MURAL Operation and Troubleshooting Guide

Version 3.9

Published: 2016-10-07

Americas Headquarters

Cisco Systems, Inc.

170 West Tasman Drive

San Jose, CA 95134-1706 USA

http://www.cisco.com

Tel: 408 526-4000

800 553-NETS (6387)

Fax: 408 527-0883

Table of Contents

Understanding the Reflex Platform

This topic provides a detailed description of the Reflex platform, its components, and how each component operates on data.

The Reflex platform is a resilient, scalable, and configurable system that powers a variety of solution suites, each tailored to ingest particular kinds of data streams and deliver insights that enable you to take action or that trigger an automatic action. Specialized platform components collect high-volume data streams, then fuse, extract, process, correlate, analyze, and store the data, making it available to the Reflex solution user interface or to third-party tools and applications.

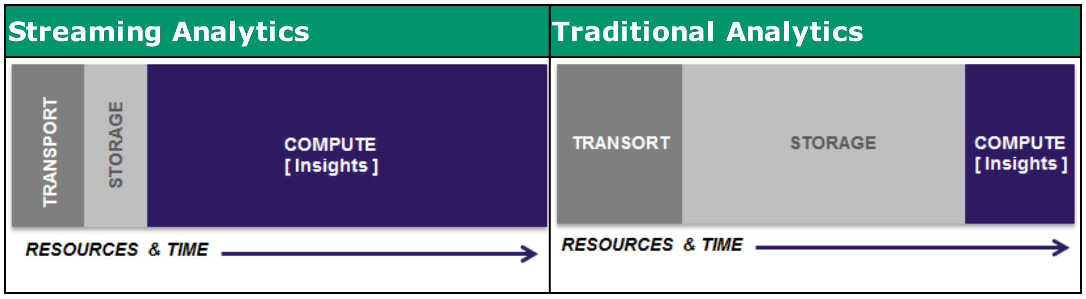
Most of the functional details of the Reflex platform remain the same across applications; however, there is variance in how the platform operates on data and how components are tuned and configured for different applications and particular deployments. For example, the type of data flows and static data ingested into the system are tailored to individual solutions.

Introducing Reflex Platform Technology

The Reflex platform technology captures multiple streams of dynamically generated data, enriches them with static data, extracts and analyzes the data, and responds to requests from Reflex solution user interfaces.

The user interfaces provide visualization tools that allow maximum access to your network data and the flexibility to slice and dice your data in new ways and drill into it on any dimension as you think of new questions to explore.

Reflex platform technology is based on streaming analytics, an innovative paradigm in which data is analyzed as it is collected—overcoming the limitations of traditional technologies. Streaming analytics moves processing to the source of the data so that resources and spending are focused on analytics rather than transporting and storing massive quantities of raw data. This approach also allows you fast access to processed results—data is continuously processed so that actionable insights for critical business decisions are quickly available.



The streaming analytics approach also leverages off-the-shelf, low-cost processing and storage components, providing answers to new analytics questions with fewer resources.

Introducing the Reflex Platform

The Reflex platform uses grid-computing architecture, with collection engines located at each data collection site to collect and analyze data, which is aggregated and made available to solution user interfaces and third-party tools for analysis.

The Reflex platform inter-operates with your existing network infrastructure and data repositories and does not require you to distribute proprietary data collection devices or probes in your network.

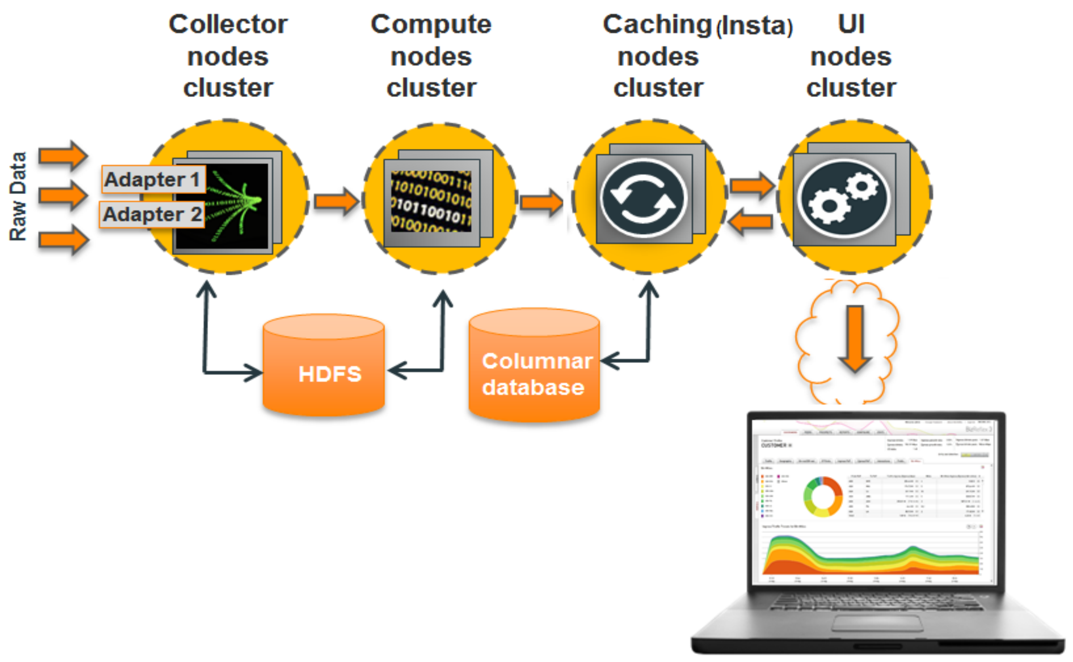
The Reflex platform operates in a fault-tolerant mode where the failure of any component is fully sustained by an alternate component. If any component fails, all components in the platform adjust their configuration to work with the new topology.

System Components and Flow of Data

The Reflex platform consists of six types of nodes, each with its own function: ingesting, enriching, processing, caching, delivering visualizations of the data to the user interface, or managing the system.

The nodes are conceptual and several types of node might be hosted on one server blade. For example, the UI and Caching nodes are often combined on one physical server and some installations (Starter Pack) combine General Management Server (GMS) with Collector and UI. The following figure shows how the data flows through the system.

Note: The GMS node handles centralized software installation on and monitoring of the other nodes, rather than data, and is not pictured.



