

# As an IT Admin I would like to be able to run the collector with the option of passing the organization, connection and server address as parameters.

### Acceptance Criteria:

1. The collector accepts organization, connector and server address as parameters

# ***User Story #377- Fix Java Client Collector***

DEFECT: After last merge with master, java collector does not run properly.

### Acceptance Criteria:

1. Java collector compiles
2. Java collector sends collected data into mongoDB

# 

# **User Story #490- Update project final deliverable**

# As a Software Engineer, I want to keep up to date the project document with the latest progress regarding the project

### Acceptance Criteria:

1. Final Project Document is updated.

# **User Story #491- Presentation Preparation**

# As a Software Engineer I want to prepare the project presentation so I can explain the relevance and characteristics of the project in a efficient way.

### Acceptance Criteria:

1. Presentation main points are outlined and practiced

# 

# **User Story #499- Software Metering tool testing**

# As an IT admin I want to make sure the Software metering tool works correctly so the decisions made based on it are accurate.

### Acceptance Criteria:

1. Software metering tool is tested.

# 

# **User Story #502- Software Updates Tool Testing**

# As an IT admin I want to make sure the Software metering tool works correctly so the decisions made based on it are accurate.

### Acceptance Criteria:

1. Software Updates Tool is tested

## Pending User Stories

All the users stories were completed.

# 

**Project Plan**

This project is to be organized and developed using a variety of different tools gears towards an agile development approach. By using an agile development approach, it ensures that the project is constantly being updated and available to deploy at any time during its life cycle.

## Hardware and Software Resources

The resources we will be using for this project are the following:

**Java**

Java will be used to implement the client data collector. The following considerations led to select Java among other programming languages:

1. Broad collection of libraries.
2. Cross-platform

**MongoDB**

A NOSQL database simplifies storage and incorporates a date time stamp for processing historical records and trends. The following considerations led to select MongoDB among the NOSQL databases:

1. A document-based data model which is a rich data structure capable of holding arrays and other documents equivalent to JSON, Python dictionaries, etc.
2. Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL.
3. No schema migrations. Since MongoDB is schema-free, the code defines the schema.
4. A clear path to horizontal scalability.

**Python**

The Python framework, specifically the Django framework, is used on the backend as the project's web server. One of the primary reasons for choosing python was to be consistent with other existing server-side technologies used at Addigy. Other important reasons for using Python include:

* Quicker prototyping/development because due to high-level data types and dynamic typing
* Existing web frameworks, like Django, that require minimal work to start serving requests from client applications
* Django is easily scalable

**AngularJS**

AngularJS allows for easy development of Views in an MVC pattern without having to build a complete MVC pipeline on the back-end. This project uses the backend to receive, process, store, and serve json data from client systems. AngularJS allows the project to make RESTful calls to the back end, receiving data for the front-end to display to the user on the dashboard view.

**Linux VMs**

We use a linux virtual machine to host all of the project files, and to host the demo

page. The server runs an Apache server.

**Bootstrap**

For front-end design and templating, this project takes advantage of Twitter Bootstrap. Bootstrap is a quick and easy way to implement consistent style on the project’s dashboard view. There are also many existing bootstrap templates, including templates for displaying graphical data, that this project can leverage for quicker implementation of visualization.

**Mac/Linux for development**

The team for this project decided to use a Mac/Linux development environment primarily because this is the existing development environment for the current team. Another benefit over other options include an integrated packet manager for quickly installing new tools (apt-get or homebrew). Lastly, the production environment for this project resides on a Ubuntu Linux server. By mirroring this environment during development, deployment should be much easier to accomplish.

## Sprints Plan

### Sprint 1

(05/18/2015 - 05/29/2015)

# **User Story #280 - Collect User Hardware Information**

As an IT Admin, I want user hardware information collected for reporting

***Tasks***

* Send Json data to server
* Parse data into json format
* Collect data using “facter” command

### Acceptance Criteria:

1. The user computer hardware information is parsed and collected into a json file
2. The data is sent to the cloud for storage
3. The collection is scheduled to collect information on a regularly timed interval

# **User Story #281 - Scheduled Audits of User Systems**

As an IT Admin, I want audits to occur on regular intervals so that I can have consistent and almost real-time data available for reporting

***Tasks***

* Create cronjob script
* Create Maven project to build java project

### Acceptance Criteria:

1. User computer automatically initiates system audits on a time interval
2. User computer data is sent to the cloud for storage

# **User Story #282 - Collect user login logout history**

As an IT Admin, I want user login and logout information collected for reporting.

***Tasks***

* Collect data using “last” command to get users login/logout history.
* Solve last compatibility issues across OS(Mac/Unix).
* Parse collected data to Json file.
* Send Json data to server.

### Acceptance Criteria:

1. The username and date and time of user login/logout information is parsed and collected into a json file
2. The json data is received in the server

# **User Story #284 - Setup MongoDB**

As software developer we want to setup and use MongoDB to stored audited data

***Tasks***

* Install MongoDB.
* Create infrastructure in the server to connect to MongoDB
* Configuring authentication schema for MongoDB connection

### Acceptance Criteria:

1. MongoDB is installed.
2. Server connects to MongoDB.

# **User Story #285 - Storing login/logout audit data in MongoDB**

As an IT Admin, I want computer audit information and stored in the database

***Tasks***

* Get data sent from the collector.
* Define database collection indexes and storage details
* Received audited data from the collector agent in the server
* Store received data in MongoDB

### Acceptance Criteria:

1. The login/logout history data is stored in the database.

# **User Story #285 - Cache mechanism for login/logout history**

As an IT admin (user) I want to have Addigy collect the logging/logout log data of their machine by checking the logs on a regular basis, and deciding if they need to be forwarded up to Addigy, or if there’s little to no change.

***Tasks***

* Add only login/out changes to the cached data
* Schedule data forwarding to server

### Acceptance Criteria:

1. The collector checks logs on regular basis.
2. The collector checks caches the changes in the log.
3. The collector forwards to Addigy only information that has changed.

# **User Story #309 - Add AngularJS framework to the project**

As an IT admin I would like have AngularJS as a javascript framework for the client-side application.

***Tasks***

* Add basic angular modules to the project
* Set up basic angular structure
* Add basic angular modules to the project

### Acceptance Criteria:

1. AngularJS framework is integrated.

### Sprint 2

(05/01/2015 - 06/15/2015)

# **User Story #313 - Show stored audits in front-end**

As an IT admin I would like to access the data stored in the database and show it in the front-end client.

***Tasks***

* Create client-side mechanism to display returned data
* Create client-side mechanism to request store data
* Create server-side infrastructure to get data from database

### Acceptance Criteria:

1. The store data on the database is shown in the client.

# **User Story #319 - Refactor login history store document format**

As an IT I would like to parse the login history in a format to facilitate later processing of database/functionality of front-end

***Tasks***

* Refactor server code
* Refactor login collector

### Acceptance Criteria:

1. The format of the login history conforms to the new requirements.

# **User Story #330 - Login History front-end UI**

As an IT Admin I would like to have most graphs in a single screen so I can see the tools the system has and display their details once I select one of them.

***Tasks***

* 334:Add collapsible menu to see user per hour details
* 333:Login History UI adjustments
* 332:Create mechanism to load in modal correspondent htmls
* 331:Create modal to show when a graph is selected

### Acceptance Criteria:

1. The main page displays the login graph.
2. When selected the login graph, a modal with the graph and further details and functionality appears.

**Modeling**

* Appendix A : Figure #2.330

# **User Story #268- Display a graph of users logging in and logging off**

As an IT admin I would like to see a graph of user login/logout history.

* Display a graph of users logging on to their computers and logging off, in a one day period
* This will be a single graph that aggregates the total number of users Y logged in, at a time of day X

***Tasks***

* 322:Discovery graphing tools
* 323:Integrating graphing tools
* 324:Displaying login activity for a fixed date
* 325:Allow for date selection to display such date activity

### Acceptance Criteria:

1. A graph of users login/logout history over time is presented.

**Modeling**

* Appendix A : Figure #2.268
* Appendix A : Figure #2.268a

# **User Story #335- Discovery-Collecting cloud base application information usage**

# As an IT admin, I would like to know if there is a way to keep track of cloud based applications usage by the users so we can do statistical analysis, productivity analysis among others

# ***Tasks***

# 336:Research tools of collecting information regarding cloud based applications usage

* 337:Install/integrate found tools.
* 338:Explore/Familiarize with the functionality and usage of such tools

# 

### Acceptance Criteria:

# Suitable tools are found that allow us to process information regarding cloud based applications and its usage.

# Found tools are integrated/install if needed.

# Enough time to familiarize with the usage of such tools is dedicated.

# **User Story #340- Collect chrome browsing history data**

As an IT admin I would like to collect information regarding users browsing history so analysis on cloud based applications can be done.

***Tasks***

* 341:Collect chrome browsing history using sqlite db
* 342:Parse collected data into json format
* 343:Send collected data to server

### Acceptance Criteria:

1. Information regarding user browsing history is collected and parsed into a json format.
2. The json data is sent to the server
3. The json data is received in the server

# **User Story #346- Store received browsing history data in MongoDB**

As an IT Admin, I want the browsing history audit information stored in the database.

# ***Tasks***

# 347:Define database collection indexes and storage details

# 348:Store received browsing history data in MongoDB

### Acceptance Criteria:

# The browsing history data is stored in the database.

### Sprint 3

(06/15/2015 - 06/26/2015)

# **User Story #351- Display top visited domains**

# As an IT admin I would like to know what are the top visited domains within the organization so I can use that information for statistical analysis.

***Tasks***

* 353:Display in the main page the top visited domains
* 352:Get Top visited domains from the database

### Acceptance Criteria:

1. The top visited domains are displayed in a graphical way

**Modeling**

* Appendix A : Figure #3.351

# **User Story #354- Create filter mechanism for displaying domain usage info**

# As an IT Admin I would like to be able to chose and filter different categories (users,domains,dates) to analyze information regarding the domains usage.

***Tasks***

* 362:Refine UI for the domains filters/graph section
* 361:Create mechanism to update graph according to selected filters
* 360:Get elements from the database that satisfied specified filters
* 359:Create mechanism to combine filters and produce corresponding query
* 358:Create dates filter
* 357:Create top filter
* 356:Create user filter
* 355:Create domain filter

### Acceptance Criteria:

1. A set of filtering mechanism exists that allows to chose among different categories.
2. A graph reflecting the categories filtered is displayed.
3. The graph is automatically updated with the filters changes.

**Modeling**

* Appendix A : Figure #3.354

# 

# **User Story #363- Extend domain history collector to include Mozilla Firefox browser history**

# As an IT admin I would like to collect information regarding users browsing history in Mozilla Firefox browser so analysis on cloud based applications can be done.

***Tasks***

* 365:Incorporate collected data to the Chrome collected data
* 364:Collect Mozilla Firefox browsing history using sqlite db

### Acceptance Criteria:

1. Information regarding Mozilla Firefox user browsing history is collected and parsed into a json format.
2. The json data is sent to the server
3. The json data is received in the server

# 

# **User Story #366- Extend collector to get browsing history of each user of the system**

# As an IT admin I would like to collect the information regarding browsing history for each user of the system so further analysis can be done with the collected data.

***Tasks***

* 368:Create a factory class in the collector to deal with OS compatibility
* 367:Get list of all systems users and their home paths

### Acceptance Criteria:

1. Browsing history for each user of the system is collected

# **User Story #374- Setup fiu server environment for demos**

# As a Addigy Team Member I would like to set up Addigy Team assigned server so the team can demo the features done for the project.

***Tasks***

* 376:Install MongoDB and configure it
* 375:Install Django and configure it

### Acceptance Criteria:

1. MongoDB is installed and configured.
2. Django is installed and configured.
3. The project is running on the server

# **User Story #383- Extend domain history collector to include Safari browser history**

# As an IT admin I would like to collect information regarding users browsing history in Safari browser so analysis on cloud based applications can be done.

***Tasks***

* 385:Collect Safari browsing history using sqlite db
* 384:Incorporate collected data to browsing history collector

### Acceptance Criteria:

1. Information regarding Safari browsing history is collected and parsed into a json format.
2. The json data is sent to the server
3. The json data is received in the server

# 

# **User Story #386- Extend collector to incorporate system applications**

# As an IT admin I would like to see system applications usage so it can be used for statistical analysis.

***Tasks***

* 387:Research way to get each user running applications
* 388:Collect user running applications.
* 389:Combine user running applications with browsing(cloud) collected applications
* 390:Create mechanism to define application lifetime
* 391:Add application type filter to dashboard

### Acceptance Criteria:

1. System Applications are collected stored and showed in the dashboard.

### Sprint 4

(06/17/2015 - 07/10/2015)

# **User Story #403- Add Software Updates Collector**

# As an IT I would like to see information regarding available software updates at the machines so updates actions can be taken.

***Tasks***

* 404: Research appropriate way to get software updates
* 405: Extend collector to collect and parse software updates
* 406: Send collected updates to server
* 407:Store collected updates in MongoDB
* 408: Create database collection to store available software updates

### Acceptance Criteria:

1. Information regarding available software updates is collected, parsed and stored.

# 

# **User Story #409- Display graph/animation of available updates count and machine count**

# As an IT admin, I would like to see in the main dashboard a graph/animation with information about how many available updates are in my organization and how many machines needs to be updated so decisions can be made on whether updating action should be taken.

***Tasks***

* 410: Get available updates count and machines needing updates count
* 411: Display using a gauge available software updates count
* 412: Display using a gauge machine needing updates count

### Acceptance Criteria:

1. Information regarding available software updates count and machines needing update count is shown on the main dashboard in a graphical way.

**Modeling**

* Appendix A : Figure #4.409

# **User Story #427- Display tree structure of information of available updates in the system**

# Display tree structure of information of available updates in the system

***Tasks***

* 428: Get available updates in the system and machine count
* 429: Create policies hierarchy table in MongoDB
* 430: Display available updates per policy in a tree structure
* 437: Refine software updates tree structure UI

### Acceptance Criteria:

1. A tree structure displaying information about available updates in the system is shown

**Modeling**

* Appendix A : Figure #4.427

# **User Story #443- Collect real information from different device**

# As an IT admin, I would like to collect real information from a device in a real company environment so I can test the application behavior in a different environment

***Tasks***

* 444: Modify collector details to allow it to run in a different device
* 445: Fix bugs that appear as a result of testing in a different environment

### Acceptance Criteria:

1. Collect data from a company device

# **User Story #431- Refactor Software Updates database structure**

# As an IT admin I would like collect software updates information in a format to facilitate later processing of database/functionality of front-end

***Tasks***

* 432: Create updates collection structure
* 433: Refactor machineUpdates collection structure
* 434: Create policies collection structure
* 435: create machinePolicies collection structure
* 436: Fix front-end and server functionality to adapt to the new software updates db structure

### Acceptance Criteria:

1. Software updates database structure is refactored.

# **User Story #440- Add updates/application usage graph**

# As an IT admin I would like to see a graph of applications related to the available updates showing usage of this applications so decisions regarding the updates can be taken

***Tasks***

* 441: Get the applications usage of applications related to available updates
* 442: Display a graph of the applications usage of applications related to available updates

### Acceptance Criteria:

1. A graph displaying the usage of applications related to the available updates is shown.

**Modeling**

* Appendix A : Figure #4.427

### 

### Sprint 5

(07/13/2015 - 07/24/2015)

# **User Story #456- Login History Testing**

# As an IT Admin I would like to make sure that the login history feature work as expected so accurate information can be extracted from it.

***Tasks***

# 468: Test the login History date selector

# 469: Test the login History user details feature

### Acceptance Criteria:

1. The login History feature is correctly working.

# **User Story #458- Sequence Diagrams**

# As an IT Admin, I need to have sequence diagrams describing the system so that future developers can add better understand and add functionality in the future.

***Tasks***

# 460: Redo existing SD for design uniformity

# 461: Create new SD for existing functionality

### Acceptance Criteria:

1. Complete the following diagrams:

* Facter Report
* Available Memory
* Volatile Data
* Non-Volatile Event Timeline
* Select Tenant

# **User Story #462- Fix login History times bug**

# As an IT admin I would like that the login History shows the information in the most accurate way.

***Tasks***

# 467: Fix login History times bug

### Acceptance Criteria:

1. The login graph displays the user’s activity login/out times in an accurate way having in consideration different login/out day.

# 

# **User Story #463- Add software metering last day filter**

# As an IT admin, I would like to be able to see the last day software activity in the software metering tool so an analysis in shorter period of time can be made.

