
# Cogniyug

A Real time BIG Data Analytics Platform for gaining Actionable Intelligence using Time Series Machine Data

---

# User Guide

September 2013

# Contents

## Overview

This document provides an introduction to 'Cogniyug', a real-time analytics platform for analyzing time series machines data. This document covers key features of Cogniyug and explains its day-to-day usage mainly from the end user's perspective.

## Intended Audience

If you use 'logs' of any kinds in your day-to-day operation, you should use this guide to understand how Cogniyug could help you to make most out of your log data. Any business generating machine data in the form of logs needs a data analytics platform like Cogniyug. We live in the 'Data Age' and it is imperative to use our own data to our own benefit. Cogniyug provides 'actionable intelligence' in 'real time' unleashing the power of your own machine data.

## Introduction

### What is Cogniyug?

In simple words, Cogniyug is an analytics platform build by TechLineage for making decisions based on facts hidden in your ***'machine data'***. We believe that your machine data is worth its weight in gold! Hidden in it are interesting facts that no naked eye can see easily. Cogniyug reveals these facts thereby allowing you to improve the Availability and QoS of your business services. It provides actionable intelligence that enables you to make quick business decisions to seize opportunities for maximizing revenue and improving the bottom line.

### What is '*machine data*'?

'Machine data' is generated by computing equipments such as Computers, Network Switches, Routers, Storage devices, etc. or non-computing equipments such as air conditioners, temperature sensors, fire alarm systems, etc. in the form of logs. Cogniyug is capable of consuming, storing, indexing and analyzing the data from diverse sources. With a patent pending algorithm at its core, Cogniyug finds patterns from this Big Data providing actionable insights in real time.

### Capabilities at a glance

- ✓ **Diverse data sources and formats:** Cogniyug is vendor agnostic. Essentially, it does not enforce any restriction on the format of the data. It can consume data from any data source, in any format. This makes it flexible and powerful enough to handle vast amount of data from diverse data sources and analyze it together.

  Consider an online shopping portal with a couple of web servers, application servers and data base servers connected with some networking equipments such as routers, load balancers, switches, etc. and some SAN/NAS storage it its backend. All these components generate 'logs' in their own proprietary formats. It is important to analyze all the logs together and hence it is important that the tool is vendor agnostic. Cogniyug is capable of consuming logs from all such different sources and analyze them together.

  (Please refer to section Cogniyug Data Format for understanding more about the diverse data sources and how you can integrate it seamlessly with Cogniyug)

- ✓ **Indexing and searching:** Seamless search with strong grammar support helps to find exactly what you are looking for with time relevance.

  Fast and easy to use search of Cogniyug renders all the required information in real time. Cogniyug supports complex expressions and grammar (like ANDing / ORing etc) so that you can tune your searches to your need. You may save your favorite 'searches' (with complex expressions and time boundaries, etc.) and re-use quickly when you need them.

  (Please refer 'The Search' section for more information about search, supported grammar and other capabilities)

- ✓ **Causal Analysis:** A single click Root Cause Analysis capability of Cogniyug is priceless as it reduces Mean Time to Resolution (MTTR) from hours to minutes. Complex causes can be analyzed and visualized with confidence measures, support levels and time relevance.

  Consider the example of the online shopping portal mentioned earlier. Assume a scenario where ten (10) out of two thousand (2000) users were not able to complete the transactions in last one month. If you were to find the exact cause of such an event, you may have to go through large amount of logs generated by all the components (web server, application servers, database, routers etc) and still, you may not be able to figure out 'why a particular user was not able to complete the transaction'. Cogniyug's Root Cause Analysis (RCA) capability will help you perform your analysis in seconds and arrive at the most likely probable cause/s real quick.

  (Please refer to 'Causal Analysis' section below)

- ✓ **Predictive Intelligence:** Cogniyug's solid mathematical model predicts key business and infrastructure events with precise accuracy. This helps businesses to seize opportunities to improve top line and minimize losses by taking appropriate preventive measures beforehand.

  Again, consider the example of the online shopping portal mentioned earlier. Won't it be useful to predict the critical event such as 'a user not being able to complete the transaction' before it actually happens and affects the business? Cogniyug comes with out-of-box prediction algorithm that can be used for predicting key events beforehand.

  (Please refer to section on Defining Predictions below to know more about it)

- ✓ **Find deviations from desired trends:** Cogniyug spots deviations from desired/normal trends in real time. This helps in preventing undesired consequences or to fine tune the processes.

  It is important to discover and track the expected patterns depicting the normal operating conditions. Consider an example where a server boots up in the morning, acquires an IP address, mounts certain files systems, performs some pseudo transactions before it becomes production ready. This 'pattern' depicts normal behavior of the system and any deviation from this would be undesirable. Cogniyug is capable of spotting such 'patterns' and detect deviation from them.

  (Note: - This feature has been disabled in the evaluation version)

- ✓ **Effect Analysis:** With knowledge of patterns, Cogniyug is capable of ascertaining effects of critical events with a single click of mouse. This empowers you to make informed decisions before making a change.

  Sometimes, it is important to know the effects of certain events and may help you to make informed important decisions. Consider the above example of 'online shopping portal' where in the database administrator is planning to perform a major backup operation on a database instance. It will be extremely helpful to find the effects of such events before they actually happen.

  (Please refer to section on 'Effect Analysis' below)

- ✓ **Complex Business Intelligence Reports:** Unstructured log data can be used to generate complex Business Intelligence reports for improving operational efficiency, reducing cost, etc.

  Every business needs various reports and most of the reports can be accurately computed using the knowledge of the archived logs. Cogniyug's reporting capability helps you to externalize the stored data and perform complex operations in the any programming/scripting language that you are comfortable with.

  (Note: - This feature has been disabled in the evaluation version)

We hope, by now you have understood what to expect from Cogniyug and perhaps have a little of bit of idea of its key features. The chapters below will walk you through all the features and explain each one in more depth. We will begin with download instructions, followed by installations, usage and gradually move on to the advanced topics such as writing your own Hooks, Administrations, Developing reports etc.

## Pre-requisites and Installation

This section assumes that you have downloaded the evaluation version of Cogniyug. If not, please write to us on support@techlineage.com and we will send you the details for downloading the evaluation version of Cogniyug.

(Skip this section if you have already gone through Quick Start guide and installed Cogniyug on a Linux server)

Note: Cogniyug is supported on only Linux (x86_64-bit) at the moment. You may run Cogniyug on Red Hat Enterprise Linux 6.x or Cent OS 6.x. Though we have not tested on other flavors of Linux such as Fedora, Ubuntu or SuSe, we do not anticipate issues running Cogniyug on these platforms. In case you face any issues while installing or running Cogniyug on these platforms, please feel free to write to us on support@techlineage.com and we will be happy to help you out.

Limitations with the Trial / Evaluation version of Cogniyug: Trial Version of Cogniyug comes with certain limited functionalities. Some of key limitations are as follows

1. Cogniyug stops working after 90-days (you may request for extension by writing to us on support@techlineage.com)
2. Distributed installation is not supported with the evaluation version.
3. The 'Watch Point' functionality will not work with the evaluation version.
4. The 'Reporting' functionality does not work with the evaluation version.
5. The evaluation version does not support multiple users.
6. The evaluation version does not support multiple Transformers.
   (If you are using a licensed version of Cogniyug, these limitations will not affect you.)

## Installing Cogniyug

As mentioned above, Cogniyug is supported to run on Red Hat Enterprise Linux 6.x or Cent OS 6.x (x86_64-bit) only. We recommend following server configuration to successfully install and run the evaluation version of Cogniyug on a single Linux server.

| Operating System | RAM (Memory) | CPU | Disk Space | Network Configuration |
|---|---|---|---|---|
| Red Hat Linux 6.x Or Cent OS 6.x (x86_64-bit) | 12 GB or more | 4 virtual cores or more | 50 GB on the partition on which Cogniyug is installed | Static IP address to the server on which Cogniyug is being installed |

If you are using licensed (commercial) version of Cogniyug, you will need to refer to the Advanced Installation Options section in the Administration Guide to plan your deployment.

Note: - Evaluation version of Cogniyug is meant to run on a single Linux server for simplicity. The goal is make you aware of all the capabilities of Cogniyug without complicating the installation and deployment. Remember that Cogniyug is an analytics application that performs a lot of computation on the data you import in it. We recommend you to provide adequate resources, mainly memory and CPUs for Cogniyug to function smoothly.

The evaluation version of Cogniyug does not pose any restrictions on the amount of data that the user can process. However, it is to be noted that the amount of data is directly proportional to the computational resources required by the server (mainly Memory and CPU). Simply meaning, more data would need more resources.

Copy the download installer file INSTALLER_COGNIYUG_1.0.00.tar.gz to the Linux server.

*Hint: If you have downloaded* INSTALLER_COGNIYUG_1.0.00.tar.gz *on your Microsoft Windows computer, you may use a scp or equivalent command to copy the file on the target Linux server. If you use ftp/sftp to transfer the file, make sure you use the binary mode while transferring the file windows to linux.*

The following section assumes that you have successfully downloaded INSTALLER_COGNIYUG_1.0.00.tar.gz on the Linux server.

Perform following steps to

1. Go to the directory where installer is copied
2. Create a directory where you want to install Cogniyug. (Ensure that you have sufficient space on the filesystem)
3. Extract the installer using tar -xzf INSTALLER_COGNIYUG_1.0.00.tar.gz command. This will create a directory 'installer' under which Cogniyug installer will be extracted. Go the installer directory (cd installer)
4. Run "setup.sh" file to start the installation of Cogniyug.
5. This is a script based installation and hence does not need any X-server support. You can run it from your terminal
6. Read the Installation instructions at each stage carefully and press Enter where needed.
7. You will have to accept the Trial License Agreement for installation to proceed further
8. Enter the full path to the directory of the installation (created in step #2). This will be referred as COGNIYUG_ROOT here after.
9. When you are prompted to choose the components of Cogniyug to be installed, you MUST enter option 3 as you have to install all the components on a single server during the Trial installation.
10. Press Enter and provide necessary information at appropriate places and let the installation finish.

Installation starts all the required processes of Cogniyug. (Please refer to Administration Guide to know more details of each components of Cogniyug and the associated processes.)

**How to Stop Cogniyug?**

The stop.sh script in COGNIYUG_ROOT directory can be used to stop all the related processes started by Cogniyug. Optionally, you may kill a Cogniyug process using kill command **without** -9 option.

**How to remove all the data and restart all over again?**

The reset.sh script in COGNIYUG_ROOT directory can be used to stop Cogniyug and clean all the data.

**How to monitor Cogniyug processes?**

The top.sh script in COGNIYUG_ROOT directory can be used to monitor all the related processes started by Cogniyug.

## Using Cogniyug

### Login

After you start all the processes of Cogniyug, you are ready to login to Cogniyug through its web interface. Open a browser (Note: Cogniyug officially support Firefox Mozilla and Google Chrome latest versions. We have not tested it on Internet Explorer yet; however we do not expect any issues with it. If you face any issues with IE, please report to us on support@techlineage.com ) and connect the following URL:

http://ip_address_of_linux_server_selected_during_installation:8000

Log in with Administrative user '**admin**' and default password '**admin**'

If you see the following window, you can assume that Cogniyug is configured properly.


The next step is to import some data into the system and see what operations you can perform on it.

### Cogniyug Data Format

Before we explain how to import your data into the system, it will be helpful to understand a bit about the Cogniyug data format. Cogniyug data format is very simple; as is as shown below.

*epoch_time_stamp: < tag1:value1 tag2:value2> actual message string*

Here

1. "epoch_time_stamp" is the standard UNIX time stamp format which is the number of seconds elapsed since 00:00:00 1st January 1970. You may read further about EPOCH time on http://en.wikipedia.org/wiki/Unix_time

   If you are not familiar with EPOCH time stamps, don't worry as the document will have sufficient examples on 'how to write your own Hooks' for importing your own data.

2. <tag1:value1 tag2:value2...$tag_n$ : $value_n$ > is nothing but metadata that provides more information about the actual message string. For example, you may add <host:*hostname* file:*full_path_of_file*> as metadata explaining the origin of the actual message string. Here 'host' is a 'tag' and 'hostname' is the 'value', file

is a 'tag' and 'full_path_of_file' is the 'value'. You may add as many tags as you want for each message. Another example of metadata could be <db_admin:*name_of_the_admin* Mobile_ Phone:*00000*> where you add name of the DBA and his/her phone number to enrich the actual message. The only restriction is that the tag and value should be separated by a colon (':') character AND the tag:value pairs should be separated by a whitespace (' ') character. There is no restriction on number of tag:value pairs. All tag:value pairs must be enclosed in angular braces <> as show above. Tags are extremely useful as they provide more information about the data. We will talk more about the tags and how to use them during a 'search' operation in the subsequent sections.

3. 'Actual message string' is nothing but the 'message' which needs to be preserved as it is, without the timestamp.

Example: Converting a raw message from syslog to Cogniyug format

Here is a message from "/var/log/messages" file:-
*Sep 30 2013 15:14:14 localhost ido2db: Error: database connection failed, forced client disconnect.*

Here, "Sep 30 2013 15:14:14" is the timestamp of the message. The corresponding EPOCH time will be '1380534254'. (Do not worry how we arrived at this number. Every scripting/programming language provides a bunch of APIs/commands to convert human readable date-time into EPOCH time. We will explain this further in some of the sample scripts that we ship with the product)
Assume that the message was generated in the server techlineage-lnx-01.techlineage.com. We can format the message as follows to conform to the Cogniyug format by adding the tags such as host and file.
1380534254 : <host:techlineage-lnx-01.techlineage.com file:/var/log/messages> localhost ido2db: Error: database connection failed, forced client disconnect..

If you carefully observe, we have split the original message into two parts,

1. The time stamp part (T)
2. The actual message string (S)

In addition, we added some metadata (tag:value pairs) to it so that we have some more information about the message. (Metadata or tags are completely optional but extremely useful. You may leave them blank but then you must supply an empty angular pair of braces <> to conform to Cogniyug data format)

We encourage you to write your scripts in any programming /scripting language that you are comfortable with to convert your data in the format explained here. Then you may easily pump in the data into Cogniyug and start analyzing it. Cogniyug ships some sample scripts (shell and python) that will help you to understand how to write your own scripts (btw, in Cogniyug terminology the fancy name for such scripts is 'Hooks'. We'll talk more about it in the section on 'How to write your own Hooks' for importing your own data)

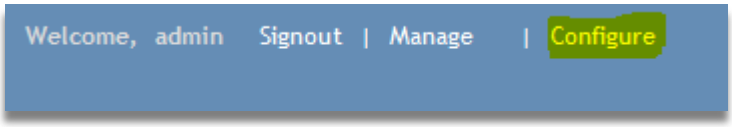
[If you face any difficulty in writing the Hook scripts for converting the data, please feel free to write back to us asking for help on support@techlinege.com. Do send sample data that you want to import so that we can help you in writing the Hook scripts]

### Importing the Data

After understanding the Cogniyug data format, we can go ahead and start importing the data into Cogniyug. There are two methods to import the data into Cogniyug:

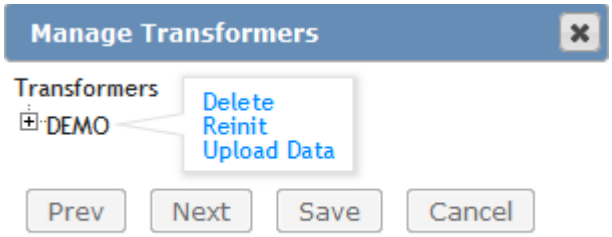
1. Uploading a file from client side (explained in 'Quick Start Guide')
2. Import Data using Hooks

#1 is definitely the simplest and the quickest method for importing the data into Cogniyug. We typically use this handy method to import a small amount of data quickly as explained in the Quick Start guide.
#2 is the most widely used method in production. It gives capability to continuously pump large amount of data into Cogniyug. We will explain this method (#2) here as #1 is already explained in the Quick Start Guide.
After you login, click on 'Configure' in the right hand corner of the 'Home' page.