The nodes perform the following functions:

- **Collector nodes**—Collect the data streams that are pushed to the Reflex platform, interpret the flows, enrich them with static data, and assemble data sets. The Collector node is optimized for low-latency, high-throughput transactions. It stores the raw data in the Hadoop file system (HDFS) and sends it to the Compute node. A Collector node cluster has any number of servers, in pairs for master and standby, and uses 1+1 redundancy (transparent failover between pairs of active-active nodes).

- **Compute nodes**—Analyze and aggregate the data, creating *data cubes*. (The term data cube is a convenient shorthand for the data structure, which is actually multi-dimensional and not limited to the three dimensions of a cube.) The Compute node cluster can have any number of servers, depending on the implementation, and uses N+1 redundancy.
- **Insta nodes**—Store and manage the processed data cubes in a columnar database, the Insta database, which stores from three to six years of data. The Insta node cluster contains two servers with 1+1 redundancy.
- **Caching nodes**—Host the Rubix engine and data cache. The Rubix engine queries the Insta nodes constantly for new data, and fetches it immediately for storage in the data cache, so that it can respond more quickly to requests from the UI engine. The Caching node is sometimes called the Rubix node. Caching nodes use N+1 redundancy.

  Note: In applications to which processed data is delivered in flat files, the data is sent from the Insta node to the Service Gateway, bypassing the Caching and UI nodes.
- **UI nodes**—Host the report generation engine (RGE). UI nodes are often co-located on the same servers as the Caching nodes, in which case the nodes are referred to as UI/Caching nodes. UI nodes use N+1 redundancy.
- **GMS**—General Management Server (GMS) provides centralized management of the Reflex platform nodes, such as remote manufacturing of blades, patch management, monitoring of all nodes and operations, and importing and running node configurations.

You can scale the system horizontally for increasing data flows or users by adding Collector and Compute nodes.

Each type of node is discussed in more detail in the following sections.

Collector Nodes

The Collector nodes collect the data flows pushed from input systems, and host the Collector process and the NameNode process.

Adapters are hosted on the Collector nodes. An adapter is a plug-in software component that collects a specific type of data stream or source. The Collector nodes write the raw data to time-defined bins in the HDFS. For information on binning,"Understanding Data Bins" on page 21.

Collector Process

The Collector process is simple—it listens at a port (or in some cases, reads a file), reads the data from the port, does a quick check, and writes it to the Hadoop Distributed File System (HDFS), which is described in the following section.

HDFS and the Collector Node Cluster

HDFS is a distributed, scalable, and portable file system written in Java for the Hadoop framework. It stores three types of data—raw data from the Collector nodes (1 to 30 days), processed data from the Compute nodes, and information bases (IBs) from other data systems. The HDFS repository is stored on a storage area network (SAN).

The connection between the Collector node cluster and the Compute node cluster is monitored by the backup_HDFS process. If the connection breaks, the Collector process stores data locally until the connection is restored.

Spark

Spark is fast and expressive cluster computing system compatible with Apache Hadoop. It improves efficiency (up to 100× faster) through in-memory computing primitives and general computation graphs. It improves usability (often 2-10× less code) through rich APIs (in Java, Scala, Python) and interactive shell.

Spark works with distributed collections called as resilient distributed datasets (RDDs). These immutable collections of objects are spread across a cluster and are built through parallel transformations (map, filter, etc).

NameNode Process

The master and standby Collector nodes also host master and standby instances of the NameNode process, which is the controller for the HDFS file system. The NameNode process tracks the file locations of all data, enabling it to transparently assign jobs in Compute nodes to run on a processor that is either co-located or near the data needed by the job. Even though this pertains to Compute nodes, the NameNode process is generally configured to run on the Collector node where the CPU load is lighter.

The NameNode process does not have high availability, therefore a process called **drbd** synchronizes the NameNode's metadata on the master and standby Collector node. The metadata file is **/dir/fsimage**. A corresponding edit log, **/dir/editlog**, records the last few insertions, deletions and movements within HDFS (which are not yet captured in the metadata).

High Availability of Collector Nodes

Collector nodes are always configured with 1+1 redundancy and with a virtual IP address (VIP). If the master Collector node goes down, the other Collector node comes up. The node is considered down if the heartbeat check which is conducted every three seconds is missed a number of consecutive times. For example, if the node is considered down after seven consecutive failures, it takes 21 seconds until the standby Collector node knows that the master is down.

Once the standby determines that the master is down, it generally takes less than a minute for the Collector process on the standby Collector node to come up, depending on the amount of data on the node.

Compute Nodes

Compute nodes take the raw data pushed from the Collector nodes and fuses, aggregates, and analyzes the data to create data cubes. The CubeExporter process periodically pushes completed data cubes to the Insta node cluster.

Compute nodes run under a Hadoop framework which provides reliability and the capacity to handle massive quantities of data quickly and efficiently. (The Compute nodes of the Reflex platform are equivalent to Data nodes in Hadoop terminology.) The number of Compute nodes in a production system depends on the amount of data the system handles, from 2 nodes to 50 or more.

As processing power has exponentially increased over time, the speed of data delivery to the processors has become a more significant bottleneck. The Hadoop framework for parsing the distribution of data to the Compute node processors is designed to eliminate this bottleneck. (It functions transparently and generally you do not need to know about the details of how it is operating, but some explanation is included here so that you can understand the processing flow.)

MapReduce

The Compute node uses the Hadoop computational paradigm called MapReduce, in which an application is divided into many small fragments of work, each of which may be executed or re-executed on any node in a cluster. Distributing the processing work among the nodes is one of the mechanisms that allows the system to complete processing in near real-time. It also reduces the flow of data across the network, instead of moving the data, computation is done on the Compute node.

The MapReduce infrastructure transparently runs tasks in parallel, manages all communications and data transfers, and automatically handles node failures without loss of data. MapReduce provides distributed processing of the map and reduction operations. *Map* refers dividing processing of data inputs into smaller sub-problems and distributing them to processing nodes; this can be done multiple times and result in a multi-level tree structure. *Reduce* refers to the subsequent step of collecting and aggregating the results from all of the processing nodes to form the output.

The Reflex platform uses the replication feature of Hadoop, keeping at least two copies of all data for high availability, so that a hardware failure never causes data loss. The Compute cluster uses N+1 redundancy.