***Tasks***

# 466: Extend controller to allow for hours labels in the graph

# 465: Extend controller to allow for last day activity request

* 464: Add UI tab to select "last day" activity

### Acceptance Criteria:

1. The software metering module allows for last day filtering.

# **User Story #470- Update user manual with Updates module instructions**

# As an IT Admin, I would like to have a instructive section in the user manual related to the Software updates so I can refer to it to use the tool.

***Tasks***

# 471: Update Addigy user manual with the Software Updates section.

### Acceptance Criteria:

1. Add Software Updates section to the user manual.

# **User Story #470- Modify collector to take a list of parameters**

# As an IT Admin I would like to be able to run the collector with the option of passing the organization, connection and server address as parameters.

# 

***Tasks***

# 473: [Modify the collecto](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/473)r [to take in a list of parameters](https://fiu-scis-seniorproject.mingle.thoughtworks.com/projects/addigy/cards/473)

### Acceptance Criteria:

1. The collector accepts organization, connector and server address as parameters

# 

# **User Story #474- Facter Data Injector**

# As an IT Admin, I want to see the sockets established on systems in my organization so I can see which software is actively using these systems via the network.

***Tasks***

# 482: New facter fact to collect data from client machine

* 483: Controller to parse socket data
* 484: Graph to display data
* 485: Table to display data
* 487: Bug fix - remove redundant data from table
* 488: Bug fix - show tenants using sockets

### Acceptance Criteria:

1. Display a graph of socket usage for a 24 hour period
2. Display a table with which machines have which sockets open
3. Display a table with which sockets a software application has open

# **User Story #475- Facter Data Injector**

# A tool to inject facter reports into the system for testing and demonstration purposes.

***Tasks***

# 480: Display latest facter report to user

* 481: Inject report into DB

### Acceptance Criteria:

1. Injects a facter report into the DB
2. Allows user to input orgId, collectorId, and facter report json

# **User Story #476- Practice presentation**

# Practice giving the final presentations for showcase event.

***Tasks***

# 477: Sunday, 7/19

* 478: Tuesday, 7/21

### Acceptance Criteria:

None.

# **User Story #479- Run collector on different devices to collect data for demo**

# As an IT admin I would like to run the collector in several devices to be able to display the collected data in the dashboard.

***Tasks***

# 473: Run collector on different devices to collect data for the demo

### Acceptance Criteria:

1. The collector is run in a few devices and the data is stored in the fiu server

# **User Story #490- Update project final deliverable**

# As a Software Engineer, I want to keep up to date the project document with the latest progress regarding the project

***Tasks***

# 498: Update collector class diagram

* 497: Update architectural and design patterns section

### Acceptance Criteria:

1. Final Project Document is updated.

# 

# **User Story #491- Presentation Preparation**

# As a Software Engineer I want to prepare the project presentation so I can explain the relevance and characteristics of the project in a efficient way.

***Tasks***

# 495: Sunday 7/19

* 494: Tuesday 7/21

### Acceptance Criteria:

1. Presentation main points are outlined and practiced

# 

# **User Story #499- Software Metering tool testing**

# As an IT admin I want to make sure the Software metering tool works correctly so the decisions made based on it are accurate.

***Tasks***

# 501: Add software metering test cases

* 500: Test Software Metering Tool

### Acceptance Criteria:

1. Software metering tool is tested.

# 

# **User Story #502- Software Updates Tool Testing**

# As an IT admin I want to make sure the Software metering tool works correctly so the decisions made based on it are accurate.

***Tasks***

# 504: Add software updates tool test cases

* 503: Test Software Updates Tool

### Acceptance Criteria:

1. Software Updates Tool is tested

# **User Story #457- Software Updates Tool Testing**

# As an IT Admin, I want to know that the application I am using is tested so that the application works properly

***Tasks***

# 492: Test cases for new functionality

* 459: Run system test cases

### Acceptance Criteria:

1. Facter Report
2. Available Memory
3. Volatile Data
4. Non-Volatile Event Timeline
5. Select Tenant

# **System Design**

This section explains how the whole system was broken down into its different components. It will explain their purpose and how they work together.

## Architectural Patterns

* The dashboard is implemented using AngularJS, a Model-View-Controller architecture in order to keep the dashboard up to date.
* The collector uses a Caching mechanism to avoid expensive transmission of repeated information. A record of the audit date and last audited information is kept in the system and used to determined only the new data that needs to be send/stored.

## System and Subsystem Decomposition

The **Agent** program that collects and sends system data to the cloud is implemented using a java package. The object model diagram for the java client package is shown in figure sd-3. This subsystem includes the Collector interface which all classes that collect data from the system implement. By doing so, the main class is able to contain an array of all the collectors and iterate through the list. The output of each collector is used to compose a single json string to send to the server for parsing and storage.

The **Server** that receives the data from the **Agent**, connects to the database to store it and retrieves the data to the dashboard is implemented using the Django framework of Apache, MongoDB, Python and Ubuntu.

The **Dashboard Controller** serves as way for the administrator to retrieve the data stored in the database. It was developed using the AngularJS Framework. The controller is responsible for connecting to the **Server** and retrieving information regarding all the machines for an organization. The controller handles administrator’s interactions such as filtering applications, users, timestamps, among others.

The **Web Dashboard View**was developed using HTML and CSS. It takes advantage of frameworks such as BootStrap and JQuery to display functionalities and data processed by the Logic Tier.

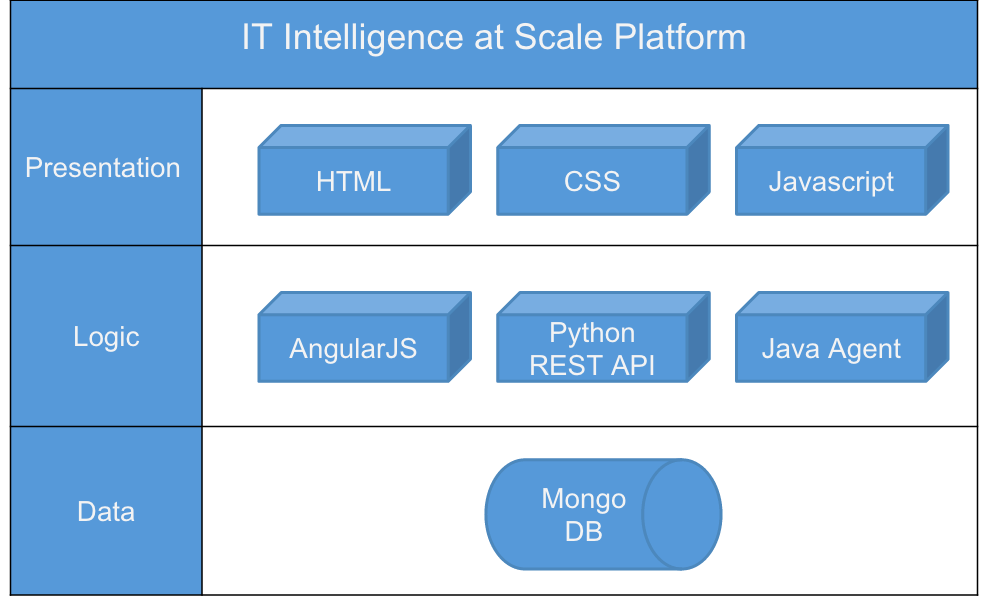
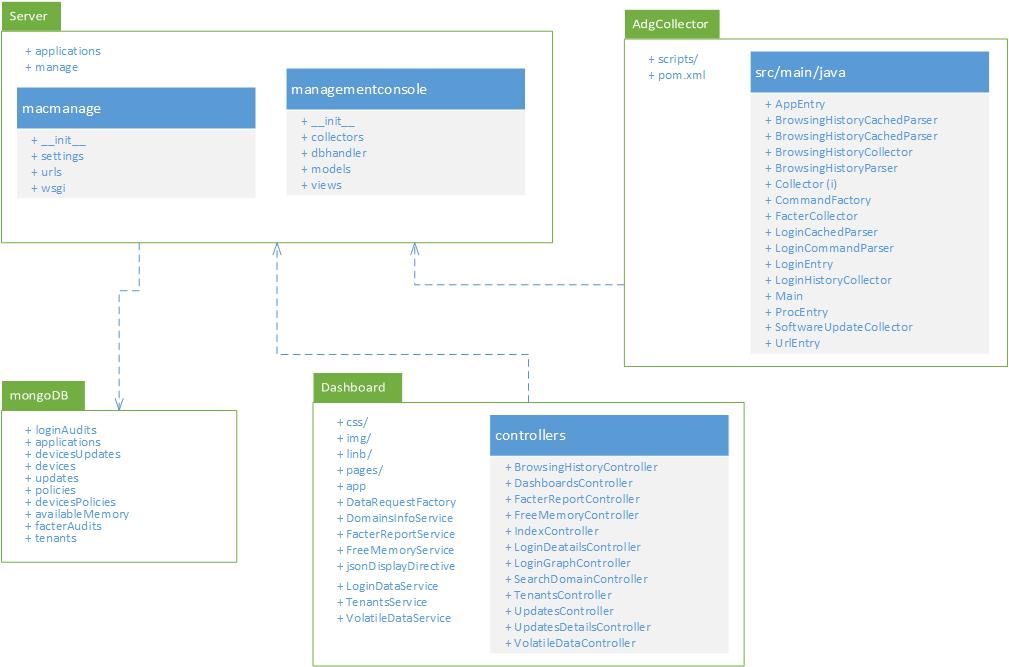
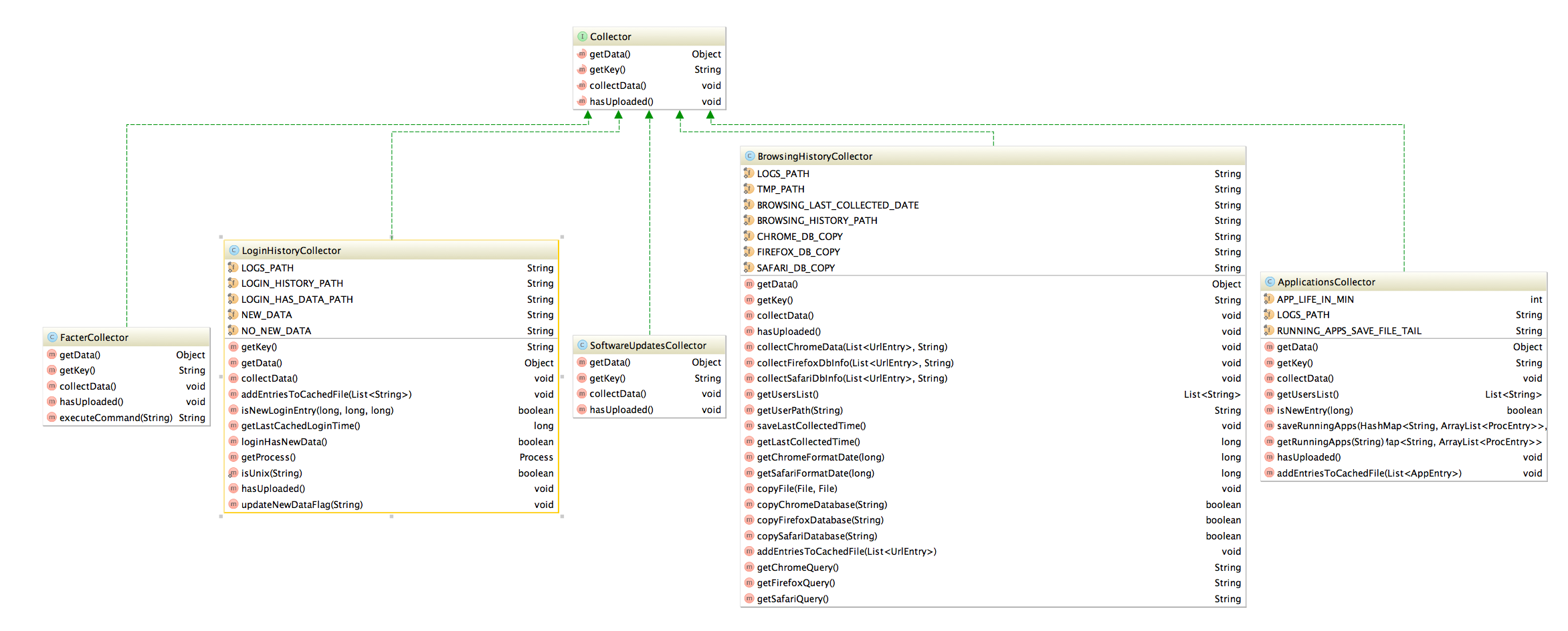


Figure sd-1 System Design

Figure sd-2 System Design Package



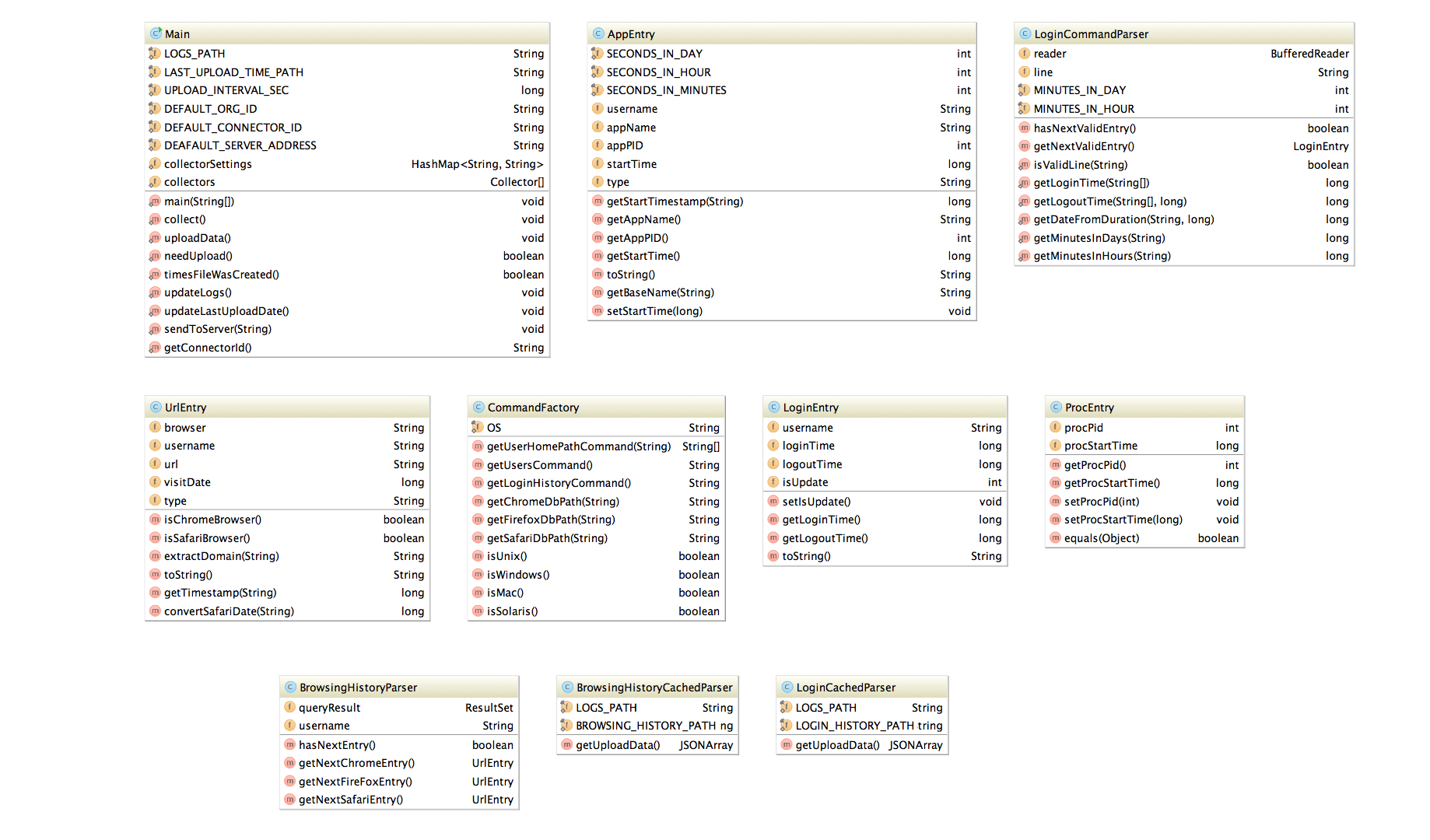


Figure sd-3 Collector Class Diagram

## Deployment Diagram

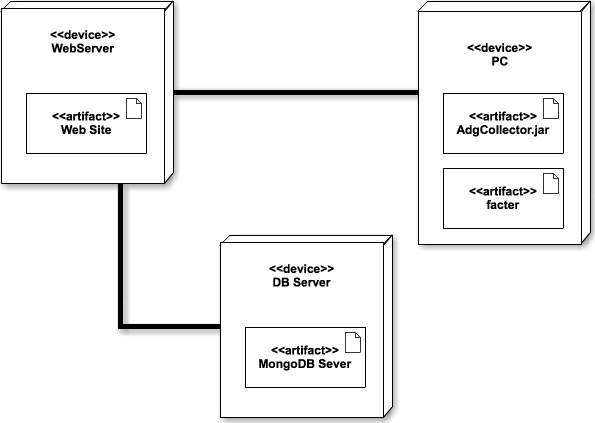


Figure D1. Deployment Diagram

## Design Patterns

**Factory Method (Creational Pattern)**

The collector agent running on the machines uses a Factory Method design in order to achieve platform independence by having a factory object generate different commands depending on the platform in which the machine is running.