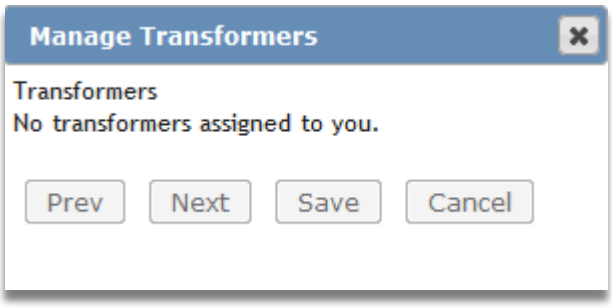



Select 'Transformer', a new window 'Manage Transformers' will open.

1. Cogniyug is shipped with a pre-packaged Transformer, named 'DEMO'. The purpose is to help you get started quickly. (Hope, you have read the Quick Start Guide and played around a bit with Cogniyug by now.) Since we want to dive a bit deeper now, let us delete the transformer by clicking 'Delete' option available on 'Demo' Transformer



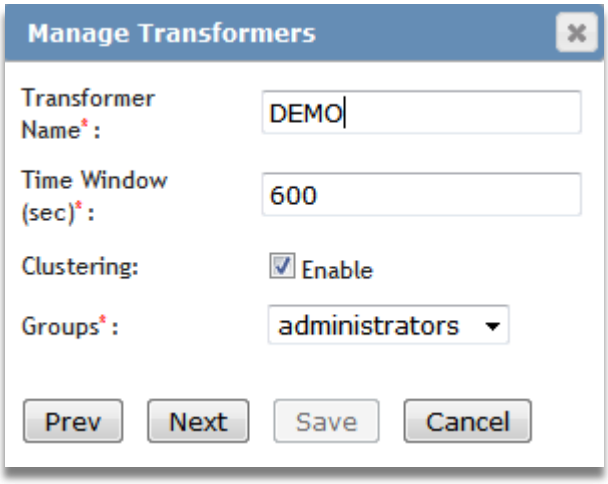

2. The 'Manage Transformer' dialogue will show that 'No Transformer assigned' currenly




Since we have cleared everything, you don't see any Transformers assigned to you. This simply means that you will have to create one yourself. (In the evaluation version of Cogniyug, you can create only one Transformer)

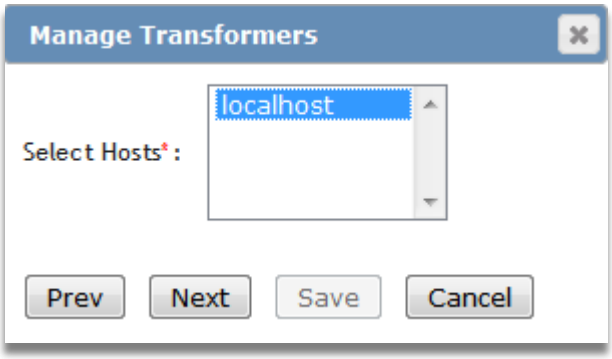
- Hover on the 'Transformers' using your mouse pointer – this will show pop up an 'Add Transformer' link - click on 'Add Transformer'. A small window will open as shown below.
- Give a logical name to the Transformer by entering an alpha-numeric string without white spaces in the 'Transformer Name' filed (say DEMO)

- Enter a numeric value in seconds in 'Time Window' field.
- Leave all the other options as is and click next.




Important Note: The 'Time Window' parameter entered here is an extremely important parameter. Cogniyug uses this 'Time Window' (in seconds) to correlate the events / log messages together while doing the Casual / Effect and Predictive Analytics. We will explain more about the importance of 'Time Window' when we talk about Causal / Effect and Predictive analytics. We will also explain how to override the 'Time Window' parameter defined at the Transformer level when we talk about 'Advanced Actions on Events'. At the moment, you can simply understand that we have defined a 'Time Window' of 600 seconds (or 10 minutes) which means that the events/log messages that are near to each other by 600 seconds will be considered for correlation analysis i.e. Causal / Effect and Predictive Analysis. Essentially, 'Time Window' provides a way to enforce *time proximity* for analysis.

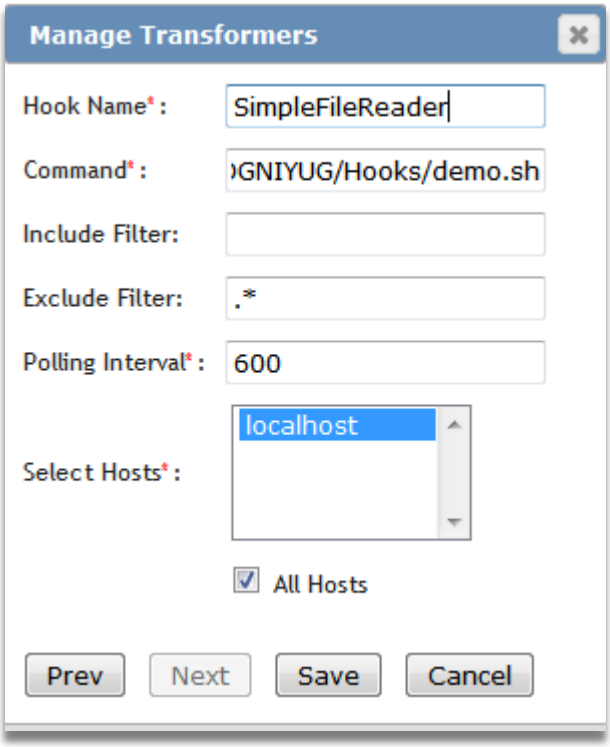
When you click 'Next', a window for selecting hosts will appear as shown below.



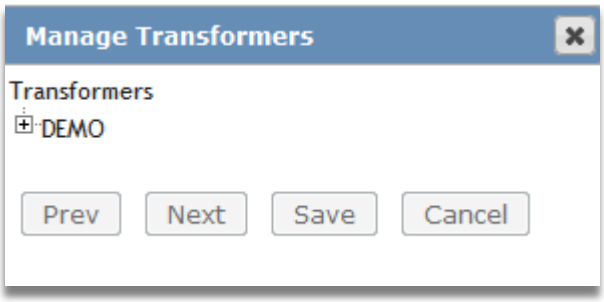

This is place where you add multiple hosts under one Transformer. To keep the demonstration simple, we just add one single host (localhost) here. However, in most practical cases, a Transformer is used to group together multiple hosts that are logically related. In this example we show you a very basic configuration of the Transformer and hence we limit ourselves to a single host here. We'll cover 'Grouping together multiple hosts under a single Transformer' in the 'Administration Guide'

Note: A Transformer is a logical grouping of related entities. In the 'online shopping portal' example, we may want to group together all the servers (web server, application server, database server), network equipments (appropriate load balancer, routers etc) and SAN devices together. Each of these entities will have their independent sources of log data. For example, web server will have syslogs, and web access/error logs, database server may have alert message logs and networking equipments may have SNMP traps as the source of data.

At the moment, just select 'localhost' and click the 'next' button. This will open up the 'Hook's' configuration window, a place where you configure the data collection script/program that reads the raw data from the data source, converts it in the Cogniyug format and stores it inside Cogniyug for further analysis. The script is called as a Hook which repeats its execution after a predefined time is elapsed ('Polling Interval')




- 'Hook Name' is the logical name of the Hook. Give any alphanumeric name without white spaces to your Hook (say SimpleFileReader).
- 'Command' is the full path to the Hook script. Remember, the script/program must be present on the specified path with appropriate permission on the specified host (that we will select in the last step)
- Leave 'Include Filter' blank.
- Enter '.*' (a regular expression "dot and a star") in the 'Exclude Filter' field. [We will explain more about 'Include and Exclude filters' in subsequent sections.]
- Specify a 'Polling Interval 'in seconds. The script will execute itself after the specified polling interval. In the above example, the script 'demo.sh' (Note: You will have provide a correct path to demo.sh. The script is installed under the 'Hooks' directory under COGNIYUG_ROOT) will execute every time after 600 seconds. The script will basically check for new data each time and convert it in the Cogniyug format. (We will explain what demo.sh exactly does latter on so that you can write your own scripts)
- Click 'Save. This will show up the Transformer "DEMO" that you have configured just now.




Congratulations! You have successfully configured your first data collector into Cogniyug using the Hook method.

IMPORTANT STEPS:

We hope we have already referred to 'Quick Start Guide' and played around Cogniyug by importing the data using 'Upload File' option. Let us reset the system to remove all the previously imported data and restart all over again.

Login to the Linux Server and go to COGNIYUG_ROOT. (cd *to_directory_where_cogniyug_is_isnatlled)*

cd COGNIYUG and run ./reset.sh script. This will clean-up everything imported so far. In #1 above, we have also deleted the Transformer 'Demo', recreated it and defined a new Hook on it. The reset.sh script will not alter the new definition of the Transformer, Hooks etc. After reset.sh is finished, run './init.sh' script to start all the Cogniyug processes again.

Log in to Cogniyug web UI again using the admin/admin credentials

**Exclude Filter:**

As the name suggests, Exclude Filter field basically acts a filter that 'excludes' messages which match the specified criteria from being sent to the pattern mining grid. By default, the value of Exclude Filter is NULL or empty. This means, nothing is excluded OR everything is sent to the pattern mining grid. This default setting makes all the messages to be readily available for causal and effect analysis. (NOTE: This is dangerous thing to do unless data is very small). In the 'Quick Start Guide' we imported the data using the 'Upload File' option. We had clicked the check box 'Send data to pattern mining grid' as we wanted you to get started quickly. However, it is to be noted that sending all the data to the pattern mining grid is not a good idea. This setting causes every message to be passed to the pattern mining grid and the causes/effects/pinpoint options are made readily available against every event in the Search Results window. (We demonstrated the same in the Quick Start Guide). Again, it is to be noted that it is NOT a good practice to send everything to the pattern mining grid. It not only puts serious pressure on the available resource of the pattern mining grid (memory and cpu, in particular) but also results into a lot of unwanted patterns as data grows. After all, not all the messages are worth considering for patterns every time.

In this example, we have deliberately entered a value of '.*' which is a recommended way of configuring Cogniyug. The '.*' value is a regular expression covering EVERYTHING. It simply configures Cogniyug to exclude every message from passing to the pattern mining grid. You may consider entering multiple regular expressions (separated by comma) here which will tell Cogniyug to not send such messages to the pattern mining grid that match the input regular expression.

The obvious question that you may have at this point is 'How to perform causal/effect analysis as we explained in the Getting Started guide after excluding everything from pattern mining?' The answer is to use 'Advanced' option to perform analysis on the events. Use of 'Advanced' option in the search results is the recommended way of using Cogniyug while performing 'causal/effect' analysis. We have explained this with appropriate examples in the subsequent sections.

**Include Filter:**
Include filter is exactly opposite of the Exclude Filter. It takes one or more regular expressions as input, evaluates every incoming message against the regular expressions and passes only those messages further to the pattern mining grid matching the regular expression. Include Filter is evaluated first followed by Exclude Filter. Default value of the Include Filter is NULL or Empty which causes everything to pass through.

**Summary:**
- ✓ Deleted the defalt definition of the DEMO Transformer
- ✓ We redefined a single Transformer (DEMO)
- ✓ Added a host (localhost) under it
- ✓ Assigned Hooks ($COGNIYUG_ROOT/Hooks/demo.sh) to it.
- ✓ Excluded everything from passing to the pattern mining grid (by using '.*' regular expression in the Exclude Filter)
- ✓ In most practical scenarios you will add multiple hosts under a single Transformer and each host may have one or more Hook scripts assigned to it. You may configure multiple Transformers in this fashion that group together the logically related entities (This capability is not available in the Evaluation/Trial version)
- ✓ Ran reset.sh and initi.sh scripts to so that the new definition of the Transformer is effective

## The Search

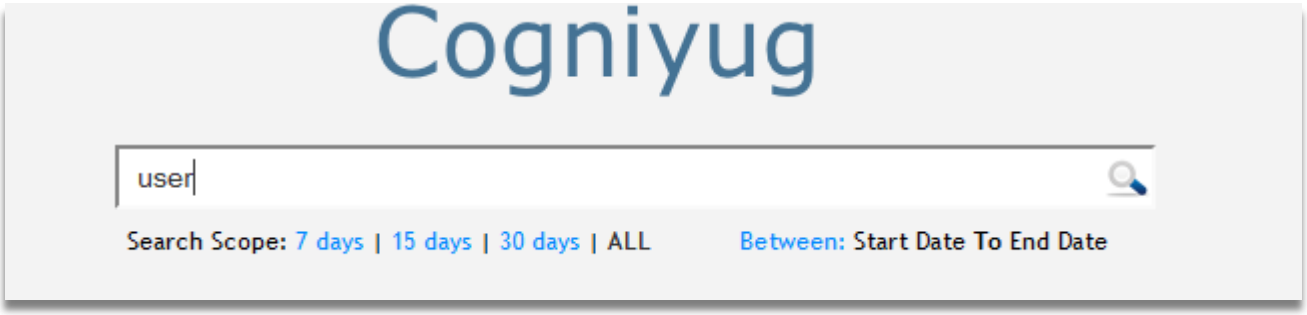
After you have configured your first Transformer successfully, the next step is look for your data in Cogniyug and then perform various operations on the data.

Cogniyug provides a fantastic search on top your data. The search is the entry point for any operation that you may want to do on your data. When you login to Cogniyug, the home page presents a search window where you can enter what you are looking for. You may enter a single word, multiple words, form complex expressions using AND, OR and NOT operators and much more. The best way to explain various capabilities is by giving examples.

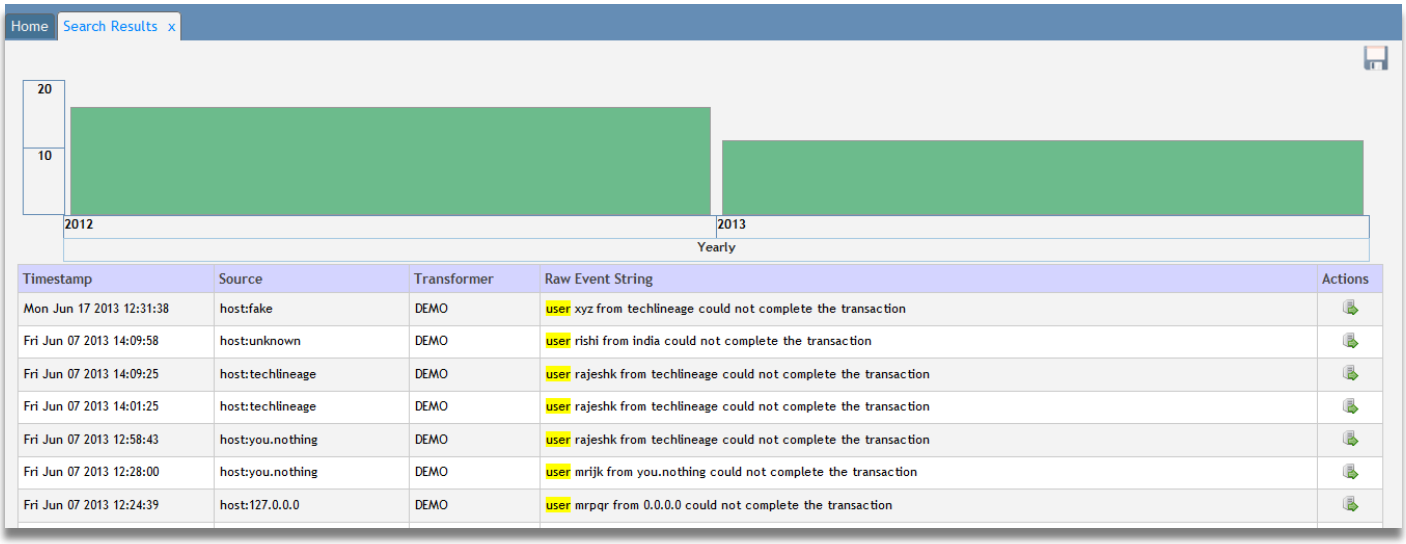
We assume that you have configured the demo.sh script as the Hook – the demo.sh reads the pre-packaged sample data into Cogniyug. In the examples below we will use the pre-packaged data shipped with Cogniyug. For convenience, you may open the demo.txt file for inspection so that you can enter various strings and search for them in Cogniyug.