MapReduce also supplies high availability on the Compute nodes. The NameNode process tracks heartbeats of Compute nodes, and if a node goes down, it copies the blocks that were on that node to another node. Likewise, if there are excess copies of data blocks, the NameNode process removes some blocks.

The MapReduce functionality allows the fusion of static data from IBs on a per-record basis during the aggregation stage.

Crux

Through Crux, a framework that only requires solutions to write business logic and define input/output cubes is made available. They are not required to think of compute jobs or optimizing them. With the new framework, simple configuration changes in the list of output cubes automatically identifies and executes only the minimum business logic required. Therefore, solutions can define all their business logic in one place and let the framework choose the context-specific subparts for each of the use-cases.

The benefit of seeing entire business logic as a single application rather than as a set of connected jobs is that management and operations becomes a lot easier. Field support teams can easily see failure status, perform restarts, and monitor statistics. It is also easier to configure parameter sets for the entire application rather than specify/tune them independently for each job.

In the new framework, annotations are first-class citizens and the primary purpose of the processing. Therefore, the focus is on the user to only write the business logic and not worry about the optimizations. The framework makes it possible to call it on map-side or reduce-side easily. It is now easier to plugin IB management and synchronization with approaches such as sharing single memory instance of IBs across multiple tasks, versioned IBs with incremental updates and so on.

HDFS and the Compute Node Cluster

The Hadoop distributed file system (HDFS) is a repository that stores three types of data: raw data from the Collector nodes (1 to 30 days), processed data from the Compute nodes (multiple years), and information bases (IBs) from other data systems (static data).

iNSTA Interface APIs

To persist an RDD of any arbitrary type, iNSTA requires the following information that is provided through iNSTA APIs:

- The id of the cube the RDD represents. All other information about the cube like the dimensions, measures and string-ID tables is present in the cube definition XML for the cube ID.
- A function to convert a value of the arbitrary type to a row containing the dimensions and the measures in the order specified in the cube definition XML.

Insta Nodes

The Insta nodes store and manage the processed data cubes generated by the Compute nodes. Processed data cubes are stored in the Insta database, a columnar database which generally contains three to six years of data. The amount of data depends on its granularity and the amount of storage available, which can vary widely depending on the deployment.

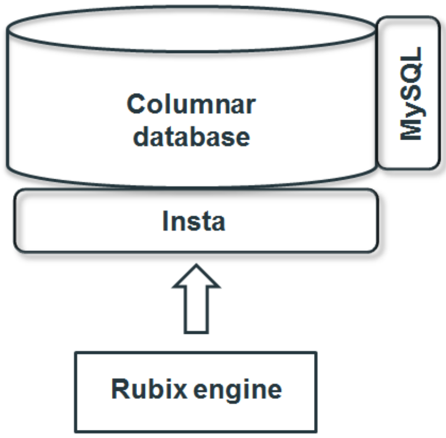
The Insta process serves data from the Insta database in response to queries from the Rubix engine, which runs on the Caching nodes.

There are two Insta nodes, configured as master and standby in active-active mode.

Insta Database

The Insta database is used by the Insta nodes to store the processed data cubes. The Insta process divides the columns (representing dimensions) of each cube on different partitions, and tracks the location of each. At least two copies of all data sets are stored, for full redundancy in the event of a hardware failure.

The Insta database is organized in a star schema with fact tables that contain numeric values and foreign keys to dimensional data, and dimension tables that contain records and associated descriptive attributes.



When the Insta process queries the columnar database, only the columns which contain relevant data are requested, thereby greatly improving the response time of the database.

In order to optimize storage space and costs, the Insta node maintains the highest level of granularity for only the most recent data. For example, the Insta node might store 5-minute cubes for the past five days. Each day it aggregates the day's worth of 5-minute cubes from 6 days ago into a single 24-hour cube. Under this scheme, data is available at 5-minute granularity for the past 5 days, and at 1-day granularity for days before that. These aggregated sets can be stored for a period of time, usually a year or several years, before being purged from the database. The amount of storage time is primarily determined by a business decision of the amount of resources to devote to storage space. This optimization removes some of the granularity, but exponentially increases the speed at which queries can be processed, in addition to allow a much longer storage time for historical comparisons.

Caching (Rubix) Nodes

The Caching node hosts the Rubix engine and a cache of data that is prefetched from the Insta node so that it is ready when requests are received from the UI. The Rubix engine can be deployed on any number of Caching nodes; the number of Caching nodes deployed in a given implementation depends on the volume of data and how many concurrent users (people making active queries) are supported.

The Caching nodes are accessed by the UI and Insta nodes by means of a virtual IP address for high availability. The system keeps track of which actual node the requested data is stored on and fetches it.

The Rubix engine forwards requests from the report generation engine (RGE) to the Insta node and manages the cache of processed data that it automatically fetches periodically. The Rubix engine constantly queries the Insta node, and when new data comes in the Insta node, the Rubix engine re-fetches and rebuilds the data cache.

Because most users are interested in what is currently happening and do not need to go back and access data beyond several months past, the cache is optimized for this scenario storing daily data rather than hourly in the Rubix cache.

The prefetched data cache consists of several days of data, and allows fast responses to requests from the user interface. If the request is in the cache, the Rubix engine serves it from there, and if it is not, it sends a request to the Insta nodes and rebuilds the cache. Users might notice that their first request, or a request for an entirely different time span, is slow to load—this is because the cache is being built or rebuilt. Subsequent to the initial building, the UI node automatically and transparently refreshes the cache. The initial build of the cache begins when the user interface first sends a query.

The cache is generally distributed on multiple nodes in order to increase availability and minimize the negative effects on performance of multiple users.

Caching nodes have N+1 high availability. If the master node goes down, the standby node becomes master and one of the normal caching nodes becomes the standby.

Acume

Acume is the query engine of the platform. Once the data has been preprocessed and made available in a data source, Acume can be configured and used to query the data using SQL interface available through REST and Blazeds (java/flex binary). The data can be optionally cached and pre-populated for the known data sets.

Acume acts on the data after the data has been preprocessed by Crux. Acume can also connect with other data sources apart from HDFS, including data bases.

Spark SQL

Spark SQL lets you query structured data as a resilient distributed dataset (RDD) in Spark, with integrated APIs in Python, Scala and Java. This tight integration makes it easy to run SQL queries alongside complex analytic algorithms. Spark is an alternate to MapReduce for much faster and large-scale data processing. It is used to provide a fast and powerful distributed SQL query interface over Spark.