**Front Controller (Structural Pattern)**

The Python server provides a centralized entry point for handling all the request requests. Based on the specific request, it then instantiates further objects and call methods to handle the particular task(s) required. This pattern is used for better organization of the project and to avoid code replication.

# **System Validation**

The system validation for the Addigy Web Dashboard was done through manual testing of the different components due to the complexity of the Javascript libraries making it impossible to test within the constraints of a test automation framework within the project timeframe.

User Story #1 - Login History Module

System Tests

|  |  |
| --- | --- |
| Identifier | TC01\_AutoRefreshLoginHistory |
| Purpose | To ensure that login history graph refreshes automatically reflecting the selected date activity. |
| Setup | Navigate to the dashboard2, Login History module. |
| Input | Select a date on the date selector on top of the login history graph. |
| Expected Output | Login graph should automatically refresh displaying the activity of the selected date. |

|  |  |
| --- | --- |
| Identifier | TC02\_SelectLoginHistoryDetailsAtAHour |
| Purpose | To ensure that details of the users logged in at each hour are displayed. |
| Setup | -Navigate to the dashboard2.  -Expand the Login History module |
| Input | -Select a date on the date selector on top of the login history graph.  -Click on Users Details button.  -Select a specific hour to display. |
| Expected Output | -A table with the users logged in at that time should appear.  -Details about the login time and logout time of each user should appear.  -If no user logged a message indicating that no users were logged in at that time should appear. |

User Story #2 - Software Metering Module

System Tests

|  |  |
| --- | --- |
| Identifier | TC03\_SelectTopXDomains |
| Purpose | To ensure that applications graph refreshes automatically reflecting the top X selected applications. |
| Setup | -Navigate to the dashboard2.  -Expand the Software Metering Module. |
| Input | Select a quantity in the Top dropdown menu. |
| Expected Output | Applications graph should automatically refresh reflecting the top X applications activity in different colors. |

|  |  |
| --- | --- |
| Identifier | TC04\_SelectUserDomains |
| Purpose | To ensure that applications graph refreshes automatically reflecting the selected user application’s activity. |
| Setup | -Navigate to the dashboard2.  -Expand the Software Metering Module. |
| Input | -Select a user in the User drop-down menu. |
| Expected Output | Applications graph should automatically refresh reflecting the selected user’s application’s activity. |

|  |  |
| --- | --- |
| Identifier | TC05\_SelectDomain |
| Purpose | To ensure that applications graph refreshes automatically reflecting the selected application’s activity |
| Setup | -Navigate to the dashboard2.  -Expand the Software Metering Module. |
| Input | Select a Applications in the Applications drop-down menu. |
| Expected Output | Applications graph should automatically refresh reflecting the selected application’s activity. |

|  |  |
| --- | --- |
| Identifier | TC06\_SelectDomainActivityTimeframe |
| Purpose | To ensure that applications graph refreshes automatically reflecting the selected timeframe application’s activity. |
| Setup | -Navigate to the dashboard2.  -Expand the Software Metering Module. |
| Input | -Select one of the two tabs in the Time Period Menu. |
| Expected Output | Applications graph should automatically refresh reflecting the selected time period’s application’s activity. |

User Story #3 - Software Updates Module

System Tests

|  |  |
| --- | --- |
| Identifier | TC08\_DisplayAvailableUpdatesTreeStructure |
| Purpose | To ensure that the available softwares display according to the policy tree structure. |
| Setup | Navigate to the dashboard2, Software Updates module. |
| Input | -Select an available software update.  -Select childs of the policies under the selected software update. |
| Expected Output | Subsections containing child policies that have devices with the selected update/ and or tables with the devices belonging to that policy that have that update available should appear in a tree structure. |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| User Story #4 - Facter Report    System Tests     |  |  | | --- | --- | | Identifier | TC09\_DisplayAvailableFacterReport | | Purpose | To ensure that the system can display an available facter report. | | Setup | Navigate to the dashboard1, Facter Report module. | | Input | 1. Click the “Display Facter Report” button. | | Expected Output | 1. The Facter Report module now displays a facter report. |       User Story #5 – Available Memory    System Tests     |  |  | | --- | --- | | Identifier | TC10\_DisplayAvailableMemoryGraphInModule | | Purpose | To ensure that the system can update and display an available memory graph in the Available Memory module. | | Setup | Navigate to the dashboard1, Available Memory module. | | Input | 1. Select a date on the date selector on top of the available memory graph. | | Expected Output | 1. The available memory graph automatically updates the graph to display data starting from the date selected. |      |  |  | | --- | --- | | Identifier | TC11\_ DisplayAvailableMemoryGraphInModal | | Purpose | To ensure that the system can update and display an available memory graph in the Available Memory modal. | | Setup | Navigate to the dashboard1, Available Memory module. | | Input | 1. Select a date on the date selector on top of the available memory graph.  2. Select the expand button on the top-right corner of the Available Memory module. | | Expected Output | 1. The available memory graph displays in the modal window. |      |  |  | | --- | --- | | Identifier | TC12\_ ChangeTenantForAvailableMemoryGraph | | Purpose | To ensure that the system can update and display an available memory graph for a selected tenant. | | Setup | Navigate to the dashboard1, Available Memory module. | | Input | 1. Select a date on the date selector on top of the available memory graph.  2. Select a different tenant from the Tenant dropdown on the top module in the dashboard. | | Expected Output | 1. The available memory graph updates the graph for the selected tenant. |     User Story #6 – Non-Volatile Event Timeline    System Tests     |  |  | | --- | --- | | Identifier | TC13\_DisplayTimelineOfEvents | | Purpose | To ensure that the system can display the Non-Volatile Event Timeline in a modal window. | | Setup | Navigate to the dashboard1, Non-Volatile Event Timeline module. | | Input | 1. Select the expand button on the top-right corner of the Available Memory module. | | Expected Output | 1. The Non-Volatile Event Timeline displays in the modal window. |      |  |  | | --- | --- | | Identifier | TC14\_UpdateTimelineOfEventsForDifferentTimePeriod | | Purpose | To ensure that the system can display the Non-Volatile Event Timeline for the selected time period. | | Setup | Navigate to the dashboard1, Non-Volatile Event Timeline module. | | Input | 1. Select the expand button on the top-right corner of the Available Memory module.  2. Select the “1 week” button to display the range of data being displayed in the timeline. | | Expected Output | 1. The Non-Volatile Event Timeline displays events for the past week. |     User Story #7 – Volatile Data    System Tests     |  |  | | --- | --- | | Identifier | TC15\_DisplayTimelineOfEvents | | Purpose | To ensure that the system can display Volatile Data for the selected time period. | | Setup | Navigate to the dashboard1, Volatile Data module. | | Input | 2. Select the “1 week” button to display the range of data being displayed in the timeline. | | Expected Output | 1. The Volatile Data module displays the raw volatile data for the past week. |     User Story #8 – Socket Data    System Tests     |  |  | | --- | --- | | Identifier | TC16\_DisplayDataGraph | | Purpose | To ensure that the system can display the Socket Data graph for a 24 hour time period. | | Setup | Navigate to the dashboard1, Socket Data module. | | Input | 1. None | | Expected Output | 1. The Socket Data graph displays the socket data usage for the past 24 hours for all users in the organization. |        |  |  | | --- | --- | | Identifier | TC17\_DisplayDataTable | | Purpose | To ensure that the system can display the Socket Data table for a 24 hour time period. | | Setup | Navigate to the dashboard1, Socket Data module. | | Input | 1. Click a time label where data exists in the graph. | | Expected Output | 1. The Socket Data table displays the socket data usage for the past 24 hours for all users in the organization. | |  |
|  |  |
|  |  |
|  |  |

# **Glossary**

* **Audit Data:** Data from client computers compiled using the client side application for Addigy that contains information related to hardware and software configurations on the machine.
* **JSON:** File format that all client audit data will be stored in. Data is encapsulated for easy access using brackets to group related data.
* **MongoDB**: NoSQL DB.
* **Agent:** A program that acts on the user’s computer on behalf of the administrator.

# 

# 

# **Appendix**

## Appendix A - UML Diagrams

### Static UML Diagrams

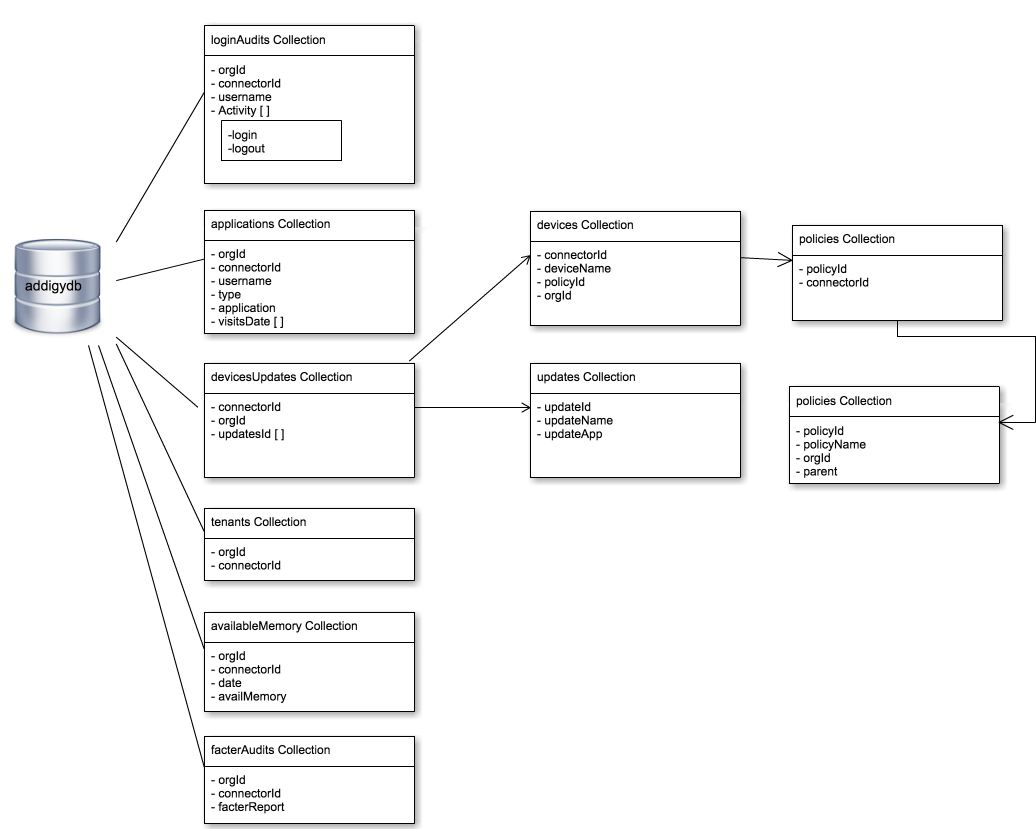


Figure A.1 MongoDB Schema

### Dynamic UML Diagrams

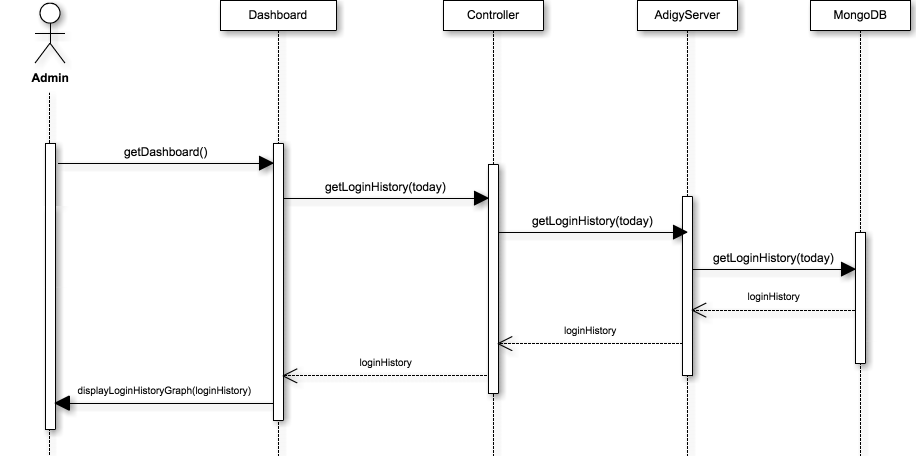


Figure #2.268 Display Today’s Login History

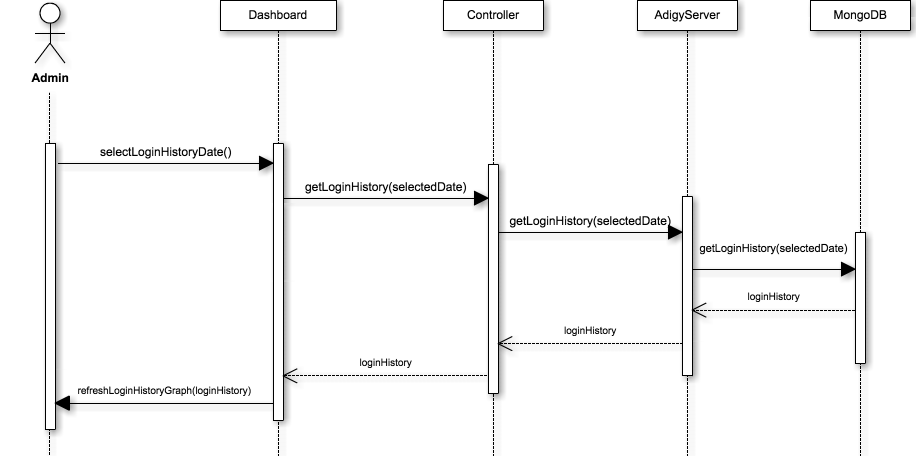


Figure #2.268a Display Selected Date Login History



Figure #2.330 Display Users Per Hour details.