### Searching for a single word, understanding the output and working with the search results

Enter a single word 'user' in the search window as shown below and hit enter




It will open up a new tab with 'search results' as shown below



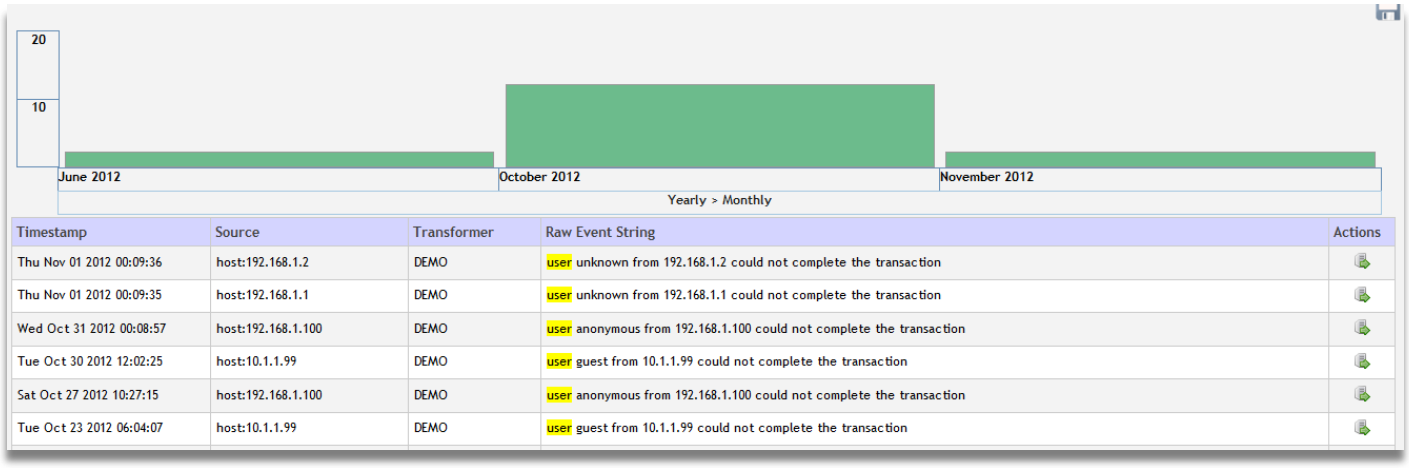

| Timestamp | Source | Transformer | Raw Event String | Actions |
|---|---|---|---|---|
| Mon Jun 17 2013 12:31:38 | host:fake | DEMO | user xyz from techlineage could not complete the transaction | |
| Fri Jun 07 2013 14:09:58 | host:unknown | DEMO | user rishi from india could not complete the transaction | |
| Fri Jun 07 2013 14:09:25 | host:techlineage | DEMO | user rajeshk from techlineage could not complete the transaction | |
| Fri Jun 07 2013 14:01:25 | host:techlineage | DEMO | user rajeshk from techlineage could not complete the transaction | |
| Fri Jun 07 2013 12:58:43 | host:you.nothing | DEMO | user rajeshk from techlineage could not complete the transaction | |
| Fri Jun 07 2013 12:28:00 | host:you.nothing | DEMO | user mrijk from you.nothing could not complete the transaction | |
| Fri Jun 07 2013 12:24:39 | host:127.0.0.0 | DEMO | user mrpqr from 0.0.0.0 could not complete the transaction | |

The 'Search Results' page is mainly divided into two halves, the top portion or the first half is the 'time-density' graph and the bottom portion or the second half shows the actual search results in the paginated format.

**X-axis (Time Density Graph):** The top half of the 'Search Results' page is a time-density bar graph with 'time' on X-axis and count of the events matching the search criteria on Y-axis. The X-axis summarizes the results and shows the biggest possible time-span to begin with. In the above example you may see that the X-axis shows the 'yearly' summary because the data matching the search criteria is spanned across two years. Had that been less than a year, Cogniyug would have shown *Monthly* summary.
**Y-Axis (Count):** Y-axis shows the number of events matching the search criteria.

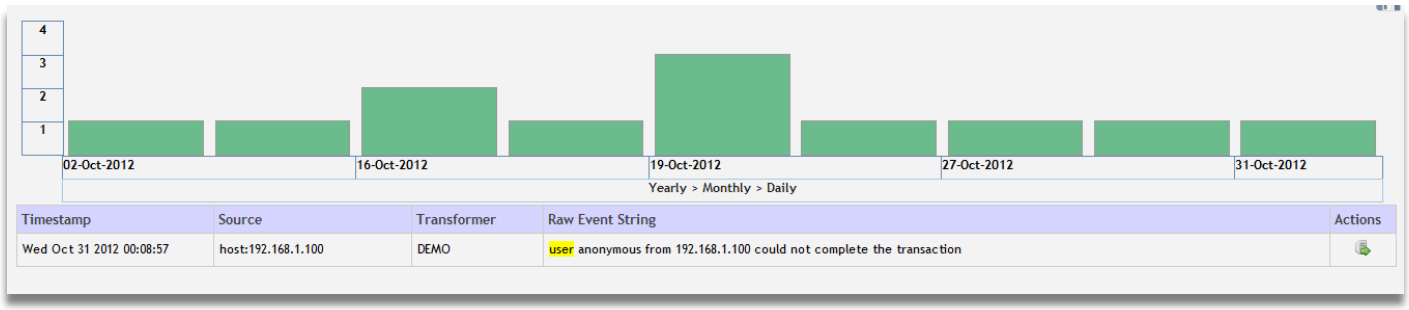
The second half of the 'Search Results' shows the actual strings in paginated manner with 10 strings on each page. It always shows the latest results at the top. In the above example, the messages of June 2013 are shown first simply because they are the latest results with the given search criteria ('user')
You may single click on any portion of the time density graph and top ten events of the selected time range will be shown in the bottom half.

You may also double click on the time density graph to drill deep into the time axis. For example, if we were to search for the events matching our search criteria on 19th Oct 2012, we simply double click on 'year 2012' portion of the above graph. This will redraw the graph and monthly summary of year 2012 will be drawn in the time-density graph. The bottom half of the search results will show 'latest 10' events from year 2012.




| Timestamp | Source | Transformer | Raw Event String | Actions |
|---|---|---|---|---|
| Thu Nov 01 2012 00:09:36 | host:192.168.1.2 | DEMO | user unknown from 192.168.1.2 could not complete the transaction | |
| Thu Nov 01 2012 00:09:35 | host:192.168.1.1 | DEMO | user unknown from 192.168.1.1 could not complete the transaction | |
| Wed Oct 31 2012 00:08:57 | host:192.168.1.100 | DEMO | user anonymous from 192.168.1.100 could not complete the transaction | |
| Tue Oct 30 2012 12:02:25 | host:10.1.1.99 | DEMO | user guest from 10.1.1.99 could not complete the transaction | |
| Sat Oct 27 2012 10:27:15 | host:192.168.1.100 | DEMO | user anonymous from 192.168.1.100 could not complete the transaction | |
| Tue Oct 23 2012 06:04:07 | host:10.1.1.99 | DEMO | user guest from 10.1.1.99 could not complete the transaction | |

Observe that we now have 'Monthly' summary on X-axis. To reach 19th Oct 2012, we should simply double click 'October 2012' portion on the graph and Cogniyug will drill into the month of October 2012 to show the daily summary as shown below.




| Timestamp | Source | Transformer | Raw Event String | Actions |
|---|---|---|---|---|
| Wed Oct 31 2012 00:08:57 | host:192.168.1.100 | DEMO | user anonymous from 192.168.1.100 could not complete the transaction | |

You may drill down deep by double-clicking on the interested portion of the graph (till second wise-summary) and locate the exact instance of the event/log message that you are interest in.
If you want to come back to 'Monthly' / 'Yearly' summary, you may simply click on particular caption on the X-axis and the graph will be redrawn to the chosen summary. The bottom half of the 'Search Results' with actual message details will be refreshed with every click and will always show the latest 10 messages at the selected/clicked time.

The bottom half of the 'Search Results' has five columns.

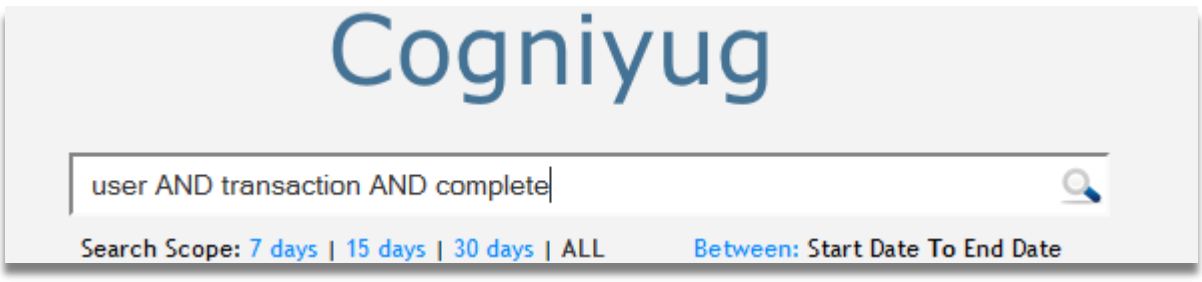
| Column Name | Explanation |
|---|---|
| Timestamp | It is the human readable timestamp of the exact occurrence of the event |
| Source | This column shows all the 'tags' that we may have added as metadata while importing the data using the Hook script. (Always in lower case) |
| Transformer | This is the name of the Transformer (DEMO in our case) |
| Raw Event String | This is the event string (except time stamp field) as it appears in the original log message. (Always in the lower case. Note that Cogniyug converts all the messages into lower case) |
| Actions | This icon shown in the last column allows various Cogniyug actions (casual analysis, effect analysis) to be taken when clicked. We will explain more about it in the subsequent sections. |

### Using the Search Grammar for constructing complex searches

In the last example we explained how to search a single word, how to interact with summary results and drill down etc. In this example, we introduce you to the most practical use of the Search Capability provided by Cogniyug i.e. 'Cogniyug search Grammar'. A single word search will not serve the purpose in most practical scenarios. The flip side of using a single word search is that you may be overwhelmed with results as the word may match a large number of strings in the Cogniyug database. This may adversely impact the response of the search engine as the large search results would consume more computing resources, mainly Memory and CPU. To get the best performance, we highly recommend you to form the exact search to the best of your ability using the 'Cogniyug Grammar' constructs explained below.

Cogniyug Grammar is very simple yet very powerful. All you have to know is three key words AND, OR and NOT in the upper cases and the curve braces ['(' and ')']. Using these simple English words you will be able construct complex search queries to match the exact results that you are looking for.

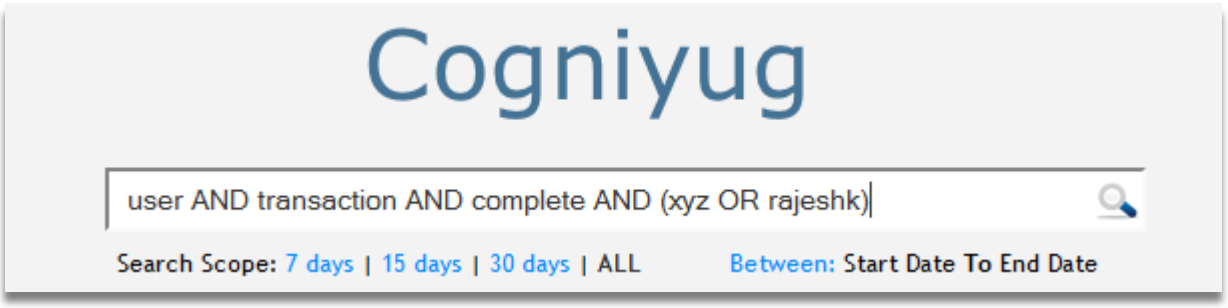
For example, consider that we have to search only those messages where a 'user' was not able to 'complete' the 'transaction'. Here, we are not interested in any other user activity. The search string can be formed like shown below



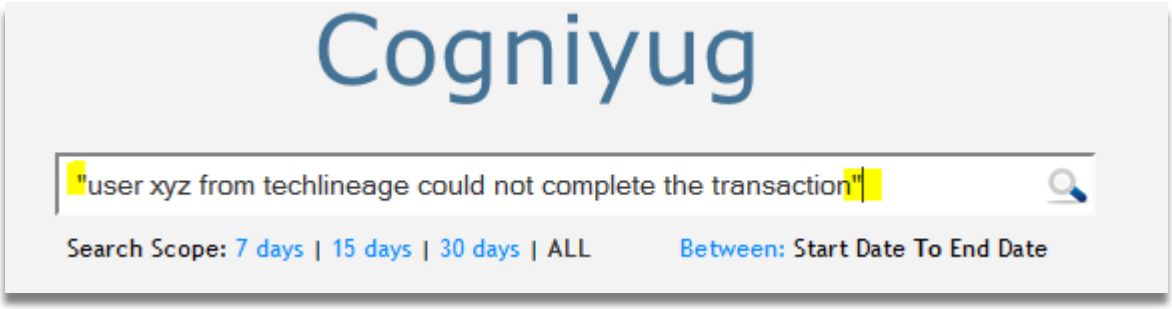
When you hit enter, the search will return only those records/events/log messages where the 'user was not able to complete the transaction'. The interpretation of output and the actions on the output remain exactly the same as explained in the previous section, with the only difference being the precise filtering.

Note that Cogniyug search treats upper case key words AND, OR and NOT differently. You MUST enter these grammar words in the upper case if you want them to be treated as Cogniyug Grammar words.

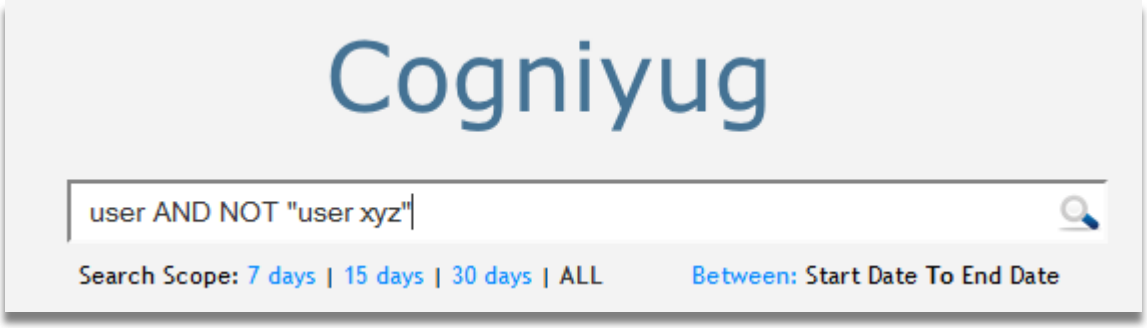
Another example of a complex search could be to get only those log messages where a user xyz or user rajeshk was not able to complete the transaction. The search string would look something like this




If you want to search a string with more than one word and you are looking for a precise match, you'll have to enclose the string in double quotes. For example if you want to search for a precise string: "user xyz from techlineage could not complete the transaction", the search string will look like as shown below: Notice the double quotes.