UI Nodes

The UI node functions are often hosted on the same servers as the Caching node functions. The UI node hosts the report generation engine (RGE), which serves as the HTTP request server for all requests from the user interface and generates files in comma-separated value (CSV) or Excel (XLS) format. Reports have more records than are displayed in the user interface, which might show only 10 or 20 records.

Some Reflex solutions do not have a GUI, in which case the data is packaged in flat files that are exported as input to other applications.

User Interface

The Reflex user interface is designed on an Apache Flex framework and displayed in a Flash-enabled browser.

For information about the user interface, see the user guide for your solution suite.

Note: A few solutions do not have user interfaces. These require only output of flat files. In these case, the implementation does not contain either UI or Caching nodes; instead the data is sent as a flat file from the Insta node to a Service Gateway.

GMS Nodes

The General Management Server (GMS) provides the ability to centrally manage the Reflex platform nodes; for example, it allows you to:

- Install software and manage upgrades across a large-scale deployment for new installations and new nodes
- Manage configuration—deploy configuration templates across the deployment, upgrade and downgrade configuration
- View the topology
- Manage clusters
- Monitor performance of all nodes and operations
- View logs and alarms for maintenance and diagnostics
- Integrate with Operations Support Systems (OSS), using a north-bound interface

GMS provides a Zabbix interface for monitoring purposes. Zabbix is an open-source performance monitoring solution with a graphical interface that enables you to monitor all the nodes configured on the GMS, view alerts, and view aggregated statistics such as number of records coming in to the Collector nodes, number of records being dropped, or statistics from each of the clusters.

Information Bases

Information bases (IBs) are lookup tables that contain dictionaries used by the system to fuse static data with streaming data, and derive and annotate new fields. For example, the system can use IBs for the following purposes:

- Assigning URLs to categories defined in the IB
- Identifying applications and device models from user agent fields

- Identifying manufacturers from device models
- Identifying clients from POP, router, or interface identifiers
- Identifying egress routers from ingress router and prefix values
- Deriving the autonomous system (AS) number of peers, next hops and destinations from ingress router and prefix values

In some solution suites, IBs are updated whenever the data mining team determines that there has been a major change in the available information and updates are made available in the next release or patch release. In other solution suites, information jbases are manually edited and updated on a regular basis.

Scheduling and Running Jobs

The Reflex platform uses the Oozie workflow scheduler, designed by Apache, to manage Hadoop jobs. Oozie is bundled into the Reflex platform, hosted on the Collector nodes, and accessed from the PMX subshell of the Reflex platform CLI. All jobs on the Reflex system are Oozie jobs, with the exception of Insta jobs. Oozie runs as a Tomcat application.

To define a job in the Reflex platform CLI, navigate to the Oozie subshell and define the job start and end times, type, frequency, actions within the job (taken from an XML file written by the developers), and pre-conditions and post-conditions for the job. You can view the complete configuration of Oozie and every job from the Oozie subshell of the CLI.

Understanding Data Bins

Data binning is a pre-processing technique in which a number of individual data values are grouped into one instance, called a bin (also sometimes called a bucket). In the Reflex platform, data for a five-minute time interval is combined in a bin specified for that interval, and is presented in the user interface as one aggregate value for the interval. Five-minute bins provide an optimal compromise between presenting data at a high level of granularity and processing it for fast delivery.

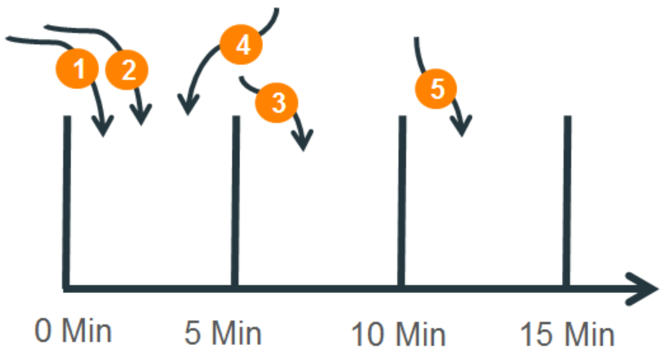
Both the number of bins open at a given time and the size of the bins are configurable on initial installation, so it is possible that your solution might vary from the standard settings—the standard number of open bins is 2 or 3 and the standard bin size is 5 minutes.

Note: The Reflex platform stores timestamps in the Unix time format (the number of seconds since January 1, 1970). Because Unix time does not include leap seconds, it is not a true representation of UTC.

The Collector node creates a bin for each 5-minute interval in an hour. The first bin of the first hour (12:00 to 12:05) is 01/00 and the last bin of the first hour (12:55 to 1:00) is 01/55 , the second hour's bins are 02/00 to 02/55, and so on. The Reflex platform stores all data with a structure type of /Y/M/D/Min.

The Collector process writes raw data to a bin until the end of the time interval. Then the Collector process creates the next bin.

The following example illustrates how the collector process decides when to create and close bins and in which bin to place an incoming packet.



The following example illustrates a system which has two open bins at a time. The timestamps of packets arriving at the Collector node determine which bin they are placed in.

A. The collector process deposits packet 1 and packet 2 in the 01/00 (0-5 min.) bin.
B. When packet 3 arrives with a packet creation time of 00:05 minutes, the collector process creates the 5-10 minute bin.
C. Subsequently, packet 4 arrives with a packet creation time of 00:04 minutes and the collector process puts it in the 01/00 bin.
D. When packet 5 arrives with a creation time of 00:11 minutes, the Collector creates the 01/03 bin and closes the 01/00 bin.
E. Any packet with a creation time before 00:05 that arrives at the Collector node after the 01/00 bin has been created is dropped, because the appropriate bin is no longer open. (Likewise, if a packet with a creation time later than 00:10 arrives at the Collector node before 5 minutes after the hour, it is also dropped, because the 10-15 minute bin has not yet been created.)

A network session is an interactive exchange between two or more devices and may have more than one message in each direction. The Reflex platform is flexible in handling session data depending on the solution, however for most solutions, the system uses the creates a packet for the entire session based on the session end time. Therefore, long sessions, such as a long download, are binned at the end of the session time. Other solutions might be designed to pro-rate a session across bins rather than wait for the close of the session to bin it.