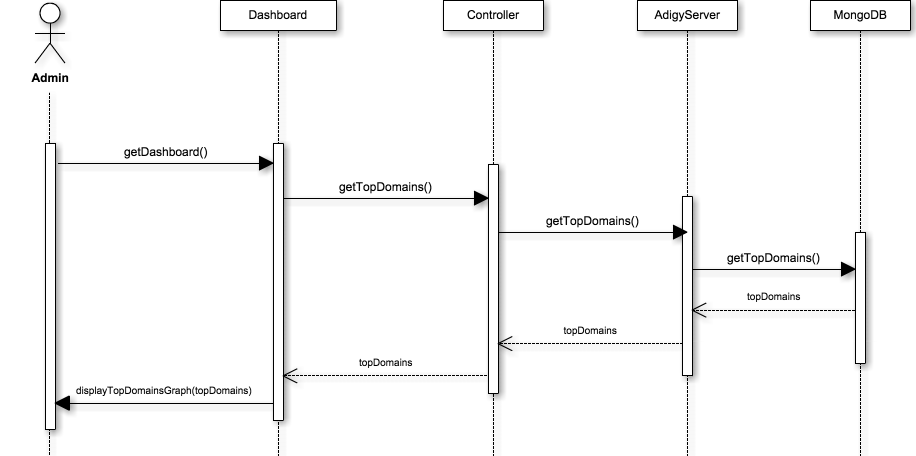
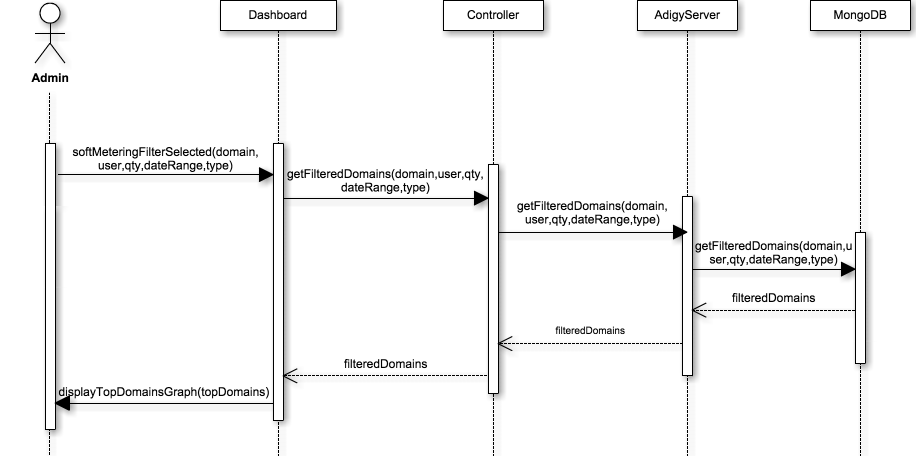


Figure #3.351 Display Top Visited Domains

Figure #3.354 Get Filtered Domains

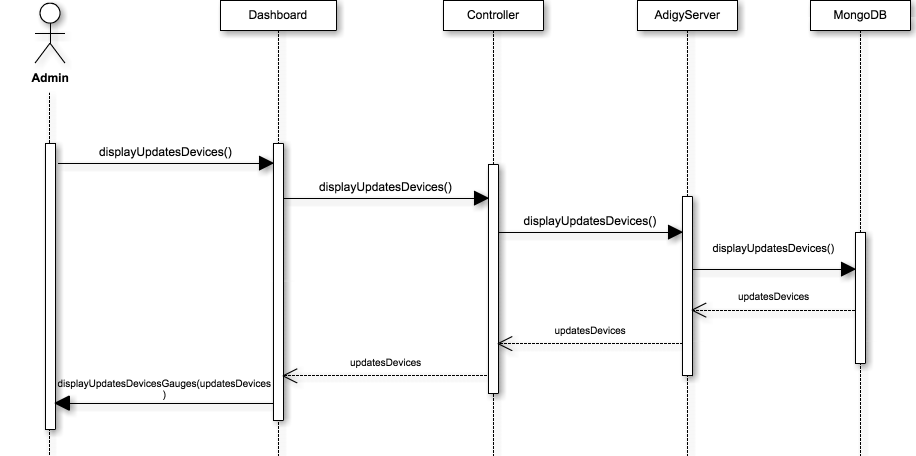


Figure #4.409 Display gauges of available updates count and machine count



Figure #4.427 Display software update details

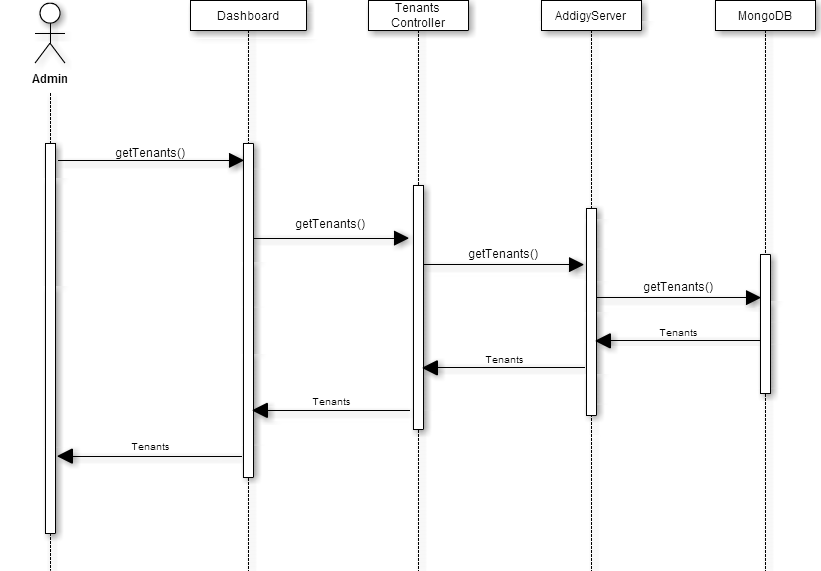


Figure #4.428 Tenants Sequence Diagram

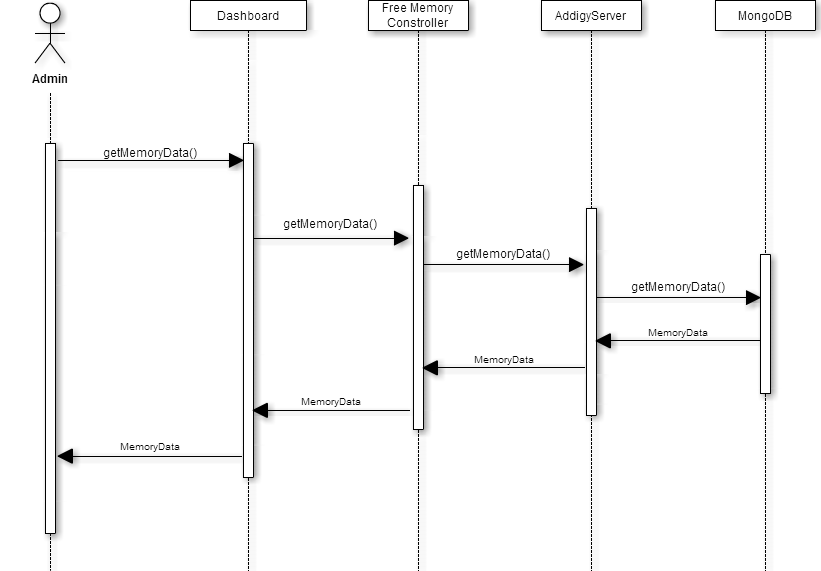


Figure #4.429 Volatile Data Sequence Diagram

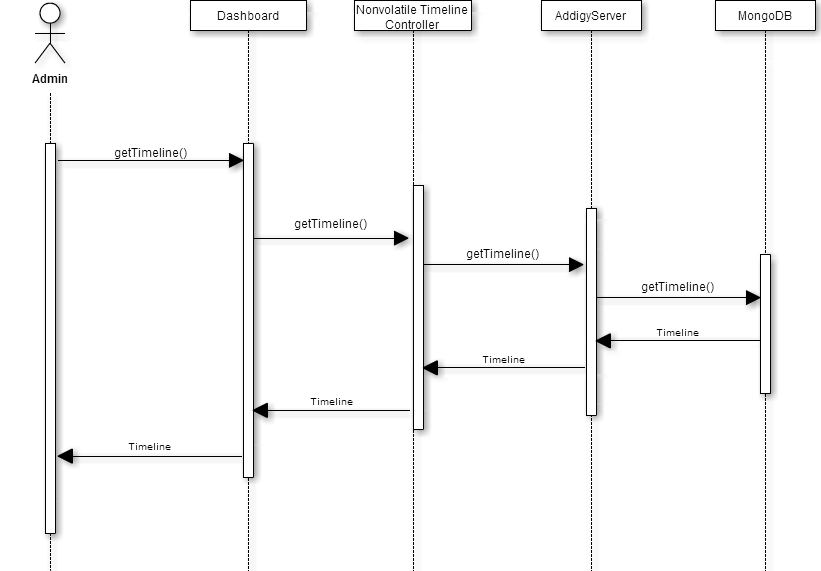


Figure #4.428 Timeline Sequence Diagram

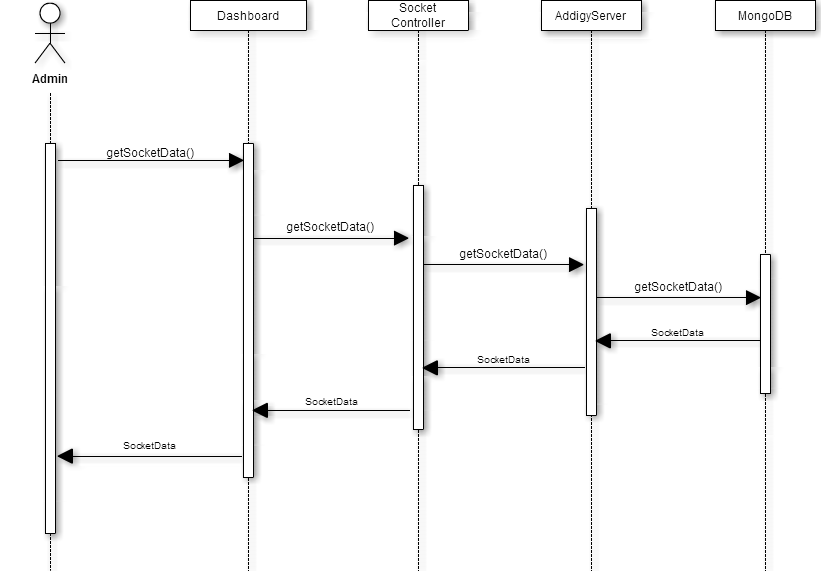


Figure #4.429 Socket Data Sequence Diagram

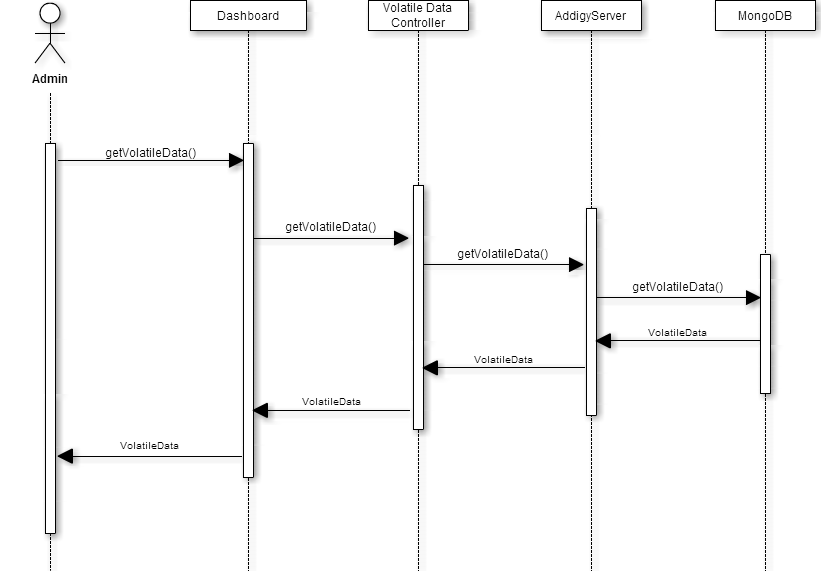


Figure #4.430 Volatile Data Sequence Diagram

## 

## 

## Appendix B - User Interface Design

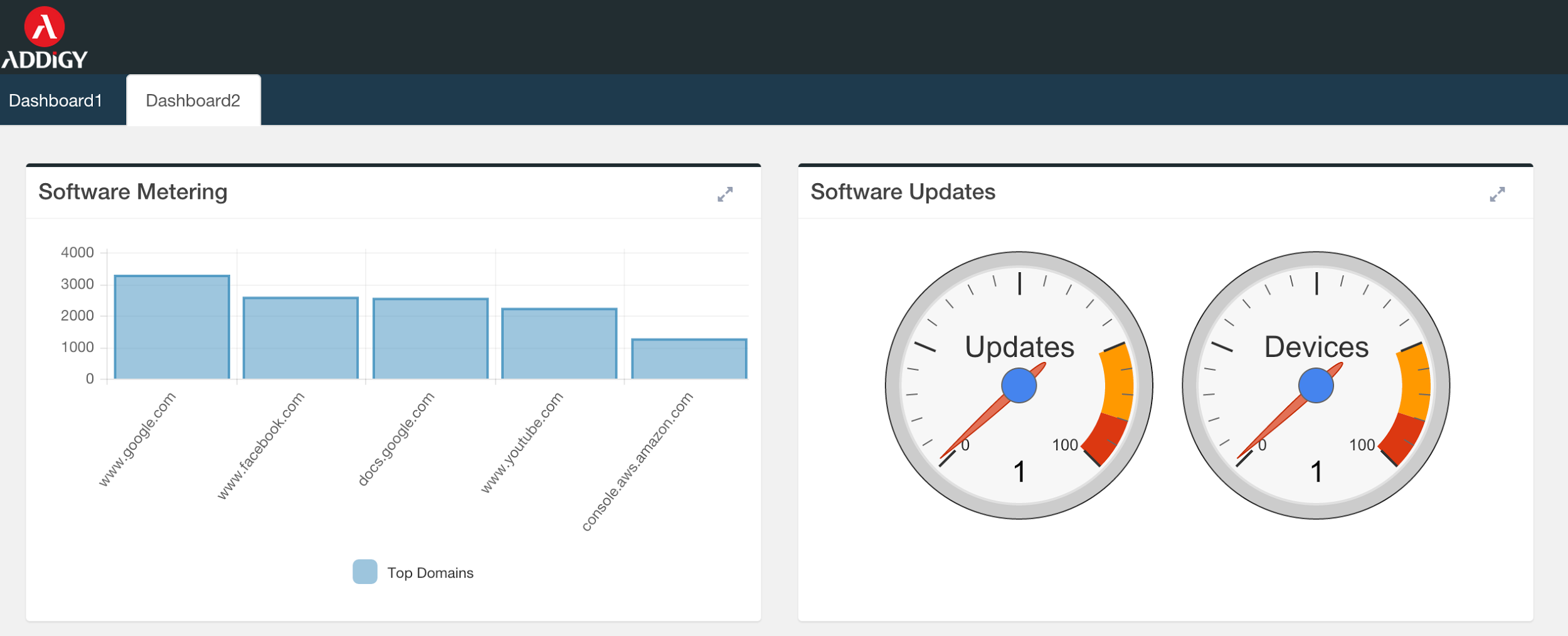
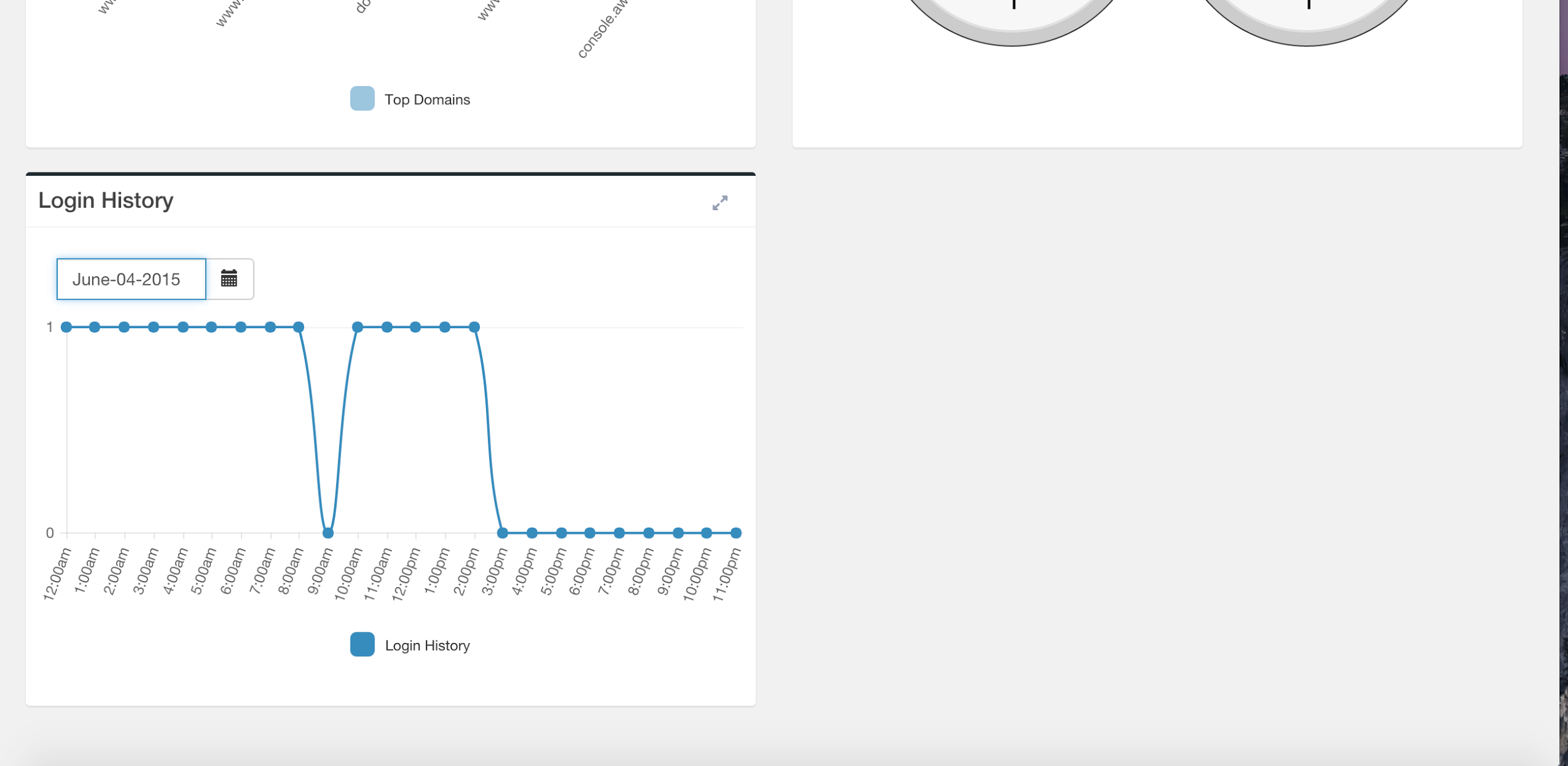
Figure B.1 Dashboard2