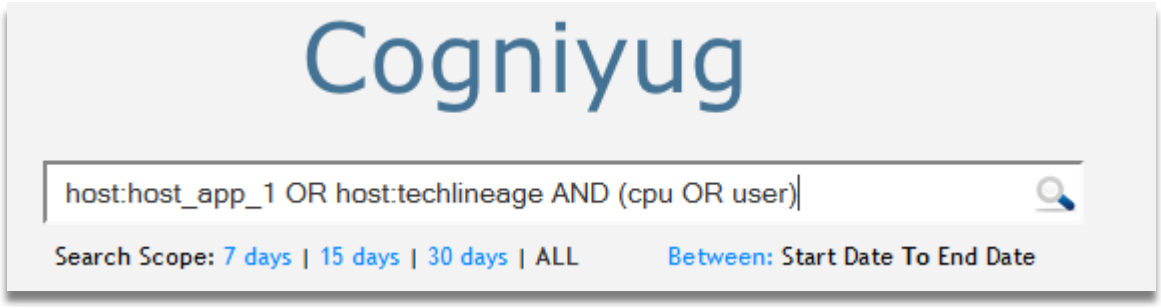



If you want to search for all the records of the 'user' except for 'user xyz', you will have to input the string like shown below



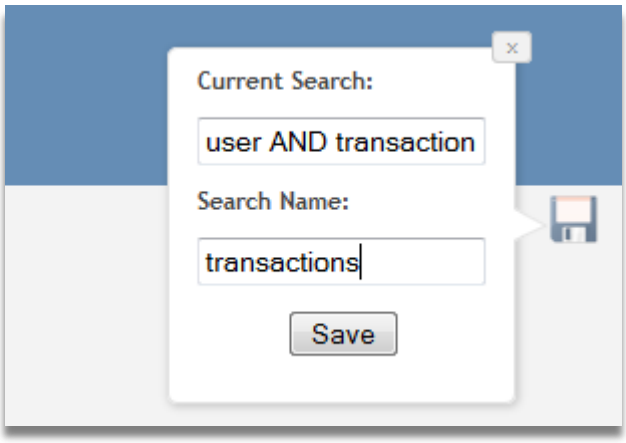
**Search using tags**

You can use the tags efficiently to quickly create good search criteria. Tags and associated values must be entered in lower case with a colon separating them (e.g. tag:value). In the example below we show how to use tags and string together to get specific results.




## Saving the search

After constructing a complex search, you may want to save it so that you can reuse the same search in future. You may save the search by clicking on the 'Save' icon on the top right hand corner of the 'Search Results' tab page. Give a logical name to the search and click 'Save'.




To retrieve the saved search, click on 'Manage' (at the top right hand corner in the browser, next to Signout) and click "SavedSearchs".

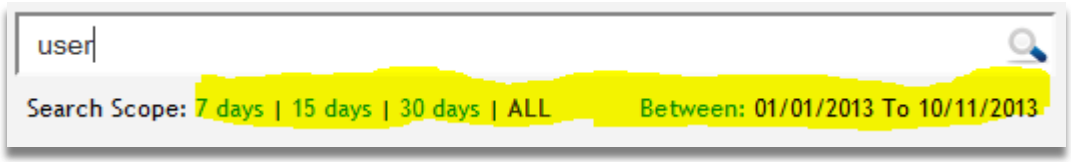

You will be able perform following actions by clicking appropriate icon in the 'Actions' column

- Re-run the search any time in future by clicking the icon
- Associate the search with an external report by clicking icon (We will talk more about the reporting engine in Administration Guide)
- Set the search period by clicking icon. This allows the user to set the search period to a specific date range or last 'n' days. The 'last n days' is an extremely useful option as it allows the user to see the latest data.

- Delete the saved search by clicking the delete icon 

### Managing the search scope

Search output increases as the amount of data increases in the Cogniyug repository. The response time of the search operation is proportional to the amount of data that you get in the output matching the given search criteria. In most practical cases, we are not interested in seeing all the data that matches our search input. In fact, most of the times we are interested in seeing only the latest results; say data over last *n* days. You can simply click on the 'Search Scope' options to limit the scope of your search. This dramatically improves the performance of the search engine and at the same time improves the productivity of the user as the output will be a lot more relevant when you limit the scope of the search.




Summary: Hope with these examples, you have a fair bit of idea about

- ✓ How to construct a search
- ✓ How to use Cogniyug Grammar
- ✓ How to interpret the search output
- ✓ How to save a search
- ✓ Hot to limit the scope of the search for getting optimal performance
- ✓ How to drill down to the exact event by navigating through the search results

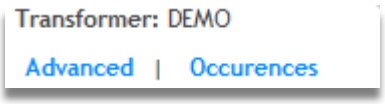
In the next section we will dive into the details of 'actions on the search results'

### Actions on Search Results

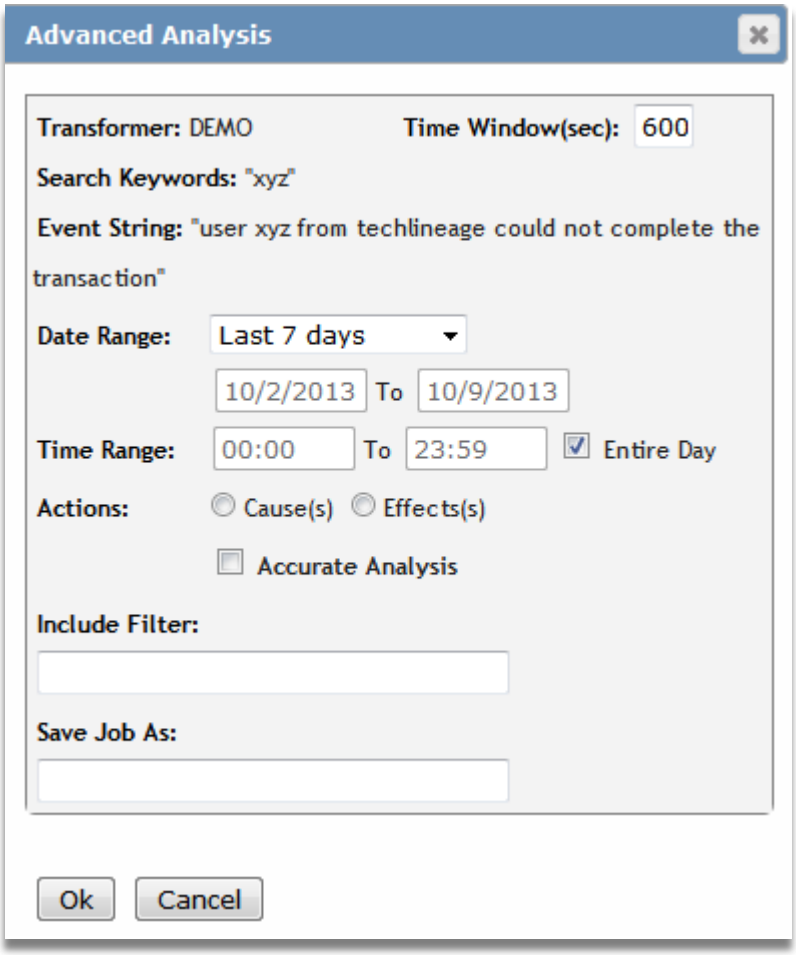
After performing the 'search' and locating the exact instance of the event/log message, you will be able to perform various actions on the selected event/log message.

### Causal Analysis

This is the most important and frequently used action performed by the users while dealing with the log messages in Cogniyug. There are few simple steps that you need to perform for initiating a 'Causal Analysis'. Click 'Advanced' option in the 'Actions' column shown below.




This is will open up the 'Advanced Analysis' window as shown below

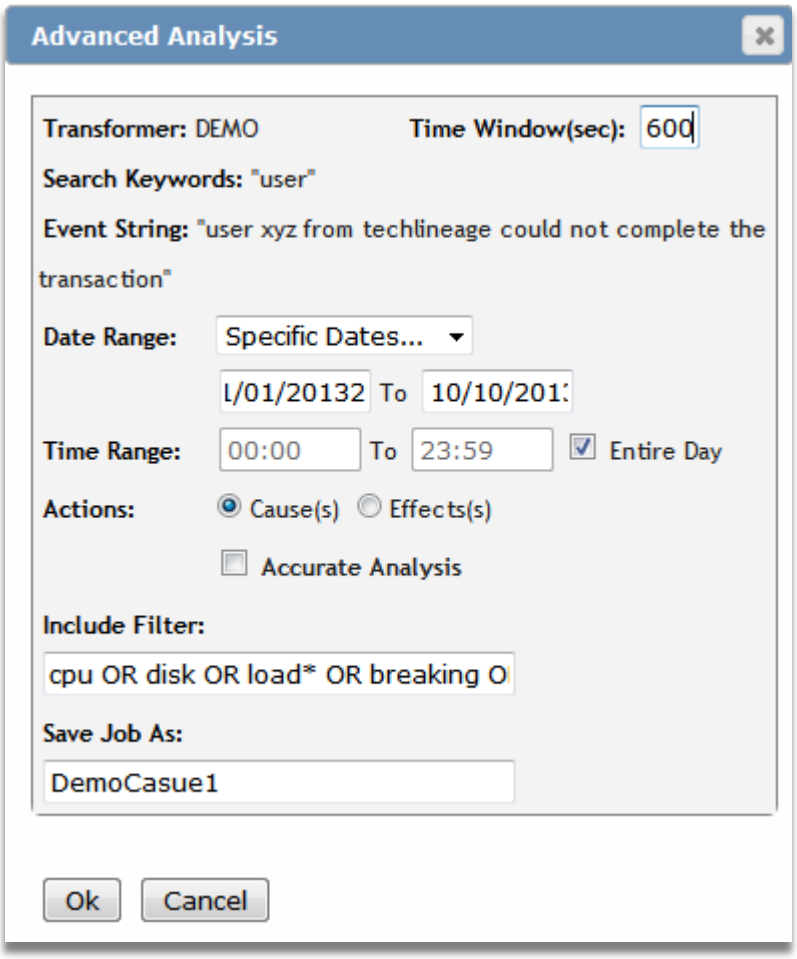



Here is the explanation of each field in the 'Advanced Analysis' window

- Transformer Name is shown as DEMO here because this was the Transformer we created in the first step. You will not be able to edit this.
- Time Window is shown as 600 seconds by default because this time windows was associated with the Transformer at the time of creation. You can override this by putting an appropriate value here. This value will be used to correlate the events that are close to the chosen event (i.e. 'Event String'). Here, if

value is 600 seconds, we will chose all such events that have their time stamps *less than* 600 seconds of all the occurrences of the chosen event *type*.

- Event String: Shows the '*type* of the event' on which you want to perform your causal analysis. Here the event string is shown as "user xyz from techlineage could not complete the transaction". This was the event we have chosen for performing the causal analysis. Cogniyug smartly identifies the "*type of the event"* at the time of importing the data into its repository. In this case, Cogniyug has identified the type of the event as "user * from * could not complete the transaction". (We will explain more about 'event types' in the subsequent sections)
- Date Range and Time Range: Here we select the time period between which we want to select the data for analysis. This allows us to have a fine control over the period of interest. From the 'Drop Down' box, you can easily select the date range OR select 'Specific Dates' to enter the dates of your choice. Similarly, you can select the time range for the specified days. By default, 'Entire Day' is selected. This means that all the data of the selected date ranges will be selected for analysis. You may uncheck the 'Entire Day' checkbox and enter the precise time period so that only the data between the selected time periods and between the selected dates will be considered for analysis.
- Click 'Causes' radio button
- In the include filter, specify the 'type of events' that you are interested in correlating with the selected event type (shown as 'Event String'). Here, you have to enter the *search criteria* exactly similar to that you used in the 'Search' section. All the Cogniyug Grammar rules apply here. In this example, let us enter a search string as "cpu OR disk OR load* OR breaking OR database"
  *TIP: You can use tags (e.g. host:host1 AND file:/var/log/messages) to specify your search criteria quickly. To make it work efficiently, you should combine multiple tag values by an AND operator. To make it more efficient, you should combine tags and strings using AND operator.*
  *Note: Try to be as specific as possible in the include filter by entering precise search criteria. This will reduce the search space and minimize the load on Cogniyug.*
- Enter a logical name of your choice to save this as job.
  Note: Cogniyug treats every 'Action' on the event/log message as a Job. It indentifies the job internally by the given Job Name.



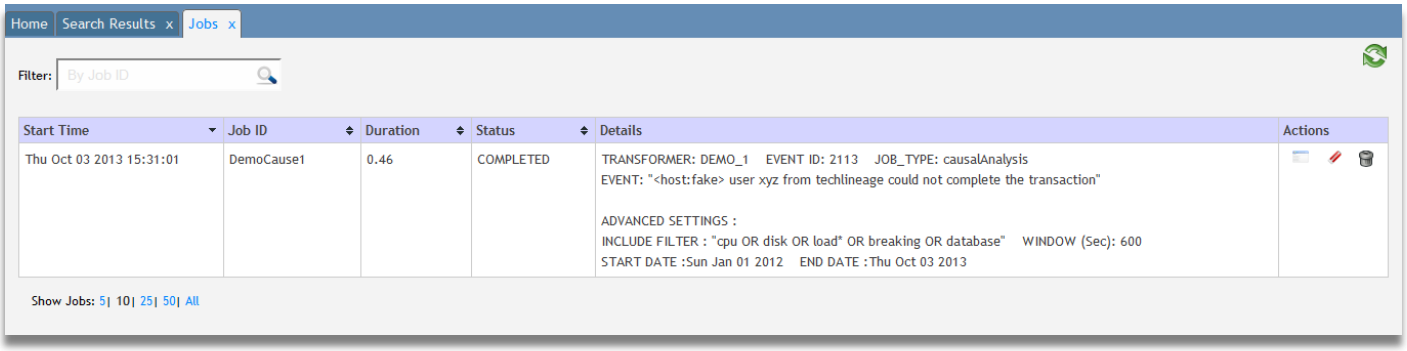

To sum-up, we can say that we have

- Chosen the exact instance of the event for causal analysis. Here we are interested in finding 'why' the particular *type of event* has happened based on the available data. We are interesting in knowing all possible causes for all such *event types.*
- We further selected the data for our analysis by entering the
    A. Time window (600 seconds)
    B. Date and Time range and
    C. Include Filter criteria (search string using Cogniyug Grammar)

  The above inputs viz. A, B and C together form a 'data selection criteria' to create a set of data that will be used for Causal Analysis. Cogniyug first selects all the *'event types'* of the selected event (Give by 'Event String'). It then applies the 'Include Filter' with 'Date/Time Range' filter to create a temporary 'dataset'. Further it applies the 'Time Window' filter (600 seconds in this example) to select only those events that are close enough to the selected event (*must have their time stamp values within 600 seconds from the selected event type).* This forms the final dataset to be used for Causal Analysis.

When you click 'Ok', you will be acknowledged with a dialog box indicating that the job 'DemoCause1' has been submitted. This means that the system has started analyzing the causes of the interested *event types* using the data selection criteria defined in the above example.