Tip: Time is based on the system clock, therefore it is important to synchronize the time on the Reflex nodes and on the hosts that provide the raw data.

If the collector process goes down for a short time (such a minute), upon restart it opens the next two bins after the last successfully closed bin in the HDFS (the most recent for which a **.DONE** file was created). If the current system time is beyond these bins, the process quickly opens the new bins and closes them as empty.

Auto Bin Sliding

Auto bin sliding is a feature that can prevent jobs from stopping in the event that data flow is interupted. When auto bin sliding is enabled, they system opens and closes bins irrespective of how data is arriving and if data stops coming in for any reason, jobs that depend on creation of bins will proceed. Hence, the collector is always at the current time and jobs are running to continue to provide the complete historical view. If auto-bin sliding is off, jobs might become stuck if data stops coming in.

Auto-bin sliding is enabled by default and only disabled in rare situations.

Attribute-Based Binning

Attribute-based binning can be used to speed up MapReduce processing time. Attribute-based binning is supported with two types of adapters, IPFixAdaptor (**srcIP** column only) and NetflowAdaptor (**routerName** column only). As the adapter parses data, it creates indexes for these columns, which speeds MapReduce jobs that process this data for a given source IP or a given router name. If attribute-based binning is enabled, there is an additional file called **.index**.

Replay Mode

Replay mode is used in a test environment to ingest or load data with older or future timestamps. In replay mode, the first bin is opened according to the timestamp of the first record.

Before switching over to a production environment, switch from replay mode to live mode.

Prorating

Some adapters, such as NetFlow, have prorating enabled by default. When prorating is enabled, the Reflex system handles any record that expands across multiple bins by proportionally splitting it into the relevant bins.The state parameter is set to 2 for records that have been prorated (even if the record is split across three bins).

The complete original record is also retained for use by jobs that only operate on non-prorated records, and its record state is set to 1. Record state is set to 1.

Frequently Asked Questions About Data Flow and Display

Q: Under what conditions will a packet be dropped in the Reflex Platform system?

A: Packets are dropped only under two conditions: when the packet is too new or too old (it lies outside of the currently open bins), or if it has a format error.

Q: Why does the total data I see in the user interface not match the total data that I find from the sending system?

A: Aggregation or rollup (summarizing data along a dimension) is inherent to big data analytics. Each subsequent rollup results in a small loss of precision. The rounding or truncating of values with each rollup can accumulate to result in as much as a 5% variance in grand totals. The choice is made to provide results quickly versus retaining accuracy to the last decimal place.

In the service provider space, data is measured in bandwidth(the total data flowing back and forth between network elements) and is impossible to be completely accurate about how many sessions and records a given amount of bandwidth corresponds to.

Note also that the the pool of data used in a test environment is smaller than in a production environment. With a smaller amount of data, the loss of a few packets is more obvious because each packet represents a higher percentage of the total.

Q: Why did I lose data with an outage?

A: If the connection between the Collector node and is down long enough that the relevant bins are closed, packets are dropped.

Q: Why is a "no data" error displayed in response to my request?

A: If a UI request arrives at the Caching node when it is pre-fetching data for its cache from the Insta node, it cannot respond immediately. The initial pre-fetch begins when the UI is opened and takes 10 to 30 minutes. You might also experience a long delay your request requires flushing the cache and rebuilding. For example, if you enter a year-long query, the system must flush the existing cache and start fetching the data needed to build the relevant cache.

Q: Why are bins configured for 5-minute intervals; can I have a more granular view?

A: Smaller bins can be exponentially more processor intensive, as well as increasing potential data loss (because packets coming in that are outside of the time span of the two open bins are dropped). The binning level is optimized to balance the need for a granular level view with prompt access to data.

Q: What start times should I specify when scheduling jobs?

Schedule your jobs to begin only after the time for the last bin is past. For example, for a job dealing with data for a given hour, do not schedule the job to start until five minutes after the end of the hour, because for the next five minutes after the end of the hour, the previous bin is still open.

Q: How do I know if a bin is closed?

A: When a _DONE file is written to the directory.

Q: Why can't I view 5-minute granularity on data from last quarter?

A: The Reflex platform uses a 5-minute granularity in order to optimize the speed at which queries can be processed, as well as storage space and costs. The highest level of granularity is retained for only the most recent data. For example, the Insta node might store 5-minute cubes for the past 5 days. Each day it aggregates the day's worth of 5-minute cubes from 6 days ago into a single 24-hour cube. Under this scheme, data is available at 5-minute granularity for the past 5 days, and at 1-day granularity for days before that. These aggregated sets can be stored for a period of time, usually a year or several years, before being purged from the database. The amount of storage time is primarily determined by a business decision of the amount of resources to devote to storage space. This optimization removes some of the granularity, but exponentially increases the speed at which queries can be processed, in addition to allow a much longer storage time for historical comparisons.

Understanding MURAL System Basics

System Basics

Logging Onto a Node

To securely access remote nodes, use the Secure Shell (SSH) protocol.

To log in as admin to a node:

```
ssh admin@<nodename/nodeIP>
```

Accessing and Using the CLI

On analytics appliances, you can execute commands from the following shells:

- **CLI**—Command Line Interface. This is a limited and specific appliance command-line interface. (Does not allow Unix commands or file system access.)
  To view available commands at this level, type a `?`.
- **PMX** & **subshell**—You use PMX to get to one of several subshells. When in a subshell, you can only execute commands appropriate to that subshell.
  To view available commands at this level, type a `?` and press ENTER.
- **Shell**—Provides access to the services level of the operating system.
  There is no keystroke method to view available commands at this level.

CLI Modes

Upon initial log into a node as admin, you are in the standard default mode, with limited commands. This is an appliance and users are not expected to be doing much in the way of configurations. From this mode, you can move to one of the other modes, if you have permissions to do so.

The CLI for Analytics nodes has 4 modes:

- **Default** (Standard) **mode** (prompt is `myhost >`)—This is the initial mode when you access a node and contains commands only to query a restricted set of state information. From this mode, you cannot take any actions that would affect the system or change any configuration. Type `en` or `enable` to switch to a mode with privileges.