Figure B.2 Dashboard2(Login History)

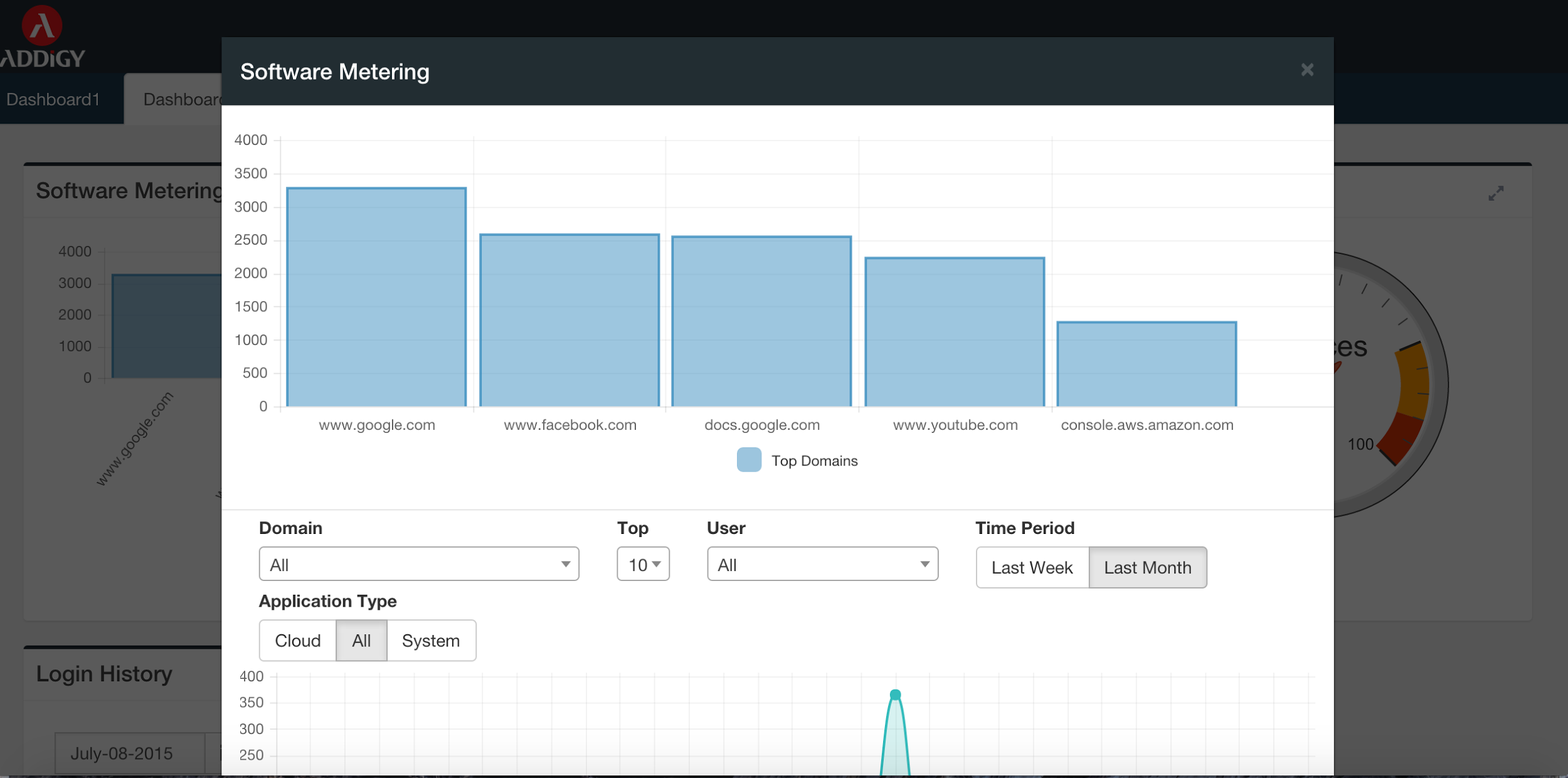


Figure B.3 Software Metering Top 5 Domains

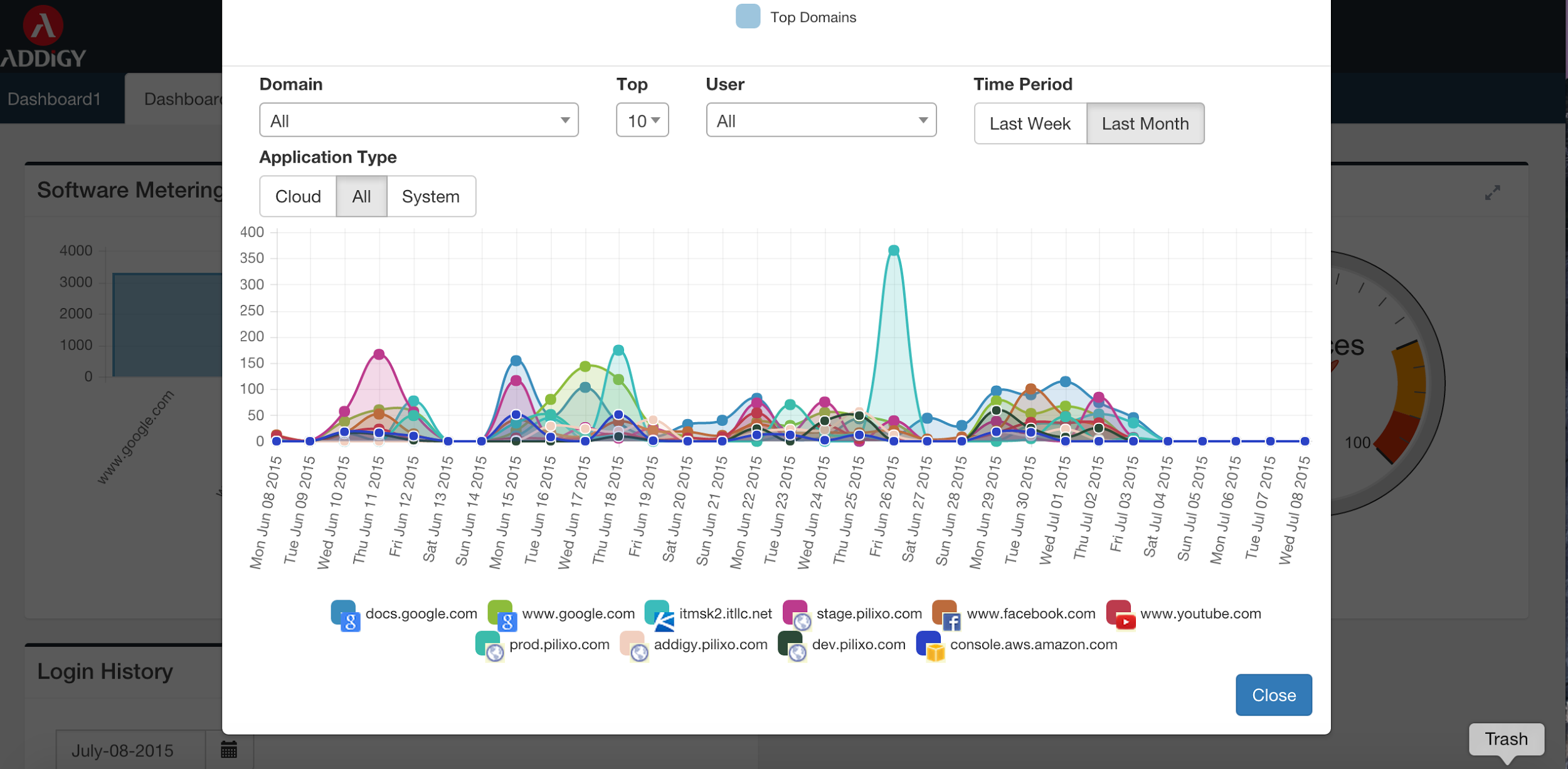


Figure B.4 Software Metering Filters

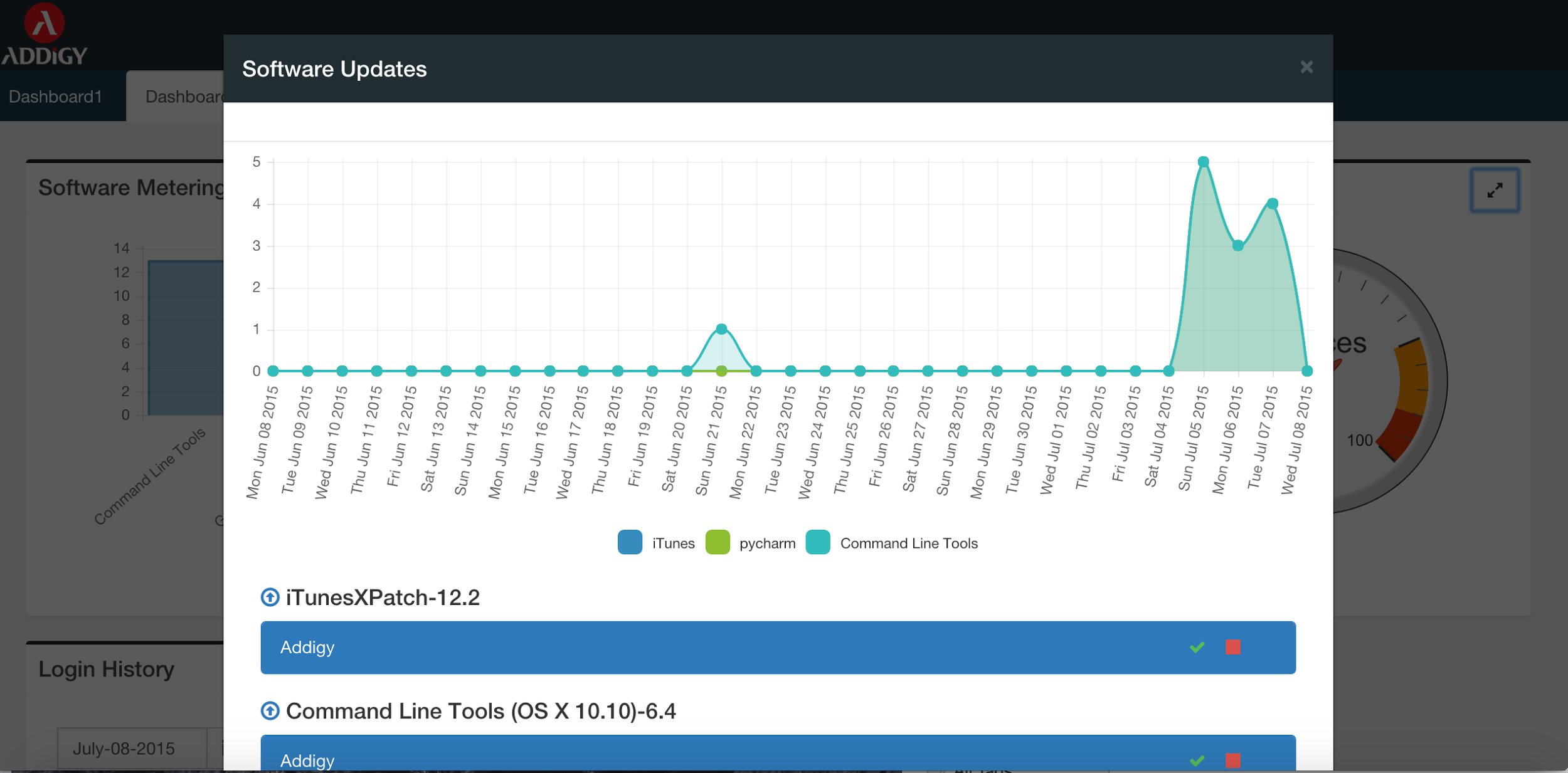


Figure B.4 Software Updates - Application Usage

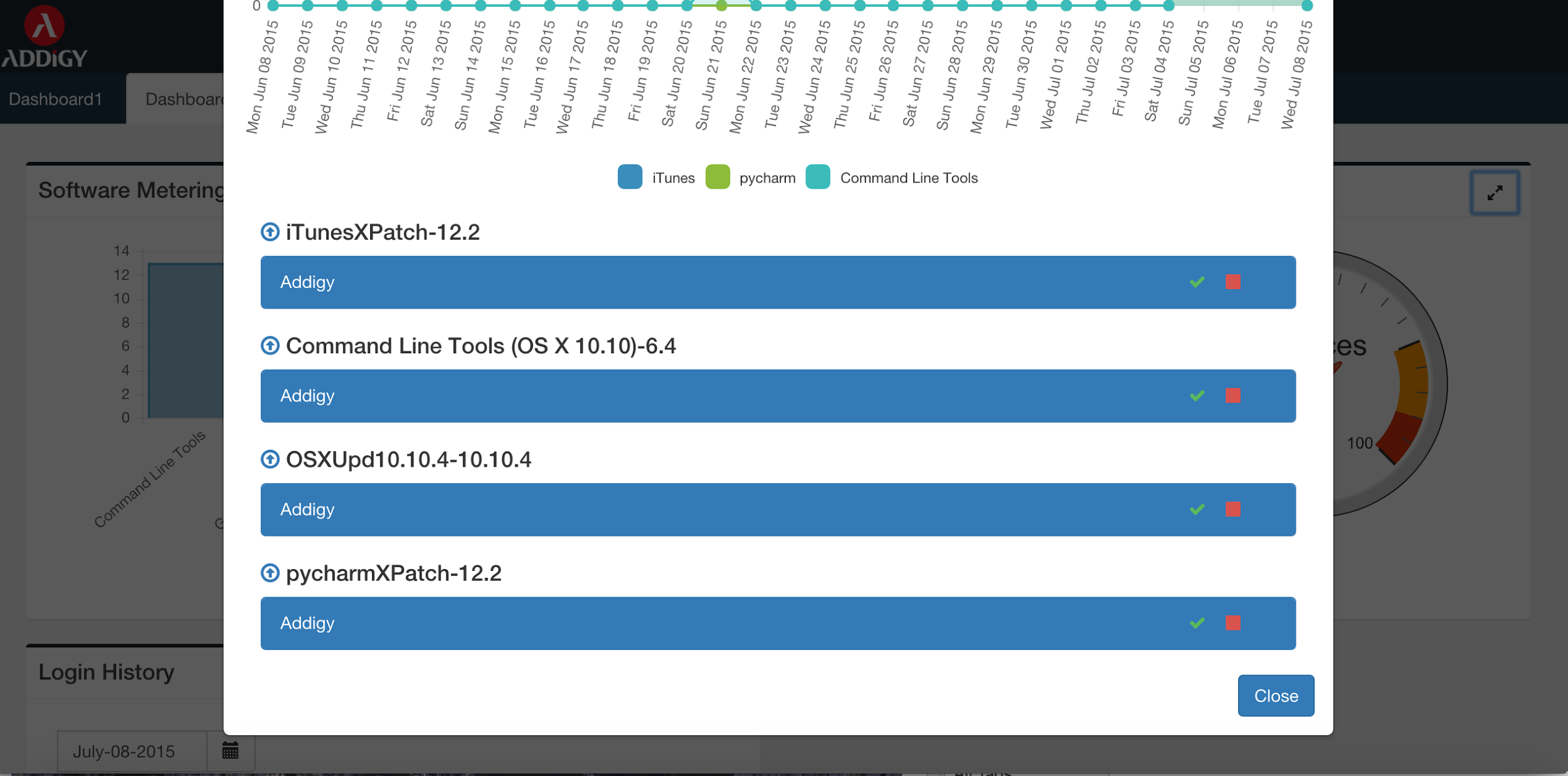


Figure B.5 Software Updates - Updates available

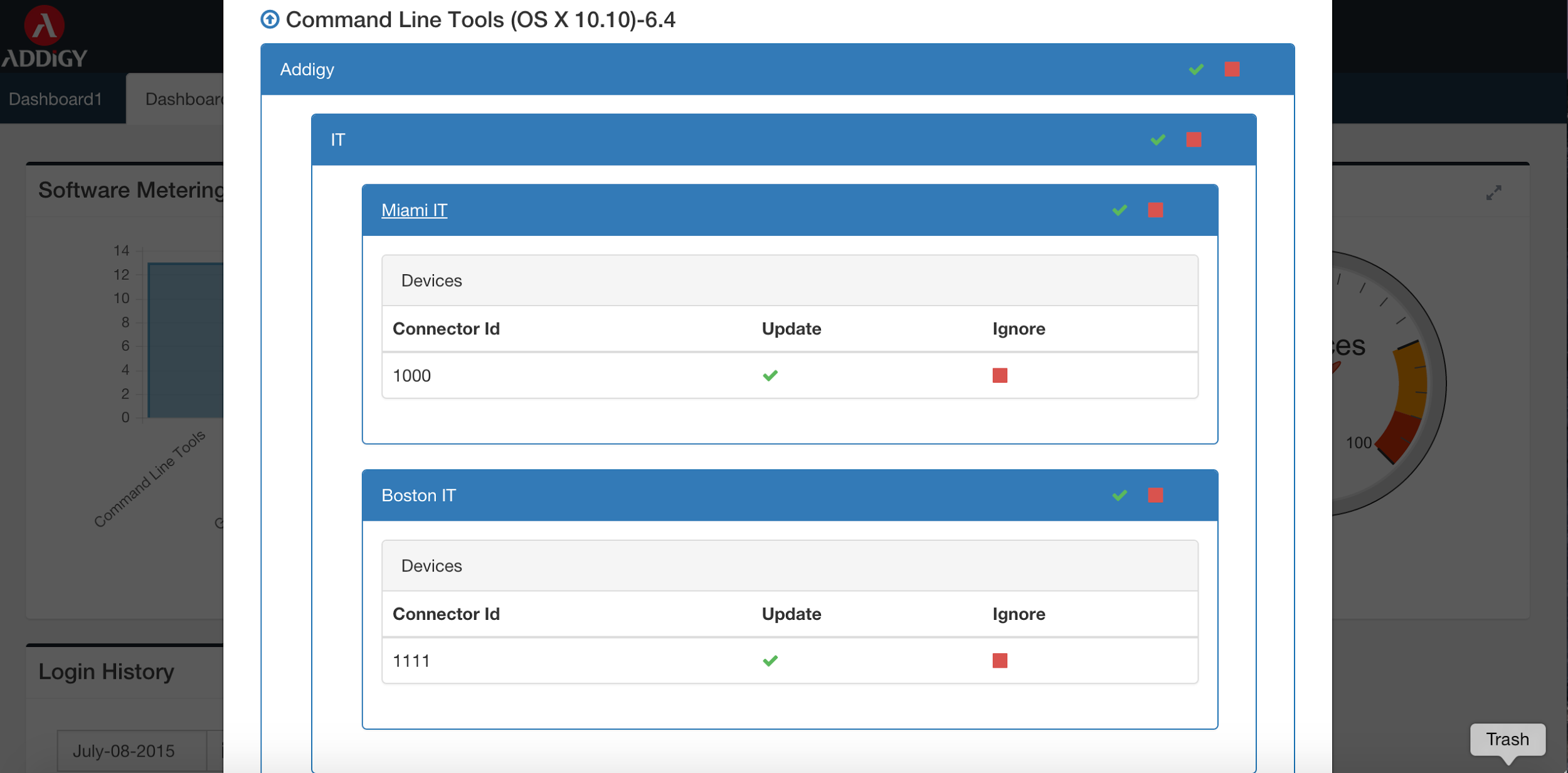


Figure B.6 Software Updates - Updates available policies hierarchy

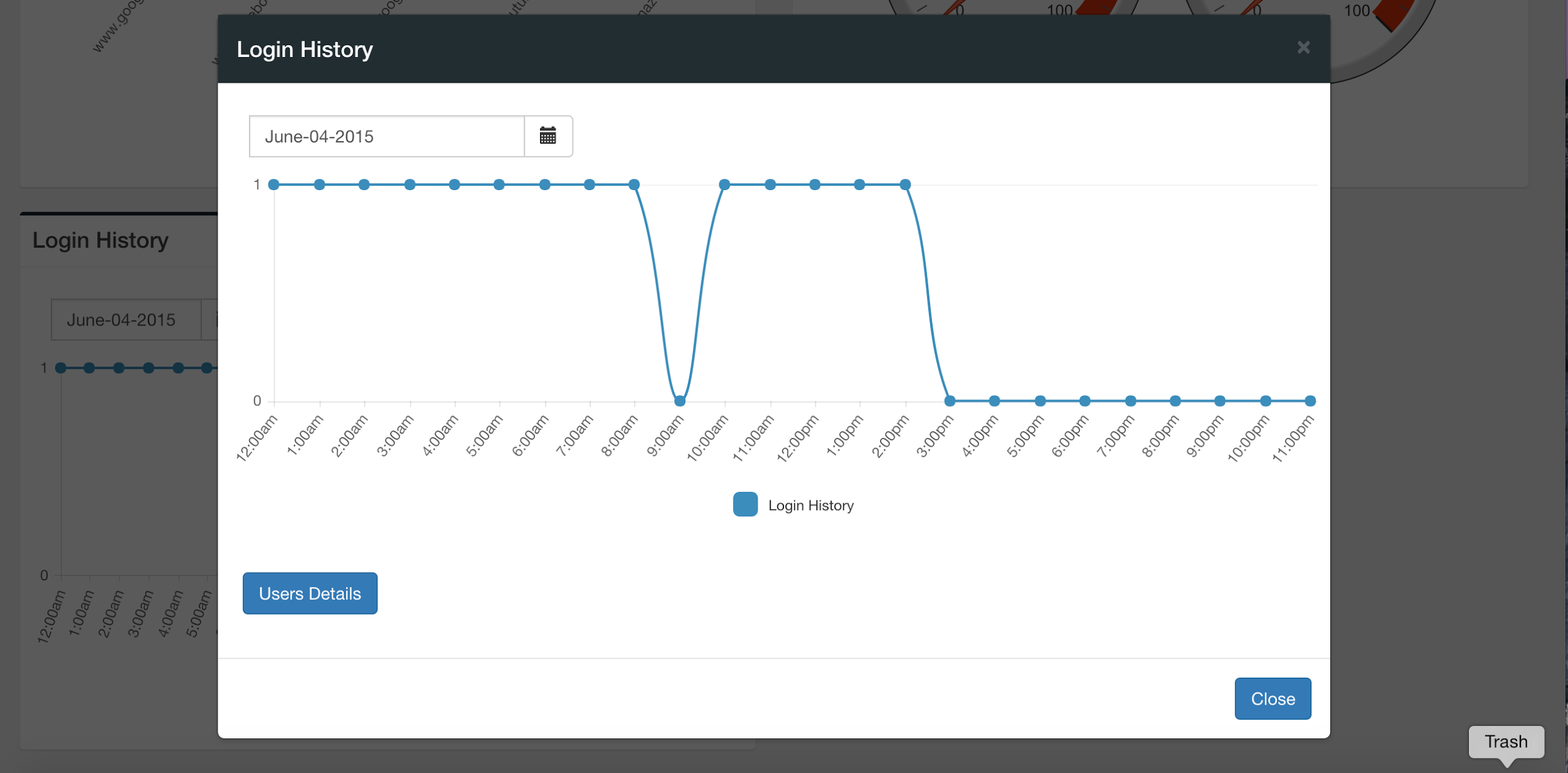


Figure B.7 Login History

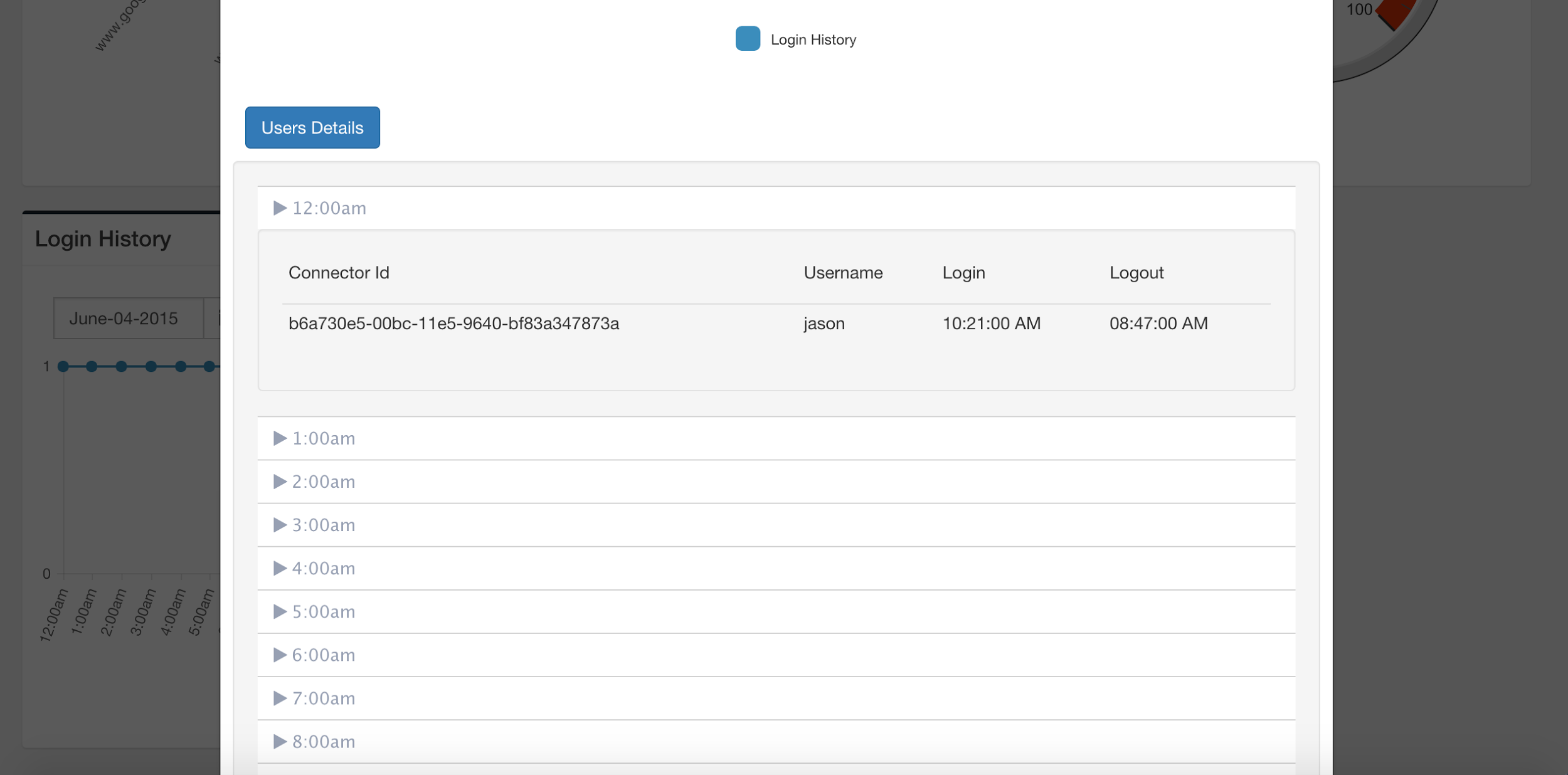


Figure B.8 Login History - User Details

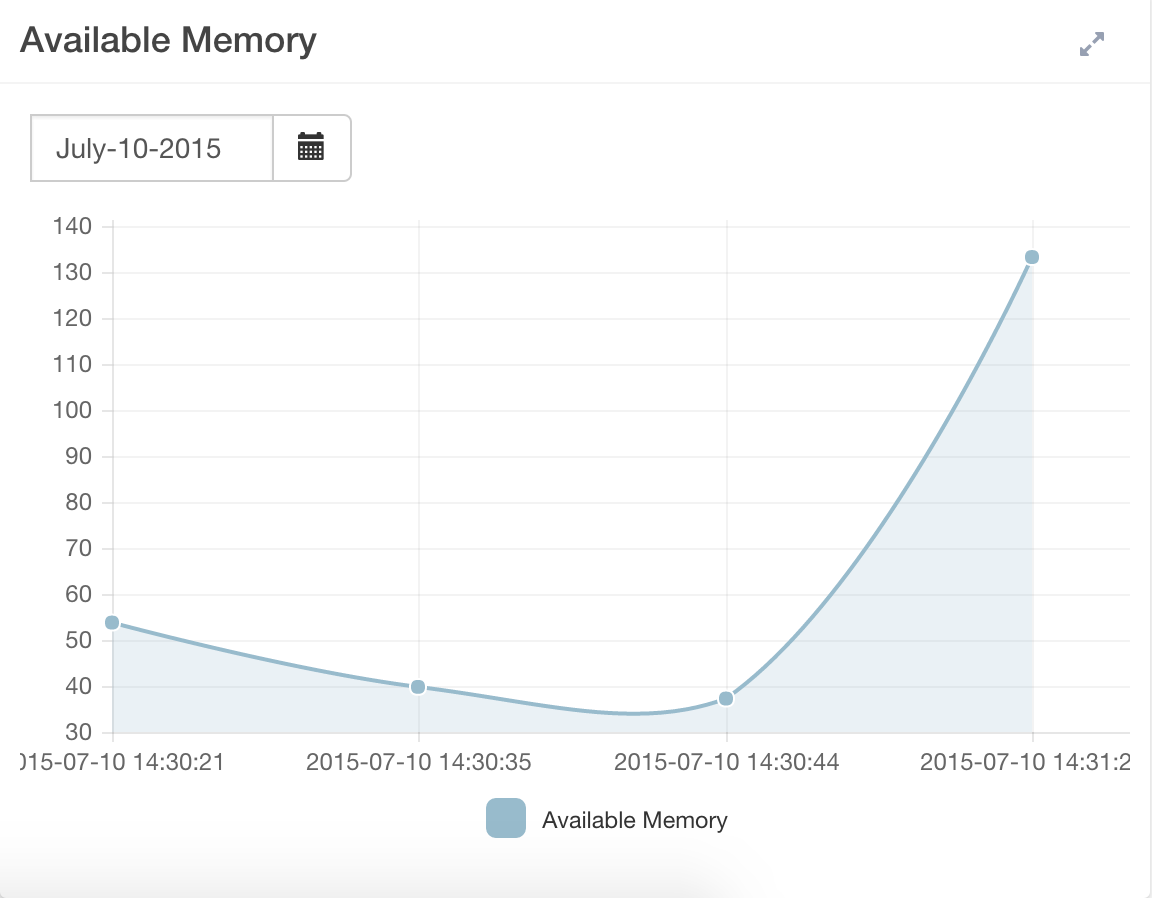


Figure B.9 Available Memory

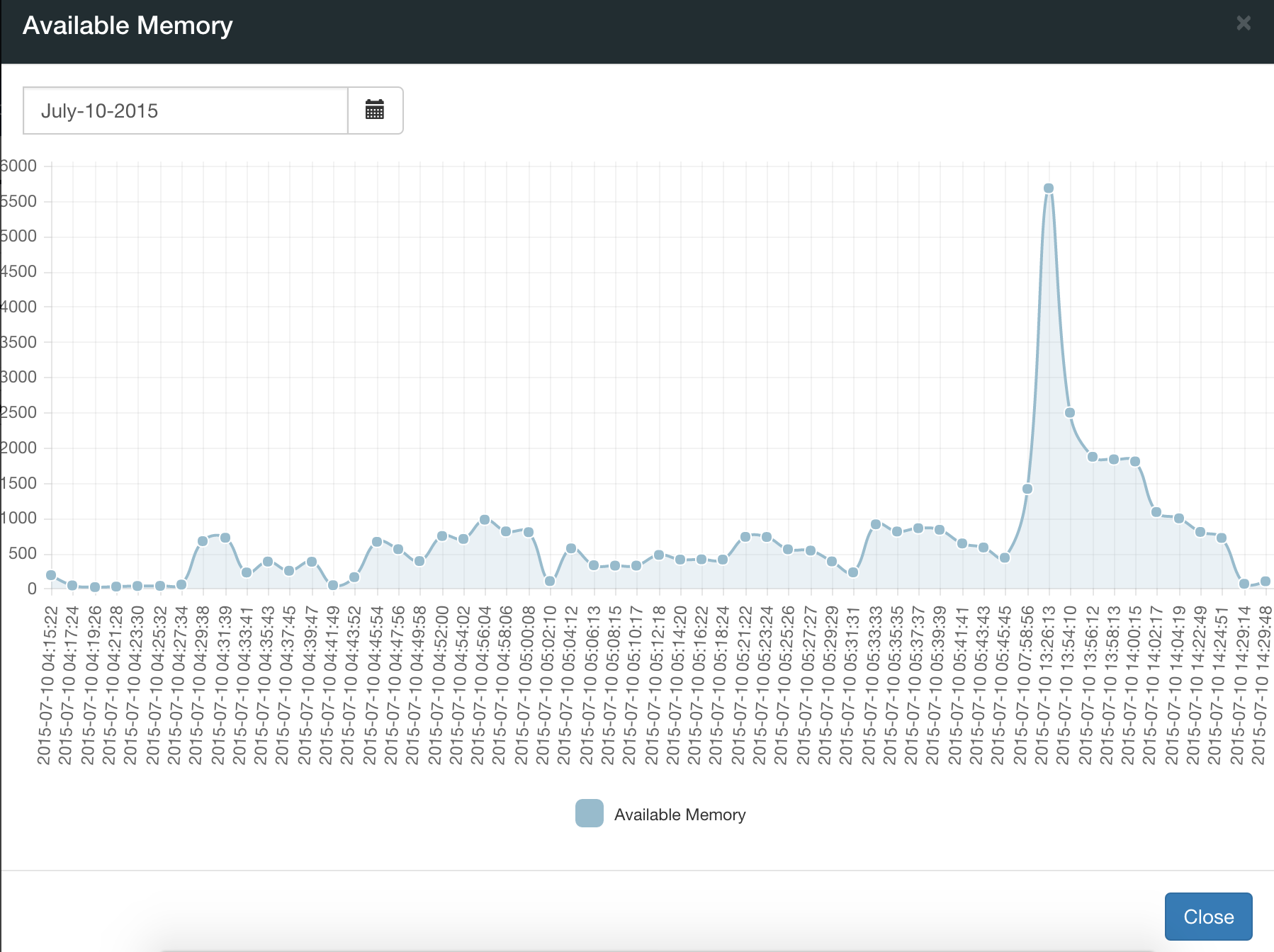


Figure B.10 Available Memory - Modal

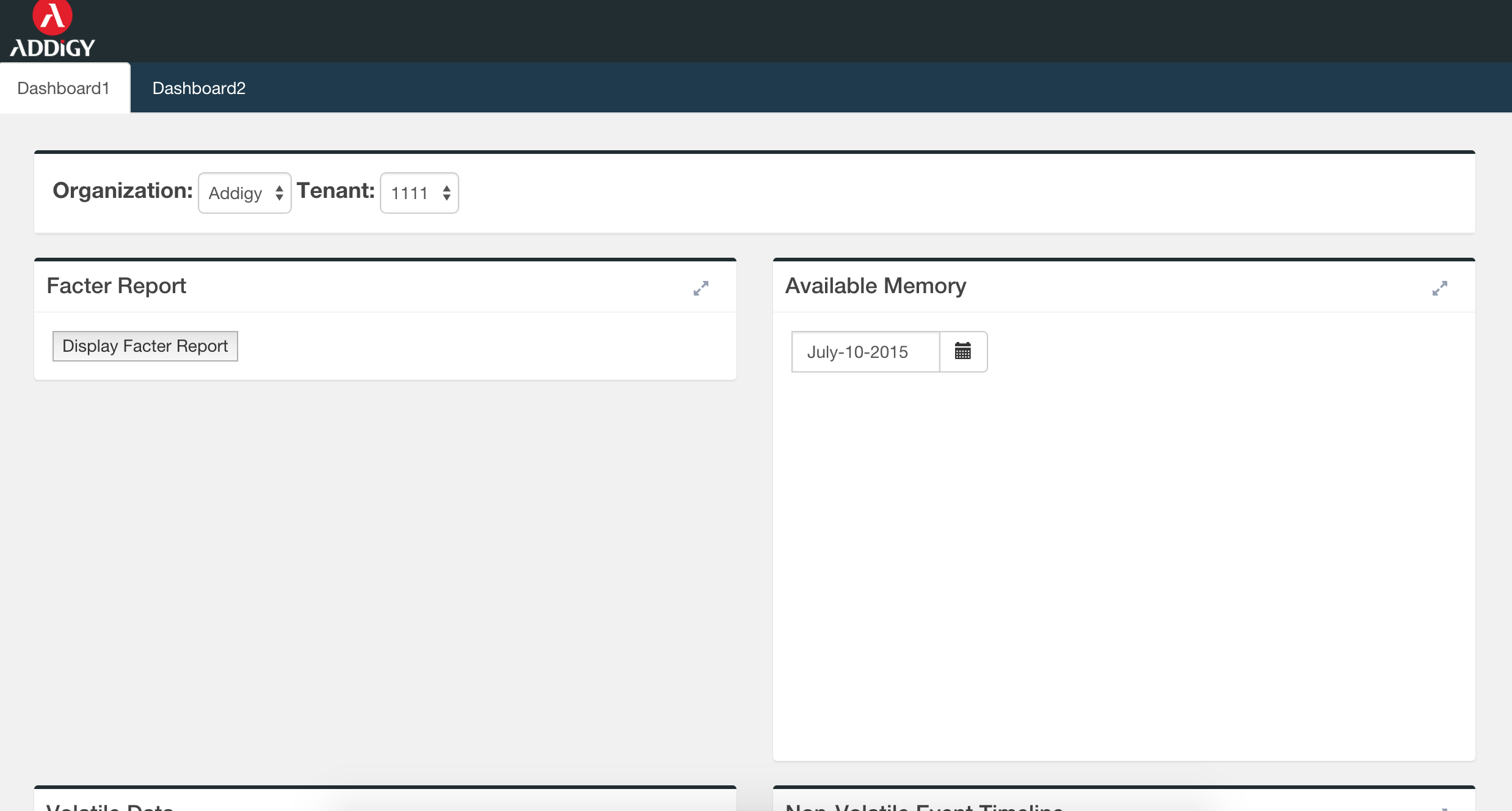


Figure B.11 Dashboard1

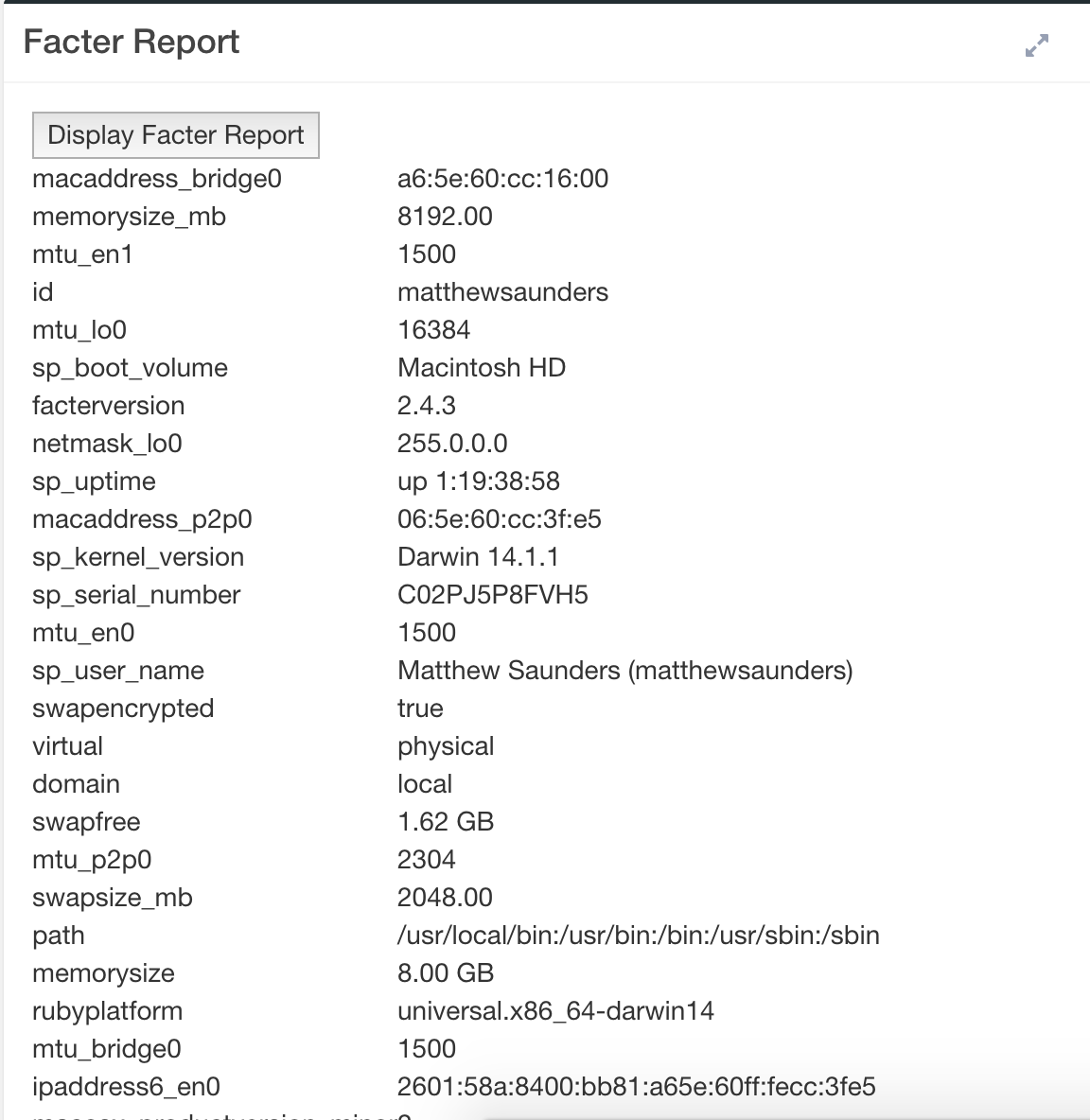


Figure B.12 Facter Report

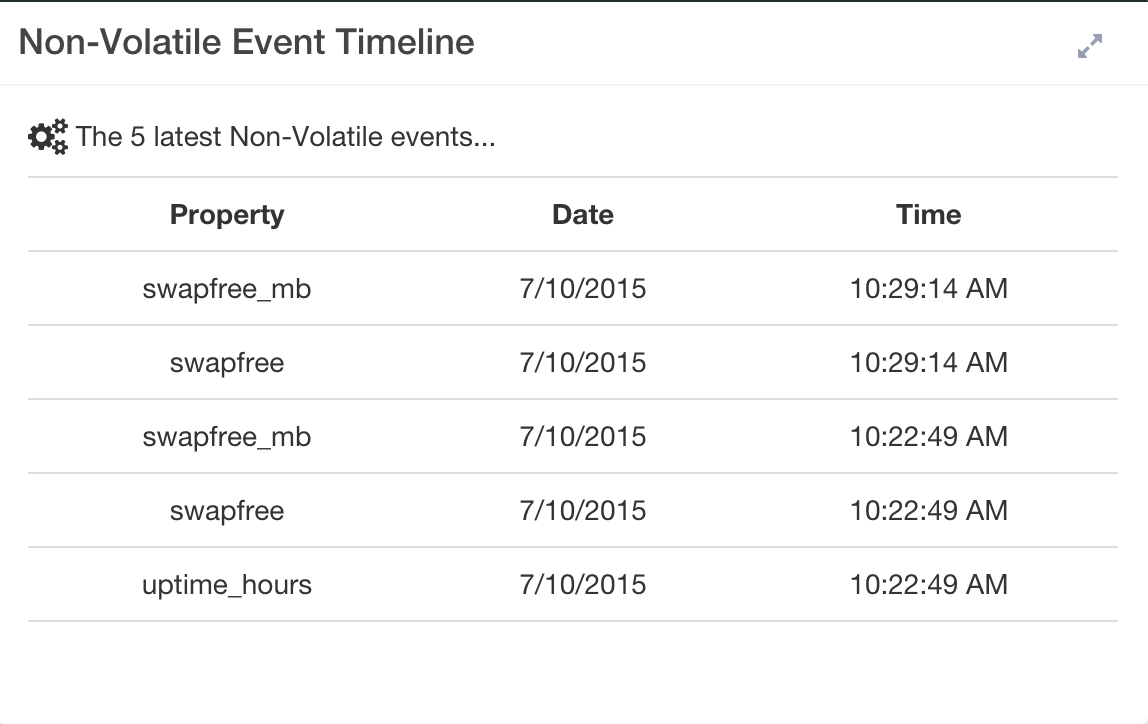


Figure B.13 Non-Volatile Event Table

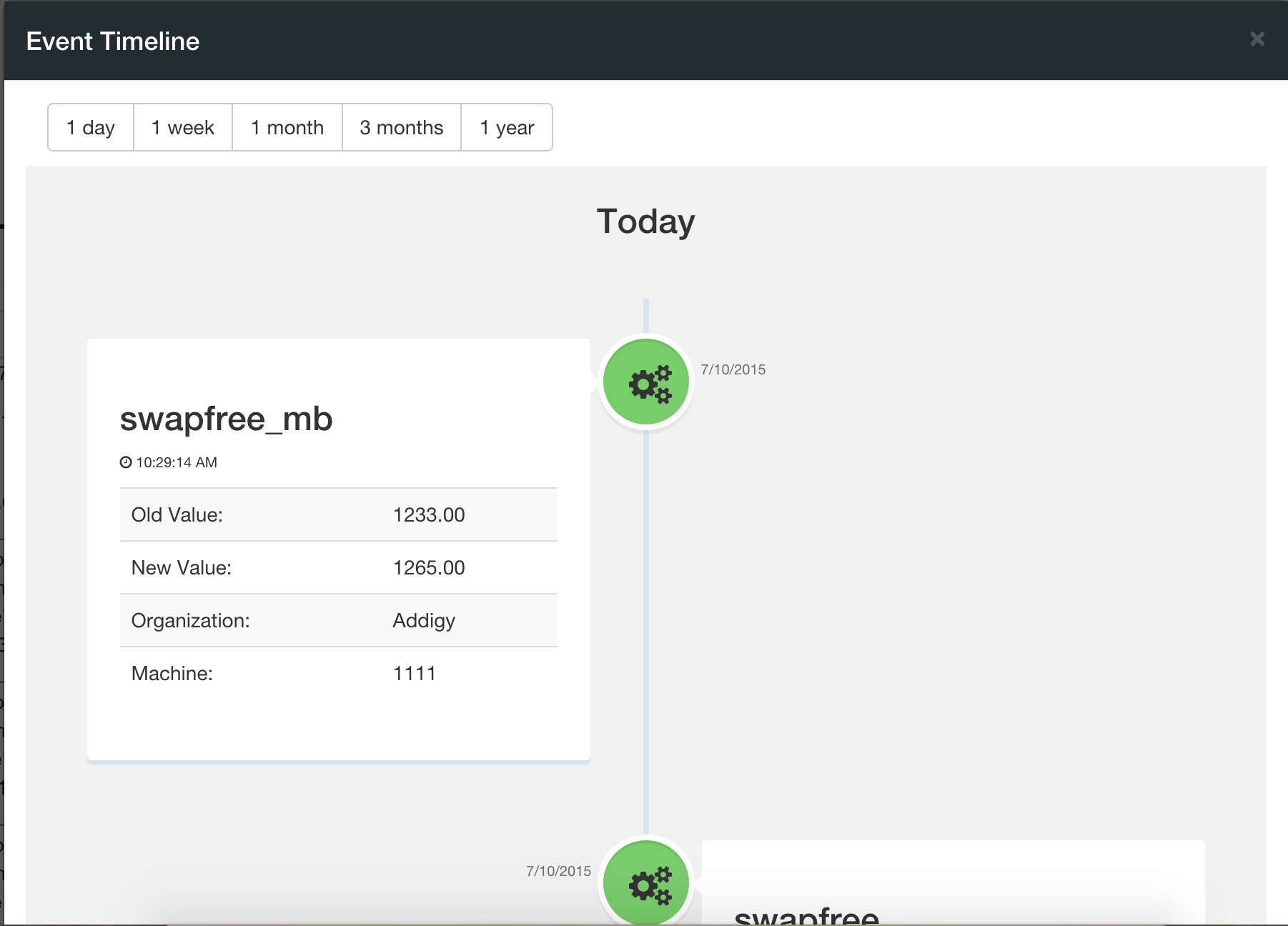


Figure B.14 Non-Volatile Event Timeline

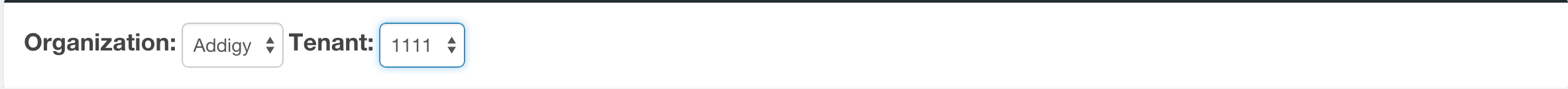


Figure B.15 Select Tenant

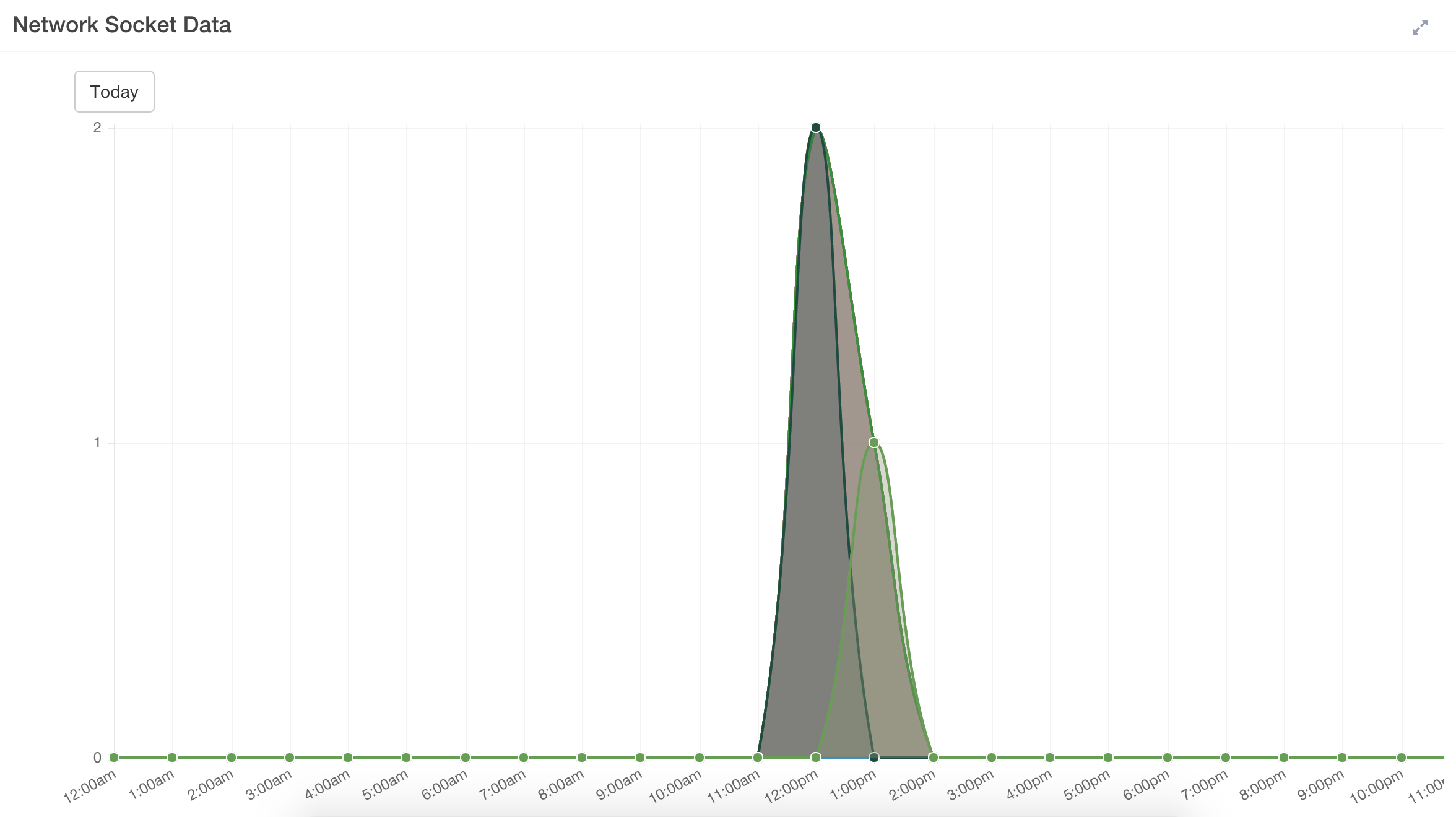


Figure B.16 Socket Data Graph

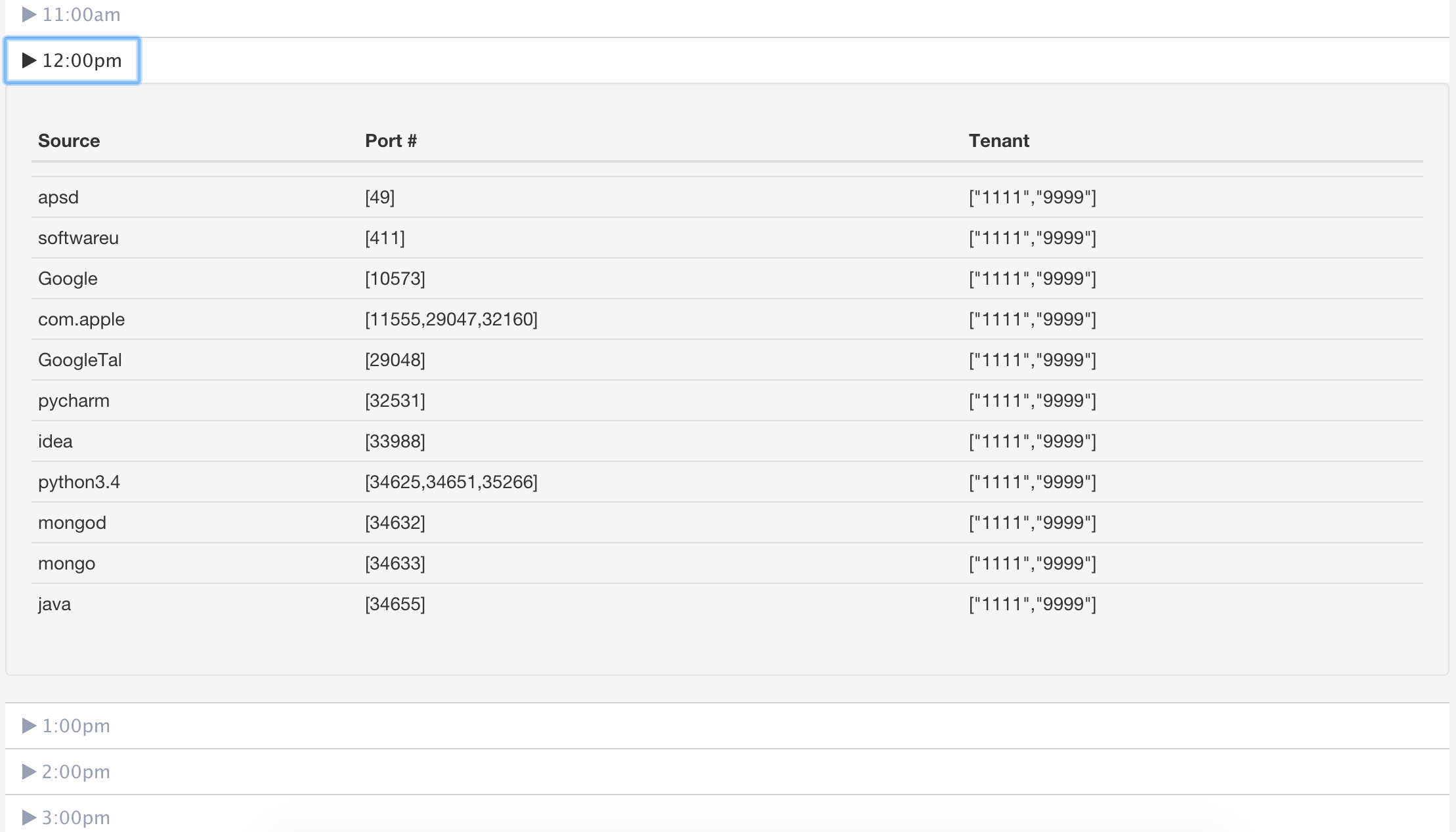


Figure B.17 Socket Data Table

## Appendix C - Sprint Review Reports

**Sprint 1 Report**

**Date:** May 29, 2015

**Attendees:** Masoud Sadjadi, Jason Dettbarn, [Matthew Saunders](https://plus.google.com/u/0/107583645216309754559?prsrc=4), Ayme Morrina

**Discussed Topics:**

Achieved:

* Infrastructure in the Java collector agent was created to allow team members to work independently on their tasks while having few conflicts when merging their work.
* A suitable tool for getting users login history was found.
* Login history was collected using a java client.
* Login history mechanism to cached login history was created. Only new data was collected and sent.
* MongoDB was installed and configured for authentication mode.
* PyMongo was installed for communication between the server and the database.
* Login history was stored on a MongoDb.
* AngularJS was integrated to the project to be user for the front-end client.
* Login history was shown as raw data on the front end.
* Hardware data was collected “facter" command.
* Collected hardware data was parsed into json format
* Hardware collected data was sent to server.
* Final Project deliverable was kept up to date with in the user stories sections and other sections regarding the product were updated as progress was made in the project.

Not achieved:

* For one of the members of the team setting up MongoDB was not achieved. Several repository changes were done increasing the time spent in reconfiguring the environments which in the specific case of Python was a bit complex task. The user story was moved to the next sprint.