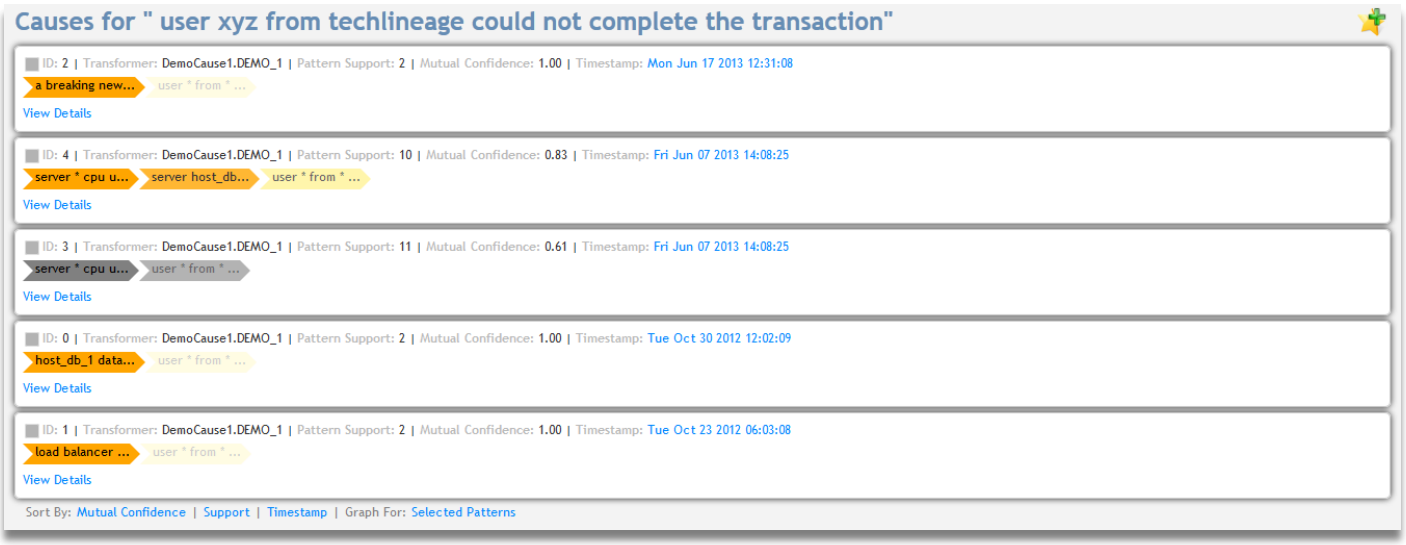
To check the status of the submitted job, you will have to click on 'Manage' -> 'Jobs' at the top left corner of the browser. This will open a new tab page with details of the jobs. Each user will be able to see and manage all the jobs submitted by him/her. As you can see in the screen shot below, the submitted job "DemoCause1" is seen with status 'Completed'




The jobs tab shows six columns as follows

1. Start Time: The time when job was initiated by the user. By default, the jobs are started with the latest job at the top.
2. Job ID: This is the logical name that you have given to the job. Sometime, system also gives its own unique names (This is case we explained in the Getting Started guide when you perform quick analysis)
3. Duration: This is the time taken by the job to complete in seconds.
4. Status: Shows appropriate status of the job. Possible values are COMPLETED, STARTED and FAILED
5. Details: Shows all the necessary details such the Transformer, Job Type, Event String, Include Filter, Time Window, Start Date, end Data etc.
6. Actions: Icon is the 'Details' icon, which when clicked shows the results of the job. Result of the job. The format and representation of the job results will vary for each job type. In our case, the job type is CausalAnalysis. Icon is the delete results icon. This deletes all the results of the job. The icon is complete delete which not only deletes the Job results but also deletes the associated data set definition.

We will click the 'Details' icon to check the results of the 'Causal Analysis' job with Job ID 'DemoCause1'. This will open a new tab page as show below

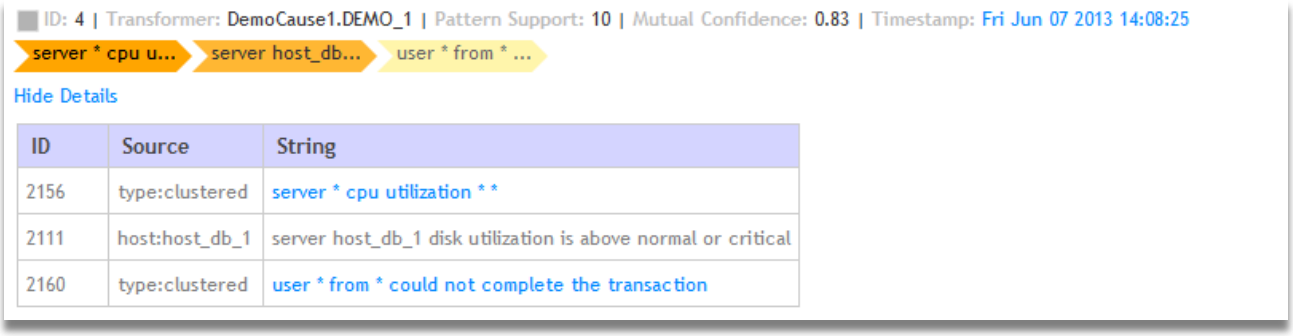


**Result of Causal Analysis**

You can see that the Causal Analysis algorithm has found five possible causes for the event *type* 'user xyz from techlineage could not complete the transaction'. The causes are nothing but patterns with different color coding. At a broad level, we can say that the Orange patterns are strong causes and Gray colored patterns are the weak ones. It is extremely important to understand this page thoroughly. Let us take a look at one pattern, say the second one and try to understand each field in it.


1. ID: This an internal field (referred as pattern ID)
2. Transformer: This is the name of the dataset on which we performed the analysis. It is formed by concatenating the Job ID with the name of the base transformer (DEMO) and an underscore 1(which actually is not important from user perspective)
3. Pattern Support: This is a numeric value indicating the exact number of times this pattern was observed.
4. Mutual confidence: This again is a numeric value between 0 and 1 indicating the confidence of the pattern. Higher value indicates stronger confidence. We will soon explain how the mutual confidence is calculated.
5. Timestamps: This is the list if the timestamps indicating exactly when the pattern has happen. Since the support of the pattern is 10, we will have exactly 10 timestamps here. If you click on the timestamp, you will be able to see the list of the timestamps. By default, we show the latest timestamp at the top in the timestamp list. For the sake of brevity, each timestamp in the list is the timestamp of the first event in the pattern (timestamp of event 'server * cpu utilization * *' in this example). The time difference between the first event and the last event of the pattern will not be greater than 600 seconds as we are performing the causal analysis over a' Time Window' of 600 seconds.

Click on 'View Details' to expand the pattern details.

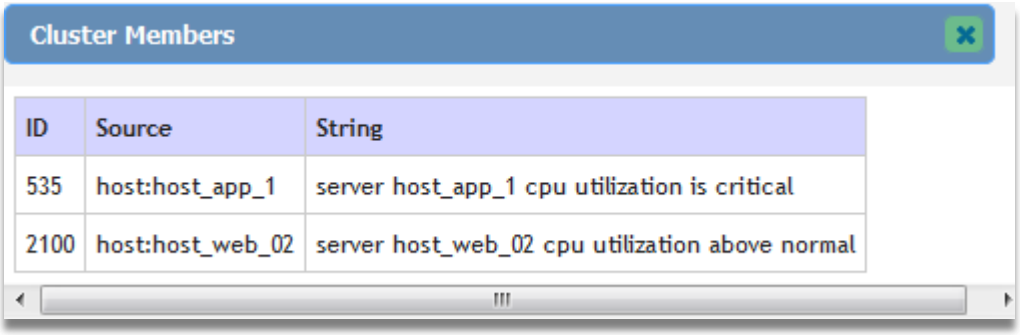



| ID | Source | String |
|---|---|---|
| 2156 | type:clustered | server * cpu utilization * * |
| 2111 | host:host_db_1 | server host_db_1 disk utilization is above normal or critical |
| 2160 | type:clustered | user * from * could not complete the transaction |

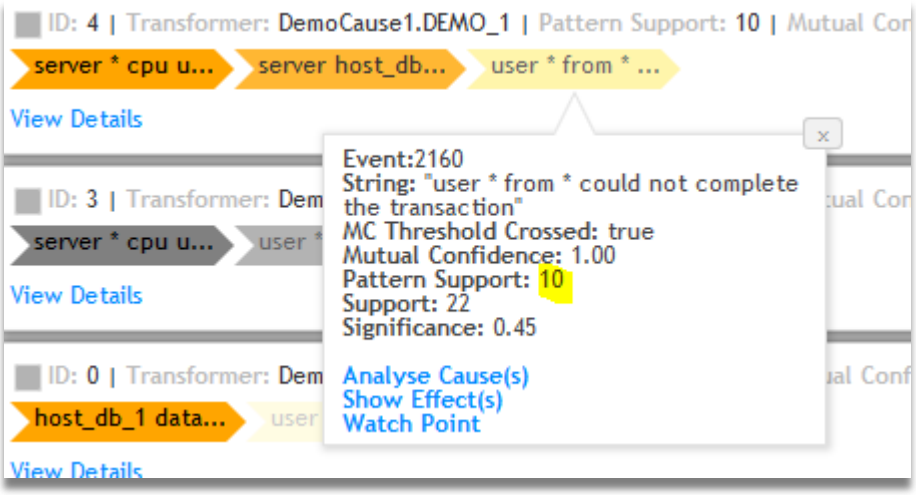
The first column, 'ID' is the internal ID of the event.

The second column is the source field. It will either show the 'tags' associated with the event/log message OR it will be marked as 'type:clustered'. Cogniyug adopts a smart method to categorize the *similar* looking events into clusters. In the above pattern, we see that there are two 'clustered' events. First is the "server * cpu utilization **" and second is the "user * from * could not complete the

transaction". Both these events are *generic* representations of the actual events happening in the system.
To examine the details of the clustered events, you simply have to click the event shown in the Details pane. When you click the event with id 2156 above, you will get the member of this cluster as shown below.

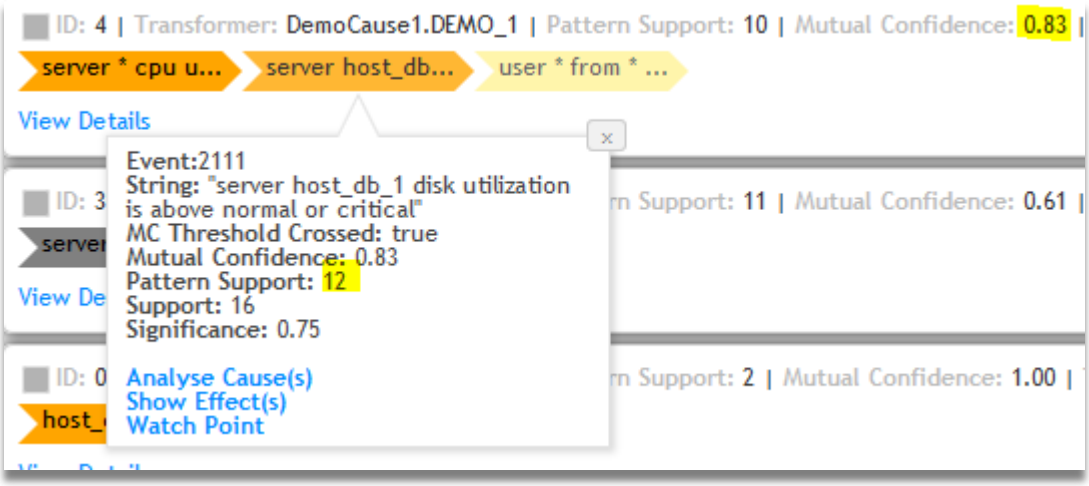



**Mutual Confidence Calculations:**
Mutual confidence is a measure that indicates how confident we are about our conclusion. Mathematically, it is the ratio of the support of the longest pattern to its immediate sub-pattern, till the ratio falls below the predefined mutual confidence threshold which is 0.7 in our case (We will explain how to change the confidence threshold in the Administration Guide)



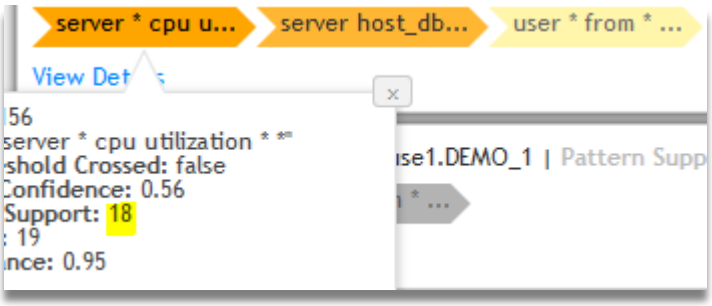

Let us start moving from right to left of the above pattern.
Click on the last node of the above pattern. It will show you various statistical details about the event "user * from * could not complete the transaction". The important thing to observe here is the 'Pattern Support' at this level in the pattern which is equal to 10 (exactly same as that of the 'Pattern Support shown at the top level). Let us call it S1 at the moment.

Now click on the 'second last node' which is "server host_db_1 disk utilization is above normal or critical".




This is the immediate sub-pattern and the pattern support at this level is 12. [Corollary: A sub-pattern will always have higher support than it super pattern.] Let us call it S2. The ratio of S1/S2 is 10/12 = 0.83 or 83%. Note that at this level we are above our predefined mutual confidence support threshold i.e. 0.83 > 0.70.

Now click on the first node in the pattern which is "server * cpu utilization * *"

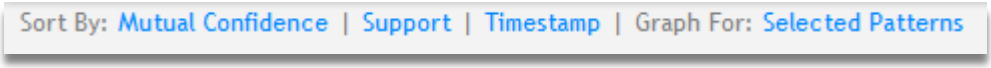



This is the last sub-pattern in the entire pattern. Let us call the pattern support at this level as S3 which is 18. The ratio of S1/S3 = 10/18 = 0.55 or 55%. At this level in the pattern we fall below the predefined threshold of mutual confidence as 0.55 < 0.7.
Based on this calculation, we can say that the probability of the final event (i.e. 'user * from * could not complete the transaction') happening is above 70% if and only if the first event (i.e 'server * cpu utilization * *") is followed by the second event (i.e. "server host_db_1 disk utilization is above normal or critical") within 600 seconds.
The probability of the final event (i.e. 'user * from * could not complete the transaction') happening is just 55% which is below our predefined confidence threshold i.e. 0.7 or 70%.
Hence, we mark our confidence at the second event (event-2) in the pattern which is 0.83 or 83%. We mark event-1and event-2 in orange to indicate that the pattern formed by these events is the strong casual pattern for event 3 to happen. In Cogniyug terminology, we call pattern formed by event-1 and event-2 as the '*minimum necessary condition pattern*' for event-3 to happen (with at least 70% confidence)

We encourage you at this time to take a look at some other patterns and try to calculate the mutual confidence yourself by hand. See if it matches the reported mutual confidence at the pattern level.

### Sorting on various fields

At the bottom of each page in the Job Results tab, you will see some fields that you may use to sort the result as per your convenience. By default, the results are sorted with timestamp fields showing the latest results first. You may click on any of the 'Sort By' options and view the results in appropriate order. To invert the sorting order, you just need to click the same 'Sort By' option again.