- **Enable mode** (prompt is `host #`)—Contains a superset of the commands available in the standard mode, including commands to view all state information and take certain kinds of actions, such as rebooting the system. Includes many commands to start/stop control parameters; does not allow any configuration to be changed.
- **Configure mode** (prompt is `myhost (config)#`)—Super powerful mode which contains a full unrestricted set of commands to view anything, take any action, or change any configuration. Commands in the configure mode are a superset of those in enable mode.
- **Shell**—The unix regular terminal (or bash shell) used for providing OS-level access only to users with admin privileges. Use the `_shell` command to drop into the shell mode. From a unix shell, you can log out using the command `exit` or get back into the CLI mode using the command `cli`.

Changing Modes

To go from one mode to another, such as going from:

- Default to enable mode, type `enable`.
- Enable to shell mode, type `_shell`.
- Enable to configure mode, type `configure terminal`.
- Shell to configure mode, type `cli -m config`.
- From each CLI level you can back out by using the command `exit`.

Example 1: Configuring Host System

To configure a system called host:

1. Run the following commands:

```
ssh admin@host> enable
host# conf t
host(config)#
```

2. From the `host(config)#` prompt, enter the commands for configuration

change that you want to make.

3. If the system encounters an error in executing the command, the response begins with **%** followed by text describing the error. Commands that succeed will not print a response.

Example 2: Changing to Shell Prompt

As another example, run the following commands to log into the system and change to the shell prompt:

```
host [cluster : master]> en
host [cluster : master]# _shell
[admin@host ~]# cli -m config
```

The resulting output may resemble:

```
Cluster ID:          COL-CLUSTER1
Cluster name:        COL1-CLUST
Management IP:       192.168.100.73/24
Cluster master IF:   eth0
Cluster node count:  2
Local name:          COL1-100-2
Local role:          master
Local state:         online
Master address:      192.168.100.2 (ext) 192.168.100.2 (int)
Master state:        online
```

PMX Subshell

From within the configuration mode, you can access a subshell called PMX. The prompt is characterized by `pm extension`. This allows specific subsystem level configuration, restricting you to the environment of the subsystem. For example:

- subshell hadoop yarn—Limited to and provides YARN level configuration
- subshell oozie—Limited to and provides Oozie level configuration
- subshell bulkstat
- subshell healthcheck—Used to inquire system health
- subshell iscsi—Used to perform tasks related to storage

- subshell patch—Used to apply patches and debug patch history
- subshell anomaly

Go to any one of these subshells using:

```
> en
# conf t
(config)# pmx subshell subshell-name
pm extension (subshell-name)>
```

You can back out of any subshell level using the command `quit`.

Creating and Viewing User Accounts

The system by default initially has 2 accounts:

- **admin**—Full privileges to do anything on the system
- **monitor**—Privileges to read almost anything on the system, perform some actions, but not modify any configuration.

You can also create additional accounts. Admin and monitor are also names of the privilege levels which can be given to user accounts. Admin privileges allow a user full privileges to do anything on the system.

The most basic user level only grants a user permission to log in, view whether the system is up, and view help.

Monitoring Account Commands

The following commands are used when monitoring accounts:

Configuring User Account Commands

The following commands, executed from the configure mode (`hostname (config)#`) apply to user accounts:

Command	Description
username *userid*	Create a new user account.
username *userid* password	Set password for a user.
username *userid* disable	Disable a user account.
no username *userid*	Delete a user account.

Command	Description
show users	View a list of users currently logged in.
show whoami	View information about the current user.
show usernames	View information about all configured users.
set username privilege admin|monitor|unpriv	Set privileges for a user account.Three privilege levels are available—admin, monitor and unpriv.

Creating a Read-Only Account

To create an account which allows read-only privileges on the system, run the following commands:

```
no username loginname disable
username loginname capability monitor
username loginname password password
```

Creating a New User

To create a new user from the user interface:

1. Log in to the application user interface, click on the **Configure** tab.
2. Click **Create User**. A pop-up window opens.
3. Enter the user account details (name, password, email ID) and select the type of user.

   Password should contain at least 8 characters, including 1 uppercase letter.
4. Verify the account by logging out and logging in using the new credentials. Verify access is available to only the pages to which you want the user to have access.

Verifying NTP Synchronization

Network time protocol (NTP) is used to synchronize clocks over a network. NTP is enabled by default on nodes.

Command	Description
ntp enable	enable NTP on a node
ntp disable	disable NTP on a node
show ntp	verify NTP status
show clock	show the current clock time

Creating a Backup of the System Configuration

Backing-up the system configuration enables you to re-populate system information on an alternate node if a node failure occurs.

For each node:

1. SSH to the node and log in as admin:

   ssh admin@node1

2. Run the following commands:

   ```
   > enable
   # conf t
   # configuration write to file audit-backup no-switch
   ```

   Where `audit-backup` is a name you assign to the file and the `no-switch` option means the system will not switch over to use the backup

3. Press Ctrl-D to log out.

   Note: To copy the backup file on a different machine, perform an SCP from the `_shell` mode.

4. To view the configuration files that are stored, run the command:

   ```
   show configuration files
   ```

System Logs

This topic explains how to enable system logging on a MURAL node.

Logging System Default Settings

The following defaults apply to all MURAL software processes that use standard system logging:

- File location—Files named in the `/var/log` directory
- Logging rotation—Daily and 7 is the default number of logging files
- Logging level—info

Note: Do not change the defaults except as directed by Technical Support. For information about the specific commands, see "Basic CLI Commands" on page 46.

Logging messages are categorized at the following levels:

Level name	Description
alert	Action must be taken immediately
crit	Critical conditions
debug	Debug-level messages
emerg	Emergency: system is unusable
err	Error conditions
info	Informational messages
none	Disable logging
notice	Normal but significant condition
warning	Warning conditions

Viewing System Logs

System logs contain important information that you can use to verify the health of the system and troubleshoot the system. You might need to capture logs to send to Technical Support.

You can use filters to find information about a variety of conditions.

To view system logs, run the following commands:

```
> ssh user@node1
admin@host> enable
admin@host# show log [continuous | continuous matching string]
```

Collecting Log Files

The following sections describe how to collect log files.

Creating a Directory to Hold Log Files

Before you collect log files, create a directory on the server from which you will collect the logs:

```
admin@host# mkdir -p /data/Log_dump
```

You will copy all logs to this directory.