**Sprint 2 Report**

**Date:** June 12, 2015

**Attendees:** Masoud Sadjadi, Jason Dettbarn, [Matthew Saunders](https://plus.google.com/u/0/107583645216309754559?prsrc=4), Ayme Morrina

**Discussed Topics:**

Achieved:

* Created client-side mechanism to display returned data from the database.
* Created client-side mechanism to request stored data in the database
* Created server-side infrastructure to get data from database.
* The login collector code was refactored to better adjust for later searching needs.
* A collapsible menu displaying login user per hour details was added.
* The UI for Login History Section was improved.
* Create a modal mechanism to display details of the selected graph in the main page.
* Research and integration of chart.js to be used as graphing tool.
* Allowed for date selection to display login history.
* Researched tools for collecting information regarding cloud based applications usage
* Collected Chrome browsing history using sqlite db.
* Collected data sent to server and successfully stored in the db.

Not achieved:

* Everything proposed for this Sprint was achieved.

**Sprint 3 Report**

**Date:** June 26, 2015

**Attendees:** Masoud Sadjadi,Jason Dettbarn, [Matthew Saunders](https://plus.google.com/u/0/107583645216309754559?prsrc=4), Ayme Morrina

**Discussed Topics:**

Achieved:

# Display top visited domains

# Create filter mechanism for displaying domain usage info

# Extend domain history collector to include Mozilla Firefox browser history

# Extend collector to get browsing history of each user of the system

# Set Up fiu server environment for demos

# Extend domain history collector to include Safari browser history

# Extend collector to incorporate system applications

# Display Timeline of Available Memory

* Final Project deliverable was kept up to date with in the user stories sections and other sections regarding the product were updated as progress was made in the project.

Not achieved:

* No static and/or dynamic diagrams have been added to the final project document.

**Sprint 4 Report**

**Date:** July 10, 2015

**Attendees:** Masoud Sadjadi, [Ma](https://plus.google.com/u/0/107583645216309754559?prsrc=4)tthew Saun[ders](https://plus.google.com/u/0/107583645216309754559?prsrc=4), Ayme Morrina

**Discussed Topics:**

Achieved:

* Add Software Updates Collector
* Display graph/animation of available updates count and machine count
* Display tree structure of information of available updates in the system
* Collect real information from different device.
* Refactor Software Updates database structure
* Add updates/application usage graph
* Allow user to select which computer to display from organization