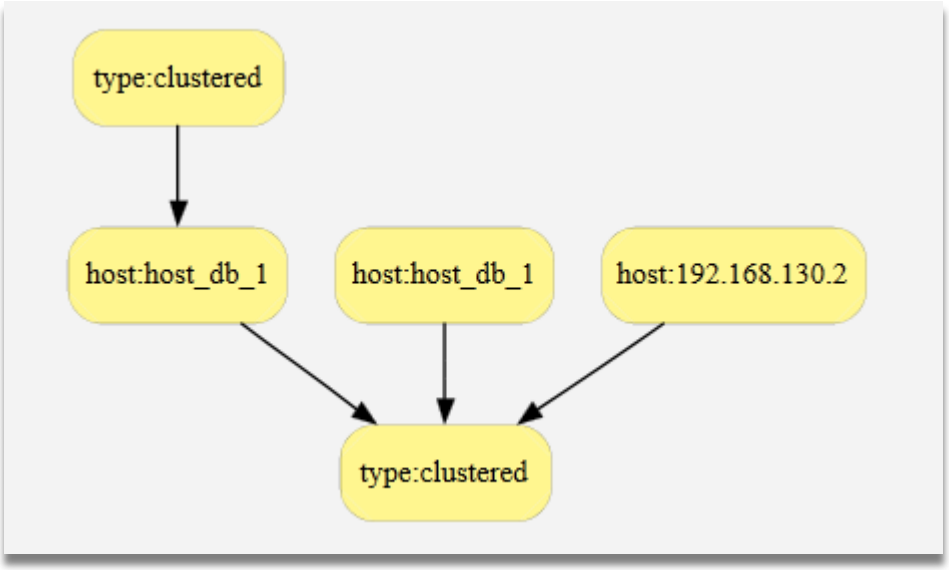



- Sorting by 'Mutual Confidence' helps in getting the strongest conclusion first.
- Sorting by 'Support' shows the patterns that have appeared maximum number of times at the top.
- Sorting by 'Timestamp' shows the latest results first.

### Graphical Representation

Observe the "Graph For: Selected Patterns" link at the bottom of every page. You can use this option to visualize the resultant patterns graphically. You may select/deselect the patterns on the current page by clicking the small grey square marked by red circle in the screen shot below. By default, the square is marked grey indicating the pattern is selected for graphical representation. You may click on the checkbox to un-select the pattern.

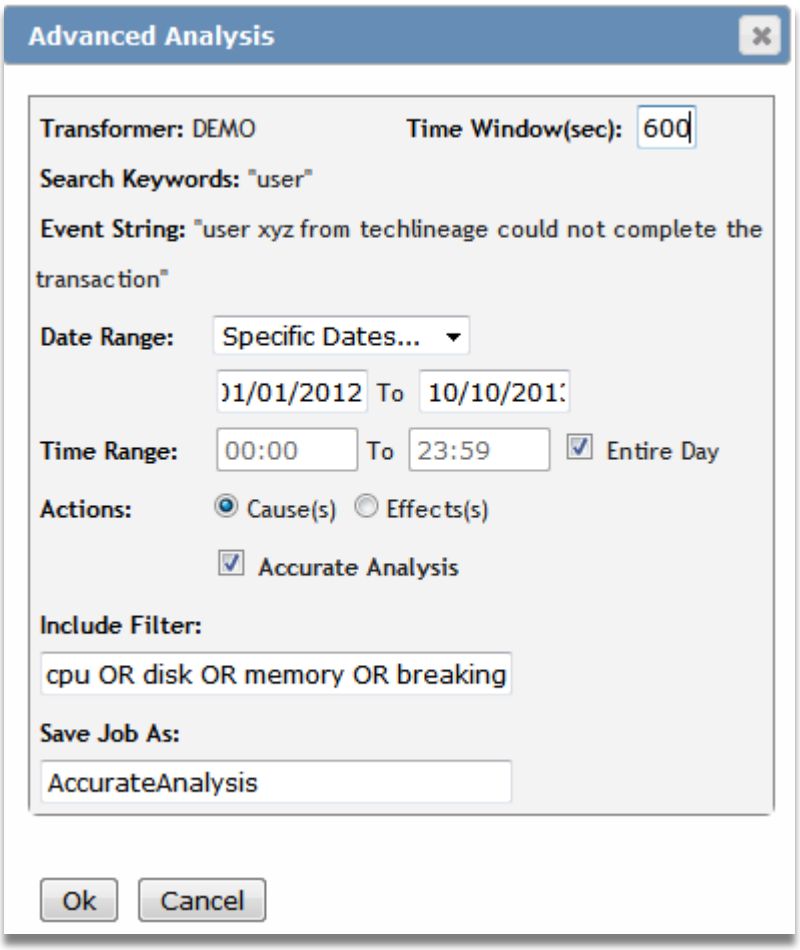

After selecting the appropriate patterns, you can click the 'Selected Patterns' option at the bottom of the page. This will draw a graph of the patterns as shown below. When you move your mouse over a node in the graph, you will be able to see the details of the node.




### Accurate Analysis

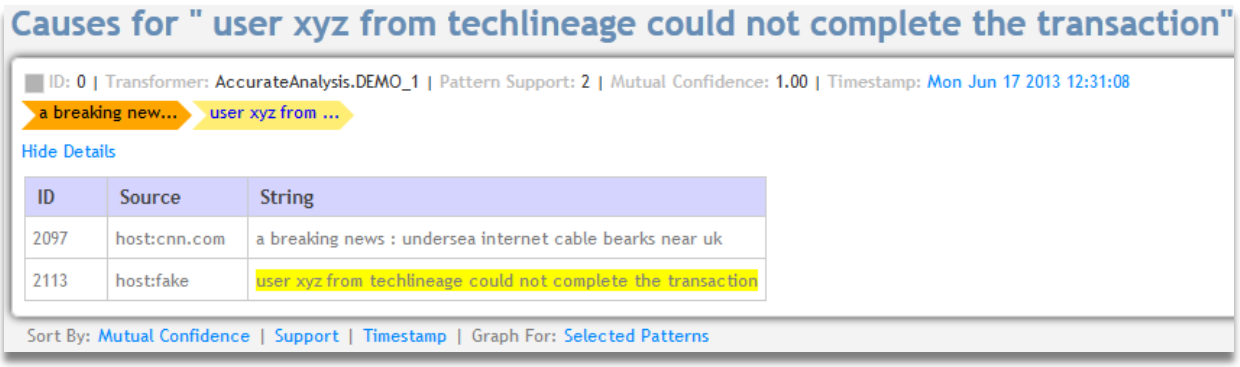
The above example showed us *all* the causes for all incomplete transactions for all the users (over the data selection criteria). This generic or wider level of analysis provides a lot of useful insight. However, in some cases you may be interested in knowing the cause of a particular instance of a particular event.

Most of the steps in the previous example remain exactly the same. Click the 'Advanced' option for event 'user xyz from techlineage could not complete the transaction'. Enter exactly the same information as entered in the previous example. Check the 'Accurate Analysis' check box after selecting the 'Causes' radio button. Provide a separate logical name to this job and click ok.




This will kick start a job in the background for finding the accurate (or exact) causes of the selected instance of the event i.e. 'user xyz from techlineage could not complete the transaction' which happened on 'Mon Jun 17 2013 12:31:38'. Here we are trying to narrow down on causes of a particular instance of the event. It will be interesting to view and understand the results of 'AccurateAnalysis' job.

Go to the 'Jobs' tab by clicking 'Manage' -> 'Jobs' OR if 'Jobs' tab is opened already simply refresh it by clicking the  icon. Click 'Show Results' icon to check the results of this job.



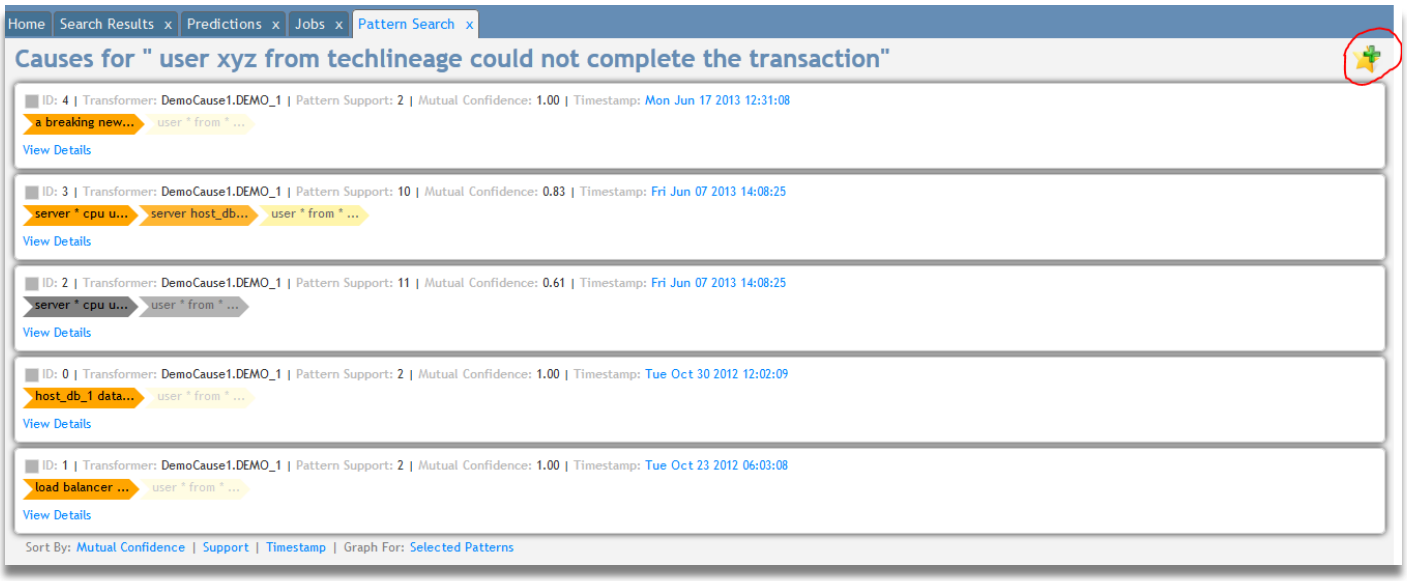

You can clearly see that the results of 'AccurateAnalyis' job have a lot less number of patterns (one in this case) than in case of generic causal analysis (Our previous example). The interpretation of the pattern output

remains exactly the same as explained before except for the Timestamp field. If you click on the Timestamp ("Mon Jun 17 2013 12:31:08), you will see only one timestamp which is maximum 600 seconds far from the timestamp of the event being analyzed (Note that the timestamp of 'user xyz from techlineage could not complete the transaction' is Mon Jun 17 2013 12:31:38). Here we report only one timestamp of the pattern as we are interested in analyzing the accurate causes for a single precise instance of the event. All the other statistical measures hold true and mean the same as explained before.

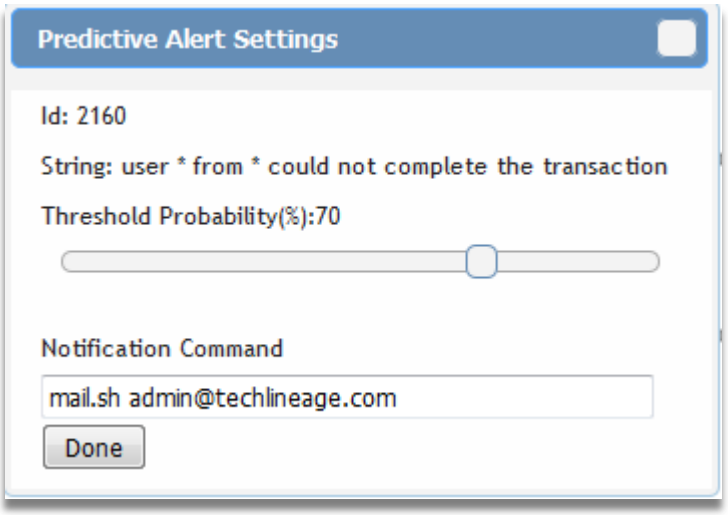
Important Note: When you select 'Accurate Analysis' checkbox, the analysis is performed on the specific instance of the event.

## Defining Predictions

After verifying the Causal results you may feel that it is worth predicting the event (Say, 'user * from * could not complete the transaction') as it is an important business event. Let us open the results of the first Job, DemoCause1. Click on the small icon (shown with red circle in the screenshot below) in the right corner of the 'Results' page




As small dialog will open as shown below




The first user input is the 'Threshold Probability' (or 'Mutual Confidence') for predicting the event. You can use the slider and adjust the probability threshold in percentage. By defining a value for the 'Probability Threshold' you tell Cogniyug to consider only those causes for prediction whose 'Mutual Confidence' is above the value of the 'Probability Threshold' defined using the slider. The default probability threshold is set to 70% and you may think this to be suitable for your prediction purpose. In the above example, if we consider the 'Probability Threshold' be 70% then four out of five discovered causes will be used for creating the prediction model as those four causes have the mutual confidence value above 70%. (See above screen shot. It shows four causes in orange and one in grey)

Also, enter a notification command in the small window shown. This command is the full path to any program or script (along with the command line arguments) that you wish to execute when the probability of any selected cause in the prediction model reaches the defined threshold value (i.e 70% in this case).

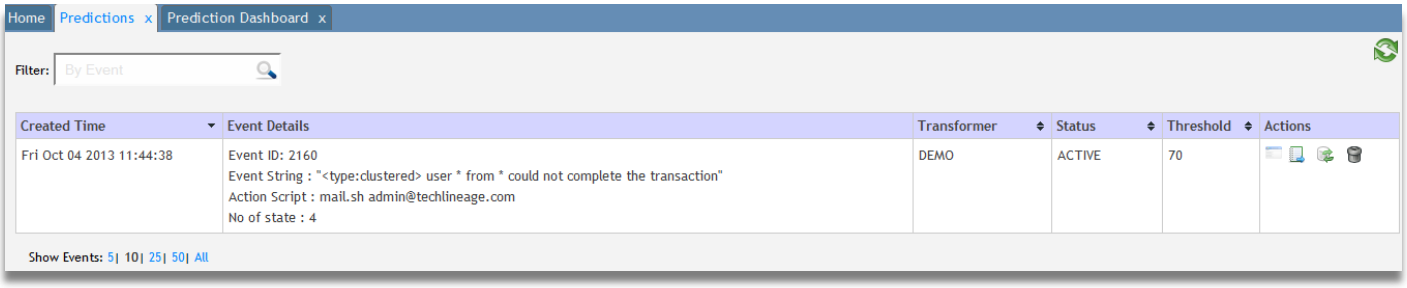
In the above example, we have chosen four out of five discovered causes for prediction. We will consider the second cause as shown below for the time being for understanding when the Notification Command will actually fire.


In this casual pattern, the Mutual Confidence will meet the defined threshold of 70% at event-2. This means, the Notification Command will fire when event#1 is followed by event#2 within 600 seconds. Remember, the causal analysis was done for Time Windows of 600 seconds and hence the predictions will be made in that window. Conclusion: - Cogniyug will predict the event#3 only when event#1 is followed by event#2 *within* 6000 seconds.

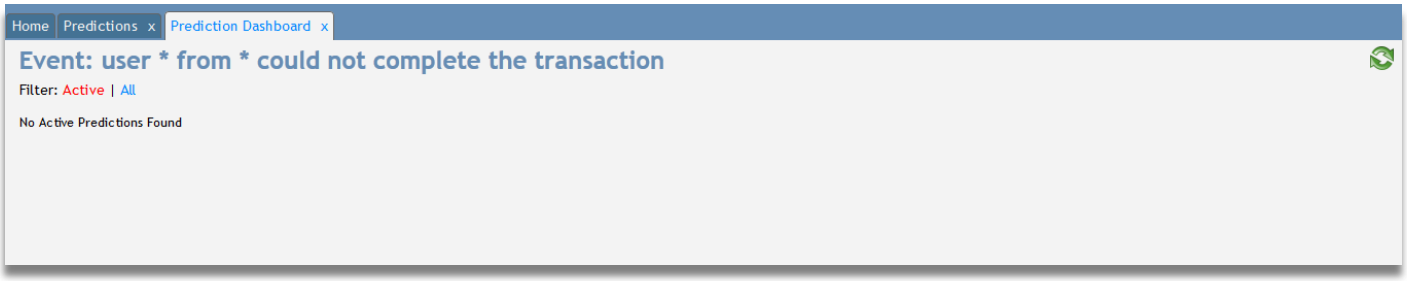
This explanation holds true for all the causes being considered by the prediction model.