Collecting Logs from Master Collector Node

To collect log files from the master Collector node:

```
admin@host# cp /var/log/messages* /data/Log_dump
admin@host# cp /data/hadoop_logs/hadoop-admin-namenode* /data/Log_
dump
admin@host# cp /data/hadoop_logs/hadoop-root-journalnode*
/data/Log_dump

admin@host# cp /data/hadoop_logs/yarn-admin-resourcemanager*
/data/Log_dump
admin@host# cp /data/hadoop_logs/mapred-admin-historyserver*
/data/Log_dump
admin@host# cp /data/oozie_logs/oozie.log* /data/Log_dump
```

Collecting Hadoop Job-Specific Logs

To identify the job ID:

1. Login and go to the pmx oozie subshell:

```
> en
# conf t
(config)# pmx subshell subshell-name
```

Note: Job ID will usually show as being on the master namenode. If you dont see job ID on the current master namenode then check the standby namenode as the master node may have been changed to a standby.

>>>>>>> cb876e717e8267ac6ead3e592adc5cd468ac8cd8

2. Show running jobs:

```
pm extension (oozie)> show workflow RUNNING jobs
```

3. Capture the files for the specific job, replacing `JOB-ID` with the specific job number (the job ID is the first string in each line against the job's name):

```
# cp -R /data/oozie-admi/JOB-ID/ /data/Log_dump
```

Collecting Logs from Standby Collector Node

To collect log files from the standby Collector node:

```
# cp /var/log/messages* /data/Log_dump/
# cp / data/hadoop_logs/hadoop-admin-namenode* /data/Log_dump
# cp /data/hadoop_logs/hadoop-root-journalnode* /data/Log_dump

# cp /data/hadoop_logs/yarn-admin-resourcemanager* /data/Log_dump
# cp /data/hadoop_logs/mapred-admin-historyserver* /data/Log_dump
# cp /data/oozie_logs/oozie.log* /data/Log_dump
```

Collecting Logs from a Compute Node

To collect log files from a Compute (Data) node, copy the files from the location below for each data node individually:

```
# cp /var/log/messages* /data/Log_dump
# cp /data/hadoop_logs/hadoop-root-datanode* /data/Log_dump
# cp /data/hadoop_logs/yarn-root-nodemanager /data/Log_dump
```

Collecting Logs from a Caching Compute (iNSTA) Node

- To collect Insta log files:

```
# cp /var/log/Calpont/*.log /data/Log_dump/
```

- To collect message log files:

```
# cp /var/log/messages* /data/Log_dump/
```

- To collect any core files (copy only the latest ones):

```
# cp /data/insta/infinidb/core.* /data/Log_dump/
# cp /var/opt/tms/snapshots/* /data/Log_dump/
```

Collecting Logs from a UI (Rubix) Node

- To collect logs RGE logs:

```
# mkdir -p /data/Log_dump/rge
# cp /data/instances/<rg-app-name>/<rge instance
id>/bin/rge.log/data/instances/rge/1/bin/rge*.log /data/Log_
dump/rge/
```

- To collect bulkstats logs:

```
# mkdir -p /data/Log_dump/bs
# cp /data/instances/rge/1/bin/rge*.log /data/Log_dump/bs
```

- To collect EDR logs on a master UI node:

```
# mkdir -p /data/Log_dump/edr1
# cp /data/instances/atlas/1/bin/rubix*.log /data/Log_
dump/edr1/
```

- To collect CacheLess Rubix Logs:

```
# mkdir /data/Log_dump/edr_cacheless/
# cp /data/instances/<rubix-app-name>/<rubix instance
id>/bin/rubix.log/data/instances/reportAtlas/1/bin/rubix*.log
/data/Log_dump/edr_cacheless/
```

- To collect Launcher logs if the Launcher is enabled:

```
# mkdir -p /data/Log_dump/launcher
# cp /data/instances/launcher/1/bin/rubix*.log* /data/Log_
dump/launcher
```

- To collect HET logs if HET is enabled:

```
# mkdir -p /data/Log_dump/het
# cp /data/instances/httperror/1/bin/rubix*.log* /data/Log_
dump/het
```

- To collect Anomaly logs (only if either Anomaly or Bulkstat application is enabled):

```
# mkdir -p /data/Log_dump/ruleEngine
# cp /data/instances/ruleEngine/1/bin/anomaly*.log* /data/Log_
dump/ruleEngine
```

- To collect FLEX logs on a UI node:

```
# cp /data/instances/atlas/1/bin/flex*.log /data/Log_dump/
```

- To collect Catalina logs on a UI node:

```
# cp /data/instances/<rge-app-name>/<rge-instance-
id>/logs/catalina.out
# cp /data/instances/<rubix-app-name>/<rubix-instance-
id>/logs/catalina.out
# cp /data/instances/atlas/1/logs/catalina*.out /data/Log_
dump/edr1
# cp /data/instances/bulkstats/1/logs/catalina*.out /data/Log_
dump/bs
# cp /data/instances/rge/1/logs/catalina*.out /data/Log_
dump/rge
```

If Launcher is enabled on the the UI nodes:

```
# cp /data/instances/launcher/1/logs/catalina.*out /data/Log_
dump/launcher
```

If the HET application is enabled:

```
# cp /data/instances/httperror/1/logs/catalina.*out /data/Log_
dump/het
```

If the Anomaly application is enabled:

```
# cp /data/instances/ruleEngine/1/logs/catalina.*out
/data/Log_dump/ruleEngine
```

Finalizing and Delivering Log Files

1. Capture the system log files:

```
> en
# conf t
(config)# debug generate dump detail
```

The resulting output may resemble:

```
Debug dump generated successfully. Script output:
SYSINFO=/var/opt/tms/sysdumps//sysinfo-sysdump-BRCISCO-COL-
151-79-20130405-104001.txt
SYSDUMP=/var/opt/tms/sysdumps//sysdump-BRCISCO-COL-151-79-
20130405-104001.tgz
```

2. Copy these files to the log collection directory:

```
(config)# _shell
# cp /var/opt/tms/sysdumps//sysinfo-sysdump-COL-151-79-
20130405-104001.txt /data/Log_dump/
# cp /var/opt/tms/sysdumps//sysdump-COL-151-79-20130405-
104001.tgz /data/Log_dump/
```