* Query MongoDB for Volatile Data over time
* Display timeline of events when non-volatile characteristic change
* Final Project deliverable was kept up to date in the user stories sections.
* Dynamic and static UML diagrams were added to the final deliverable.

# 

Not achieved:

* Testing to the existing features and add the test cases and results to the documentation.

**Sprint 2 Report**

**Date:** July 24, 2015

**Attendees:** Masoud Sadjadi, Jason Dettbarn, [Matthew Saunders](https://plus.google.com/u/0/107583645216309754559?prsrc=4), Ayme Morrina

**Discussed Topics:**

* Achieved:
* Added last day filter to software metering tool.
* Modify collector to take in a list of parameters.
* Create Facter data injector.
* Manual Testing for all the features.
* Bug fixes were done.
* Update final progress document with test cases, new sequence diagrams, new user stories, architectural and design patterns and class and system diagrams.
* Plan and Practice presentation.

Not achieved:

* Everything proposed for this Sprint was achieved.

## Appendix D - Sprint Retrospective Reports

**Sprint 1 Retrospective**

**Date:** May 29, 2015

**Attendees:** Masoud Sadjadi, Jason Dettbarn, [Matthew Saunders](https://plus.google.com/u/0/107583645216309754559?prsrc=4), Ayme Morrina

**Discussed Topics:**

* Successfully created a project structure that allowed members of the team to independently work on their tasks and integrate their work with low amount of conflicts.
* Team realized on time that the DynamoDB (chosen to store the data) was not suitable for the type of processing required later. MongoDB was chosen to substitute DynamoDB and the changes were made early in the sprint so there were not implications of the change.
* Most user stories destinated for this sprint were completed.
* Final Project deliverable was kept up to date with in the user stories sections and other sections regarding the product were updated as progress was made in the project.
* Extra time was spent due to change of repositories. Each time the repository was changed the Java and Python environment needed to be reconfigured which was time consuming specially for the python one. This should not be a problem anymore since not more changes of repositories should be needed.
* Mingle was not kept up to date during the first week of the sprint. Tasks were not being added to the user stories and points were not being added to the user stories. Members agreed to keep up to date mingle to reflect their daily work.
* Need to better communicate setup/infrastructure difficulties with team for assistance.

**Sprint 2 Retrospective**

**Date:** June 12, 2015

**Attendees:** Masoud Sadjadi, Jason Dettbarn, [Matthew Saunders](https://plus.google.com/u/0/107583645216309754559?prsrc=4), Ayme Morrina

**Discussed Topics:**

* Successfully created a front-end structure that allowed members of the team to independently work on their tasks and integrate their work with low amount of conflicts.