### Managing and Monitoring Predictions (Prediction Dashboards)

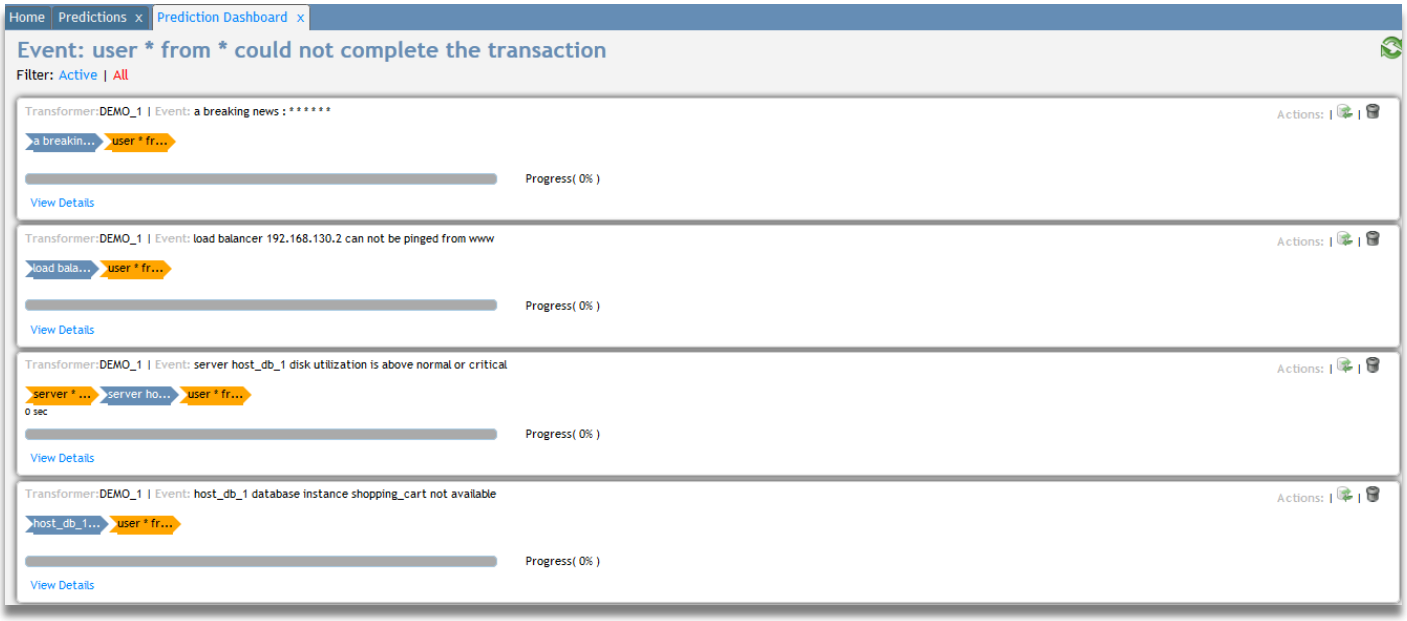
After a prediction for a particular event is defined, we can check the actual prediction model, manipulate it by deleting unwanted causes and monitor the real-time progress by clicking the 'Manage -> Predictions' link available in the top right hand corner of the browser. This will open up a 'Predictions' tab with summary of all the predictions defined by the user. The screenshot below shows only one prediction model defined for the event "user * from * could not complete the transaction". All the columns are self explanatory. Click the first icon  (Show Dashboard) in the 'Actions' column.




This will open up 'Predictions Dashboard' tab as show below.



By default, the dashboard is meant for monitoring the active states of the model and their real time progress. Since no data is being consumed by the system at this point of time, there are no 'Active' states of the model. Click on 'All' to check the defined prediction model for the event "user * from * could not complete the transaction".




This shows all the 'Association Rules' of the prediction model for event "user * from * could not complete the transaction". Each Association Rule consists of a list of temporally ordered events/log messages with one event in each rule being marked in 'blue'. The event node marked in 'blue' indicates the point at which the mutual confidence of the pattern will cross the defined probability threshold and system will *fire* prediction to trigger the Notification Action as explained before. For example, Check 3rd rule in the above screenshot which has 2nd event node marked in blue. If we were consuming the data continuously (which we will soon do) and if event#1 was followed by event#2 within 600 seconds, rule#3 would have fired causing the Notification Command to execute.

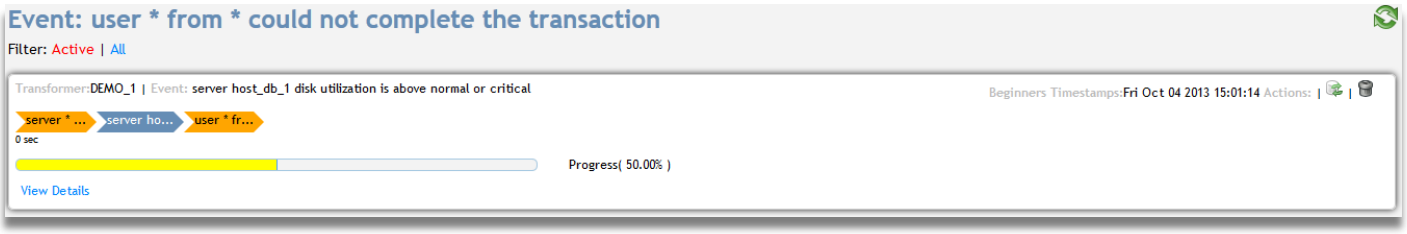
Click 'View Details' below each rule to check the details of the Rule. It will show you complete events stings and internal event IDs.

You can easily monitor the progress of the prediction model on the above dashboard. Since we are not consuming the data continuously at the moment, the progress of all the above rules is shown as 0%. In the sample below we will pump in some data to Cogniyug. Remember, the 'Polling Interval' for the Hook defined above (Page 13) is 600 seconds. This means, Cogniyug will look for new data every 600 seconds.

Perform following steps

1. cd to the directory where Cogniyug is Installed. (We call it COGNIYUG_ROOT)
2. cd $COGNIYUG_ROOT/Hooks
3. If you do ls -l in this directory you will find demo.txt and slow_prediction.sh files there.
4. Fire the command "./slow_prediction >> demo.txt" (Note: check the double redirection. This means that we are appending the demo.txt file. )
5. Our Hook script is written in such a way that it reads the new data from demo.txt file at every polling interval and writes on the stdout. The stdout of the Hook script pipes the data back to Cogniyug.

6. Wait for some time. Meantime examine the demo.sh script and try to figure out what that script is doing. Basically, the script reads demo.txt file each time it executes. Each time it stores the number of lines it read in a file, during the next execution, it refers to that record and reads the next sections of the file, if new data is available.
7. Now refresh the prediction dashboard, it should show one 'Active' state for the rule-2 as below




The progress of the rule is 50% as one event (i.e. event#1) out of event#1 and event#2 has been pushed into Cogniyug so far. Note: We are considering only event#1 and event#2 for calculating the progress of the rule as the rule will fire the prediction once it reaches event#2.

You can delete the rule from using the delete button in front of each rule in the prediction model. When you delete the rule from the model, the predictions will not happen for the causes of that particular rule.

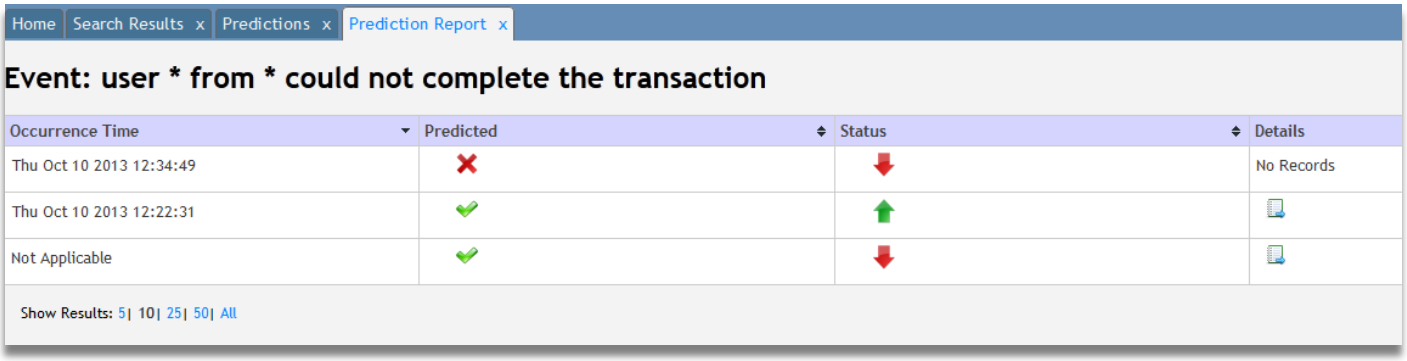
You can also reset the state of the rule by click the reset button. Resetting the state of the rule initializes the rule again. Simply meaning, Cogniyug will start watching for the causes of the event from the beginning. All the old states will be deleted.

## Prediction Reports

Go back to the 'Predictions' tab. In the 'Actions' column, the Report icon is available against each event to be predicted. The purpose of a prediction report is to measure and know the accuracy of the prediction model. Prediction report provides following statistical facts about the predictions for a particular event:

1. The number of times Cogniyug predicted the event and the event actually happened (True +ve)
2. The number of times Cogniyug predicted the event but the event actually did not happen (False +ve)
3. The number of times the event actually happened but it was not predicted by Cogniyug (False -ve)

When you click on the Report icon , a new Prediction Report Tab will open.




The Prediction Report tab has following columns

1. Occurrence Time: This shows the occurrence time stamp of the event that we are predicting

2. Predicted: This column indicates whether Cogniyug predicted the event. A correct sign indicates that Cogniyug predicted the event and a cross sign indicates that Cogniyug failed to predict the event when it actually happened.
3. Status: This column indicates whether the prediction was right or wrong. An up green arrow indicates the prediction was right. This is the case when the prediction is made by Cogniyug and the event actually happens (True +Ve) within the time window defined at the time of causal analysis. The down red arrow indicates that the prediction is wrong. There are two possibilities for this to happen
    A. When Cogniyug predicts the event but the event does not happen within the time window (False +Ve)
    B. When the event happens but Cogniyug fails to predict it (False -Ve)
4. The details column shows the actual causes that caused the event to occur. When you click on the details icon, it opens up a dialogue showing the details of the prediction patterns as below


You can see

- The exact prediction timestamp at the top indicating when prediction was fired
- A complete trail of the events in the pattern with time offset from the start of the pattern. In the above example, we can say that the prediction fired at 12:21:01, which is the timestamp of the event marked in blue. The event 'user * from *...' happened after exactly 90 seconds from firing the prediction.

### Reinitializing the Prediction Model

Go back to the 'Predictions' tab. In the 'Actions' column, the Re-initialize the icon is available against each event to be predicted. When you click this icon

1. All the existing states of all the rules associated with the event are deleted.
2. New rules are constructed based on the latest available data.
3. The dataset for creating the new rules is still the same as that was used while performing the Causal Analysis on the event to be predicted.
4. The same probability threshold value and notification command will be used in the new model.
5. If you wish to change any of #3 or #4, you must delete the model and redefine the prediction all over again.

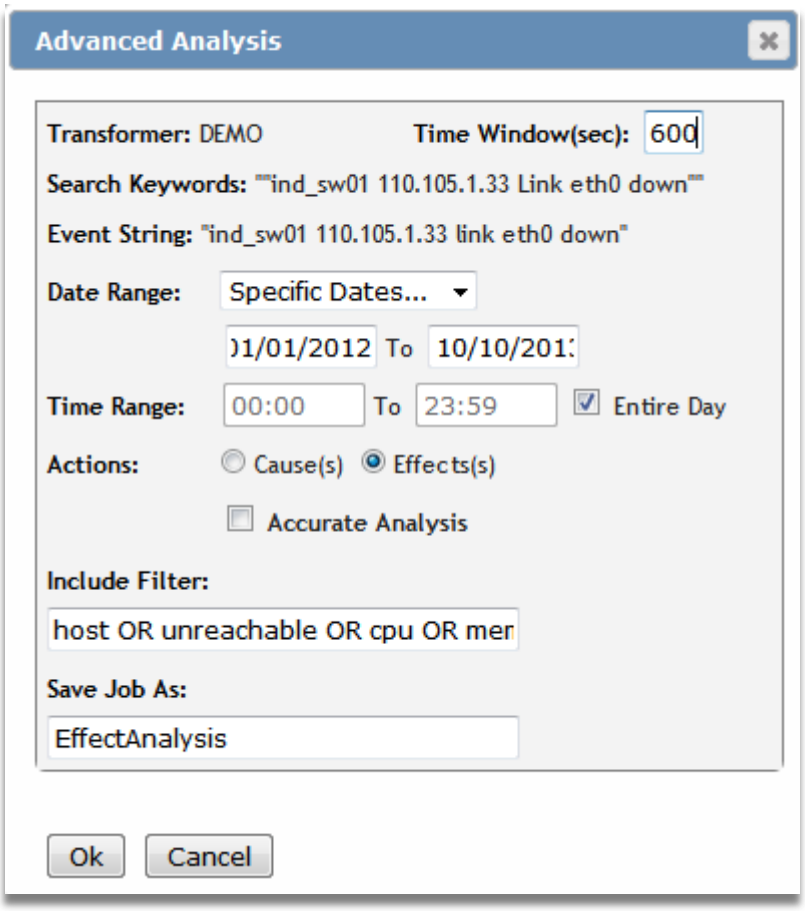
### Deleting the Prediction Model

Go back to the 'Predictions' tab. In the 'Actions' column, the delete icon is available against each event to be predicted. Click this icon to delete the model completely.

### Effect Analysis

Sometime, it is important to ascertain the effects of a certain events. It is quite possible that certain apparently benign events could have some serious implications. Perform the following step in order to do the effect analysis on a particular event.

1. Search the exact instance of the event using Cogniyug Search (Say "ind_sw01 110.105.1.33 Link eth0 down")
2. Click 'Advanced' and enter all the information
3. The search criteria can be constructed based on your knowledge in the Include Filter.
   *TIP: Use combination of tag:value pairs and strings to construct an efficient search criteria. Also, make smart use of Cogniyug Grammar.*
4. Select 'Effects' radio button, give a name to the job and click ok




Cogniyug will start processing the 'Effects' job in the background. Go to the 'Jobs' tab (or open it using 'Manage' -> 'Jobs') and locate the job you just submitted. (The last submitted Job appears at the top of the job list. If you don't see your submitted job, simply click the refresh icon. )

### Understanding Effects Output

Effects output is very similar to the output of Causal Analysis. The patterns discovered by Cogniyug are displayed with all the statistical measures as that in Causal Analysis. The explanation of all the statistical measures remains exactly the same. The only difference here is that we are trying to understand what happens 'after ' the event, where as in causal analysis we tried to analyze what happened before the event.