* All user stories destinated for this sprint were completed.
* Final Project deliverable was kept up to date with in the user stories sections and other sections regarding the product were updated as progress was made in the project.
* Login Feature was completed.
* Domain History monitoring feature was started.
* User stories were not assigned priority. For future sprints priority should be set up to the user stories at the creation moment so team members can prioritize tasks to be achieved.

**Sprint 3 Retrospective**

**Date:** June 12, 2015

**Attendees:** Masoud Sadjadi, Jason Dettbarn, [Matthew Saunders](https://plus.google.com/u/0/107583645216309754559?prsrc=4), Ayme Morrina

**Discussed Topics:**

* Browsing History Feature was completed.
* Applications were incorporated to browsing history feature.
* Need to display timeline of non-volatile facter report facts.
* Need to run collectors to collect real information to show on dashboards
* Need to work on documentation diagrams. For the next Sprint dedicate time to work on documentation diagrams.

**Sprint 4 Retrospective**

**Date:** June 12, 2015

**Attendees:** Masoud Sadjadi, Jason Dettbarn, [Matthew Saunders](https://plus.google.com/u/0/107583645216309754559?prsrc=4), Ayme Morrina

**Discussed Topics:**

* Software Updates Feature was completed.
* Applications usage of those related to the updates are shown together with the updates.
* Need to on testing. For the next Sprint dedicate time to do system testing and add test cases and results to final deliverable.
* Collector was run in a company machine and real data was able to be collected.
* Need to wrap up all coding in first week of next sprint to spend second week testing, fixing minor bugs and finishing documents.

**Sprint 4 Retrospective**

**Date:** July 24, 2015

**Attendees:** Masoud Sadjadi, Jason Dettbarn, [Matthew Saunders](https://plus.google.com/u/0/107583645216309754559?prsrc=4), Ayme Morrina

**Discussed Topics:**

* This sprint the code was wrapped up and finished by the first week of the sprint which allowed to do manual testing the second part of the sprint and successfully determine and fix a few bugs regarding some of the tools.
* The sprint was well planned so time would be spent on planning and preparing presentation demo.
* The program was successfully run in a different environment being able to collect the data and store it on the db.
* The document was successfully updated and finished.
* Last week is necessary to reserve some time for presentation practice and to collect recent data for the demo.

# **References**

You must reference any work that is not your own.