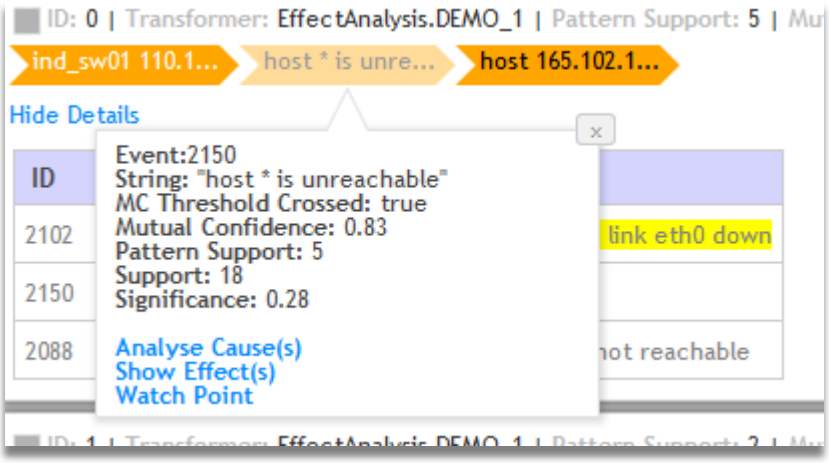

The above output shows that the event "ind_sw01 110.105.1.33 Link eth0 down" has two effects. One with a high confidence of 0.83% is marked in Orange and the other one with low confidence of 0.33% is marked in grey. (Again, the default mutual confidence threshold of 70% is used here in this example to decide whether the patterns to be marked in Orange or Gray.)

Note that the event being analyzed is marked in 'white text' with 'Orange background'

### Mutual Confidence Calculation

Mutual Confidence calculation of Effect Analysis is slightly different from the causal analysis as we want the Mutual Confidence measure to reflect the confidence of the after effects of the event. In the above example, consider the first pattern. Click on the node in the pattern where the event "ind_sw01 110.105.1.33 Link eth0 down" appears, i.e. the event with 'white text' and 'orange background'. Note down pattern support at this level. In this case the pattern support is 6 and the node appears as the first node in the pattern. Let this be S1 (Note: It is possible that the event being analyzed for effects may not be the first node in the pattern. It could be anywhere except at the end in the pattern. We have to consider the pattern support at his level as S1 for mutual confidence calculations.).

Now, move to right and click on the event next to this event in the pattern. Note the 'pattern support', S2 at this level which is equal to 5. The mutual confidence at this level = S2/S1 = 5/6 = 0.83. This is above our predefined threshold of 0.7 and hence we can move on further to the right in the pattern with a hope of finding a longer pattern that meets the threshold support. Remember, as we move the right S1 remains constant and S2 can only decrease. This means, the ratio of S2/S1 (the mutual confidence at a given point in the pattern) is going to decrease as we move to the right with pattern length increasing. We can hope for the ratio S2/S1 to remain above threshold value of 70% as we move to the right in the pattern from the event being analyzed.

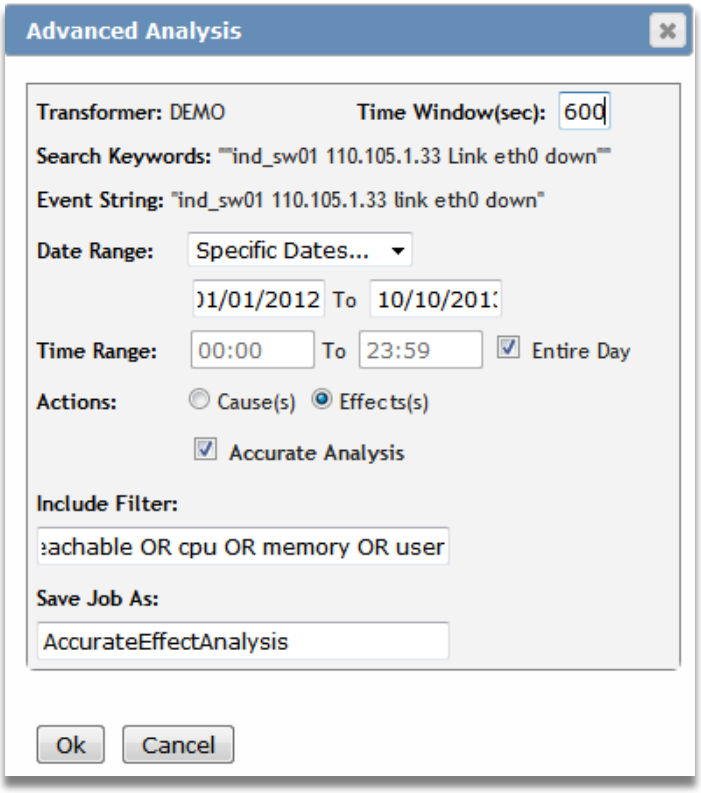



Click on the next event and check its pattern support and store it in S2. If S2/S1 is above 0.70, keep moving to the right till you either reach end of the pattern OR the ration S2/S1 falls below 0.70. In this case the pattern support till the very last event (i.e. till event-3) is 5 and the ration S2/S1 is 0.83. Hence we say that the mutual confidence is 0.83 for this pattern. In case, if the ratio S2/S1 at any level falls below 0.70, we mark that node in grey and everything after that node is marked in grey color. The pattern support of the previous node is considered for mutual confidence calculation in such cases and pattern is marked in orange till that node.

It may happen that the mutual confidence just after the node being analyzed for effects is below the threshold. In such cases, entire pattern after the node is marked in grey color and the value of the ratio S2/S1 at that level is considered as the mutual confidence of the pattern.

### Accurate Analysis for Effects

In some cases, you may want to ascertain the effects of a 'specific instance' of 'a specific event' at a 'specific time'. As explained in the section on Casual Analysis, you need to select checkbox 'Accurate Analysis' while defining your dataset in the 'Advanced Analysis' window as shown below. Give a name to the Job say "AccurateEffectAnalysis" and click Ok.




Click on Jobs tab and view results of this Job after it is completed.


You can see that only one pattern is returned by Cogniyug. The explanation for 'Accurate Analysis' in case of effects remains exactly the same as explained in the section for Accurate Casual Analysis (Page 27). The explanation for the Mutual Confidence calculation of the Effect Analysis holds true here.

This concludes the basic set of operations that you could perform with Cogniyug.

Following sections cover some miscellaneous topics

### What if 'Include' and 'Exclude filter' fields are Empty?

As explained in the section on Exclude Filter (page 14), a value in the Exclude Filter field tells Cogniyug what messages are to be excluded from sending to the pattern mining grid. Similarly, a value in the Include Filter field tells Cogniyug what messages are to be included for sending to the pattern mining grid. When both these fields are empty, the default behavior takes over which causes everything to go the pattern mining grid. This is exactly what we have experienced in the Quick Start guide. When a message is passed directly to the pattern mining grid, it is available for 'Quick Pattern Analysis'. All the quick pattern analysis options such as 'Causal / Effect / Pinpoint' analysis are available for every such event that readily goes to the pattern mining grid as a result of Include/Exclude filter configuration. When you click on 'Actions' against such events, you will see all the options as shown in the screenshot below.

Except for the 'ease of use', such configuration does not provide any advantages. Rather it has two main disadvantages:

- The analysis in such cases is performed on a dataset formed by all such events that pass to the pattern mining grid through the filter criteria. The dataset can be huge or pretty small causing to generate a lot of noisy patterns or missing important patterns.
- The Time Window for correlation is pretty much static and is equal to the value defined at the time of configuring the Transformer.

So, even after defining such criteria you will be forced to use the 'Advanced' option explained above.

TechLineage strongly recommends to leave the Include Filter empty and a value of '.*' in the Exclude Filter. Though it compromises a bit of convenience, this configuration has following two vital advantages.

- The data set for analysis is formed after careful consideration by the user and hence the results are more meaningful
- This ensures minimum load on the pattern mining grid as it works with only specified dataset.

### How to write your own Hook?

You may have your custom applications logging messages in their own format, databases storing messages in their own table structures, typical files or some third party applications that generate data and make it available through APIs which you may wish to import continuously into Cogniyug. Hence it is important to be able to write your own Hook scripts so that you can pump your data into Cogniyug and start analyzing it easily. It is extremely easy to write your own Hook Scripts as Cogniyug does not pose any restriction on the source of the data as long it conforms to the standard Cogniyug Data Format. (Please refer to section on Cogniyug Data Format on Page 8).

There are three main steps to be considered while developing your own custom Hook script.

1. Read the data from the source: This is perhaps the most critical step as #2 and #3 are relatively simple. It may involve reading a simple text file or may involve reading the data from a third party application using the third party APIs. In all the cases, you'll need to ensure that you have appropriate read access

to the source of the data. (NOTE: The user account executing the Hook script must have read access on the source data)

2. Convert the data into Cogniyug data format. This step is pretty straight forward as you already know the Cogniyug Data format. It is important to know the format of the source data format that you are reading as you have extract timestamp (date and time) from each message. We recommend you to add meaningful tags in this step so that it becomes easy to deal with the data in Cogniyug at a later stage.
3. Write the Cogniyug formatted data line-by-line on the stdout and exit after the available data is written

Cogniyug executes the Hook script after every 'Polling Interval'. It is important to make sure that the Hook script/program reads only the incremental data at each execution. Make sure that the Hook does not re-read all the data each time it executes. Consider a Hook which is a shell script that reads lines from a text file. It is important that the Hook script knows how many lines it read during the last execution and read only next available lines, if any, during the subsequent executions. We explain it with an example below.
Open demo.sh script installed in $COGNIYUG_ROOT/Hooks directory

```
**********************************************************************************************************
      1 #!/bin/bash
      2 if [ -e /home/COGNIYUG/Hooks/out.txt ]
      3 then
      4         line_already_read=`cat /home/COGNIYUG/Hooks/out.txt`
      5         no_of_lines=`wc -l /home/COGNIYUG/Hooks/demo.txt|awk '{print $1}'`
      6         lines_to_read=$(($no_of_lines - $line_already_read))
      7         echo `date` " : " $lines_to_read>>/home/COGNIYUG/Hooks/log.txt
      8         tail -n $lines_to_read /home/COGNIYUG/Hooks/demo.txt|awk '{printf("%d ",
    $1);for(i=2;i<=NF;++i){printf("%s ", $i);} printf("\n");}'

      9 else
     10         lines_to_read=`wc -l /home/COGNIYUG/Hooks/demo.txt|awk '{print $1}'`
     11         echo `date` " : " $lines_to_read>> /home/COGNIYUG/Hooks/log.txt
     12         head -n $lines_to_read /home/COGNIYUG/Hooks/demo.txt|awk '{printf("%d ",
    $1);for(i=2;i<=NF;++i){printf("%s ", $i);} printf("\n");}'

     13 fi
     14 wc -l /home/COGNIYUG/Hooks/demo.txt| awk '{print $1}' > /home/COGNIYUG/Hooks/out.txt
**********************************************************************************************************
```

- The script basically checks for out.txt file in certain directory. This file is used to keep the record of total lines read so far. If you are executing it for the first time, the file out.txt wont exist and hence we'll execute the *esle* part on line 9
- Here we use wc -l command (line 10) to find the number of lines in file demo.txt, which is our source file to read. We store in a variable *lines_to_read*
- We log an info message on line 11
- On line 12 we read *line_to_read* number of lines from the top using head command of the source file
- line 13 is Unix shell syntax for indicating the if loop is ended
- On line 14 we calculate the number of lines in the source file using the *wc -l* command and store it in out.txt which we refer everytime. The script ends here.
- During the next execution, we find the demo.txt to file (line 2) and read how many lines we read the last time (line 4). We store it in variable *line_already_read.*
- On line 5 we find total lines in the source file demo.txt using the wc-l command and store it in variable *no_of_lines*.

- On line 6 we take the difference between the number of lines in file right now (*no_of_lines)* and lines read till previous execution (*lines_already_read).* If the file is appended during the polling interval, the difference will be a positive number (*lines_to_read)*
- On line 7 we log an informative log message
- On line 8 we read exactly *lines_to_read* number of lines at the end of the file using *tail* command.

You may have observed that this script reads a simple text file and writes the message on stdout (using head and tail commands of Unix). It stores the number of lines read (in a file) so that the next execution refers to it and reads only the appended portion of the file. It does not perform any epoch time conversion because the source file (demo.txt) has the time stamp in the epoch format. This is an unlikely case as most application would log the timestamp in a human readable format which will have to be converted to the epoch time so as to conform to Cogniyug format. In the next example, we read the /var/log/messages file (the syslog format) using a python script and explain how to convert a human readable timestamp into an epoch time.

In the $COGNIYUG_ROOT/Hooks directory, open the file read_syslog.py

**********

```
10 import time
11 import socket
12
13 input = "/var/log/messages"
14 #Change this path to Cogniyug's Hook directory or some safe location
15 meta_data = "/tmp/metadata.txt"
16
17 try:
18      lines_read = int(open("/tmp/metadata.txt", 'r').read())
19 except:
20      lines_read = -1
21
22 if (lines_read == -1):
23      skip_lines = 0
24 else:
25      skip_lines = lines_read
26 raw_input
27 count = 0
28 with open(input, "r") as f:
29      for line in f:
30           if ( count >= skip_lines):
31                split_line = line.split()
32                month = split_line[0]
33                date = split_line[1]
34                year = str(time.gmtime().tm_year)
35                timestamp = split_line[2]
36                hours = timestamp.split(':')[0]
37                minutes = timestamp.split(':')[1]
38                seconds = timestamp.split(':')[2]
```

```
39                  MyStr = year + " " + month + " " + date + " " + hours + " " + minutes + " " + seconds
40                  s = time.strptime(MyStr, "%Y %b %d %H %M %S")
41                  epoch_time = str(int(time.mktime(s)))
42                  #Write the output to stdout in Cogniyug format
43                  print epoch_time+": <host:"+socket.gethostname()+" file:/var/log/messages>" ,"
".join(split_line[3:])
44                  lines_read += 1
45              else:
46                  count += 1
47                  continue
48
49 open("/tmp/metadata.txt", 'w').write(str(lines_read))
```

**********

The core logic of read_syslog.py is more or less the same as that of demo.sh explained earlier. It basically reads the syslog format file (/var/log/messages) all at once. Stores the number of lines read in '/tmp/metadata.txt' which it refers at each execution. It skips already read lines and reads only incremental data. It parses the file to extract the date field from each line, converts it in Cogniyug data format and writes the output on stdout. The important step is parsing of the data i.e. lines 31 to 43.

Before we actually wrote this simple parser, we had a look at the data by opening the /var/log/messages file to understand the format of the file. We observed following

- Each line has timestamp at the beginning
- The format of the time stamp is consistent across the file
- The date field, which is at the beginning of the file has a peculiar format: "MM DD Hours:Minutes:Seconds"
- The important thing to note here is the year is missing from the timestamp.

This much information is good enough for us to write the parser. This is how we wrote our parser

1. We split each line into a list of words (using whitespace as the separator. This works in most of the log files)
2. Now, we know that the first word in the line is a Month, second is the date and the third is the timestamp, which we further split using ':' as the separator. This gives us Month, Date, Hours, Minutes and Seconds, pretty much all the fields but the year, which we populate using the value of the current year. (Note: python indexes start with 0)
3. On line 39, 40 and 41 we use simple python time library functions to convert this date into epoch time. Please refer to the documentation of strptime() and mktime() library functions that we have used here on http://docs.python.org/2/library/time.html. If you use any other scripting/programming language such as perl, JAVA, Ruby, C, C++ etc, you should get access to many standard time library functions as all the popular languages provide a rich interface to time library.
4. Other part of the message remains the same
5. Finally on line 43, we print the message in Cogniyug Data format on stdout.
   *Note: Needless to mention, but the account executing the' Hook Script' must have read permission on /var/log/messages file*

We plan to ship a bunch of Hook scripts along with Cogniyug soon. We also expect community to provide a bunch of free Hook scripts so that you can readily use them. In the meantime, if you need any help for writing the Hook scripts, please write to us on support@techlineage.com