3. Tar and gzip the `/data/Log_dump/` directory and upload or secure copy (SCP) the tar file to the appropriate FTP server:

```
# cd /data/
# tar -zcvf Log_dump_module-name_role-name.tar.gz Log_dump
```

Where the naming structure for the tar or gzip file contains information about what is being zipped:

- module-name
- role-name

For examples, see the following commands:

```
# tar -zcvf Log_dump_collector_master.tar.gz Log_dump
# tar -zcvf Log_dump_collector_standby.tar.gz Log_dump
# tar -zcvf Log_dump_compute_1.tar.gz Log_dump
# tar -zcvf Log_dump_compute_2.tar.gz Log_dump
# tar -zcvf Log_dump_insta_master.tar.gz Log_dump
# tar -zcvf Log_dump_insta_standby.tar.gz Log_dump
# tar -zcvf Log_dump_ui_master.tar.gz Log_dump
# tar -zcvf Log_dump_ui_standby.tar.gz Log_dump
```

Collection of Logs using Centralized Log Management

Prerequisite

Centralized Log Management feature is by default enabled while installing nodes via GMS.

- To disable log collection:

  ```
  (config)# no logcollection enable
  ```

  Note: If log collection is disabled and you want to activate an alternate configuration, the log collection will not be enabled automatically. You need to re-enable log collection.

- To re-enable log collection:

  ```
  (config)# logcollection enable
  ```

The CLI of **logCollection** needs to be called manually on all the GMS nodes (master as well as standby).

Procedure

The directory which keeps the logs is mounted on an external storage with space allocated as ~700 GB to 1 TB depending upon cardinality of data pushed. Logs are automatically mounted on the SAN during installation without running any specific commands.

While preparing XML on GMS node, you need to specify the directory where to keep the logs (`/data/central/logs`). This will be a different storage (LUN) than where logs get collected.

Note: Mounting is done automatically while you run GMS.

To specify `/data/central/logs` as the directory where to keep the logs:

1. Log into the VIP of the GMS cluster and navigate to the following directory to collect the logs:

```
> en
# _shell
# cd /data/central/logs
```

Note: Log messages from each node, including `/var/log/messages`, depend upon the application running on that node being collected on the GMS. For example, all hadoop logs in the `/data/hadoop_logs` directory on the Collector node also get copied to GMS under the parent directory which is made by the hostname of that node, like `Collector-BRUCS-2`.

2. Run an `ls` command to list directories which contain the logs of nodes configured by GMS:

```
# ls
BS-RGE-UCS-1     Collector-UCS-2  EDR-Rubix-1  GMS-UCS-1
Insta-UCS-2
BS-RGE-UCS-2     Compute-UCS-1    EDR-Rubix-2  GMS-UCS-2
Rubix-UCS-1
```

3. Configure the total space of all logs from a particular host such that it does not exceed 100 GB. Configure the retention period of these logs upto 30 days. Files beyond these bounds are deleted.

   Note: Do not alter these configurations except as directed by Technical Support. For information on specific commands, see "Basic CLI Commands" on page 46.

4. Logs are synced between both master and standby GMS nodes automatically.

   This list represent the directories created under `/data/central/logs` by the hostname of each node whose logs are getting collected

```
Collector-UCS-1  Compute-UCS-2    EDR-Rubix-3  Insta-UCS-1
Rubix-UCS-2
```

Starting and Stopping Jobs

You can start or stop all jobs, or any specific job.

Starting or Stopping All Jobs

When starting or stopping all jobs, the commands are the same except for the final command line where you need to specify `run` or `stop` as needed.

1. Log into the node, invoke a new CLI shell before going to the oozie subshell:

```
host [cluster : master]> en
host [cluster : master]# configure terminal
[host [cluster : master](config)# pmx subshell oozie
```

2. Run the following command:

```
pm extension (oozie)# action job all
```

   Where `action` is replaced with:

   - `run` to start all jobs
   - `stop` to stop all jobs

Stopping and Restarting Specific Jobs

You can stop a job and re-run it from a specified timestamp. To display the running jobs, execute the `show workflow RUNNING jobs` command in the oozie subshell. In this section, the EDRMaster job is used as an example.

Caution: Do not alter the configurations of jobs that are currently running except as directed by Technical Support.

1. Log into NameNode (Collector node), go to `_shell` and invoke a new CLI shell before going to the oozie subshell:

```
host [cluster : master]> en
host [cluster : master]# _shell
[admin@host ~]# cli -m config
host [cluster : master](config)# pmx subshell oozie
```

2. Stop the MasterEDR job:

   ```
   pm extension (oozie)> stop jobname MasterEDR
   ```

   The resulting output may resemble:

   ```
   Killing job MasterEDR : 0000615-121101135430892-oozie-admi-C
   ```

3. In order to stop and re-run the MasterEDR job you must also stop and start the CubeExporter job:

   ```
   pm extension (oozie)> stop jobname CubeExporter
   ```

   The resulting output may resemble:

   ```
   Killing job CubeExporter: 0000617-121101135430892-oozie-admi-C
   ```

4. Start the MasterEDR job:

   a. Run the following command:

      ```
      pm extension (oozie)> set job job-name attribute jobStart
      new-time
      ```

      The resulting output may resemble:

      ```
      setting attribute (jobStart)
      ```

      For example:

      ```
      pm extension (oozie)> set job MasterEDR attribute
      jobStart 2012-11-05T03:00Z
      setting attribute (jobStart)
      pm extension (oozie)> set job CubeExporter attribute
      jobStart 2012-11-05T03:00Z
      setting attribute (jobStart)
      ```

      Note: Ensure that you specify a value for `jobStart`. In its absence, the job fails and the "cannot concatenate 'str' and 'NoneType' objects" error message is received.

b. Run the following command to start both the `MasterEDR` and `CubeExporter` jobs:

```
pm extension (oozie)> run job job-name
```

The resulting output may resemble:

```
writing file /data/configs/oozie_conf/job-
name/job.properties
writing file /data/configs/oozie_conf/job-
name/coordinator.tmp
writing file /data/configs/oozie_conf/job-
name/workflow.xml
Deleted hdfs://BRMUR-COL-CLUST:9000/oozie/job-name
job: job-id-number-oozie-admi-C
```

c. `Quit` until out of the oozie and pmx subshells:

```
pm extension (oozie)> quit
host [cluster : master] (config)# quit
```

d. Validate the jobs are running properly. Use the job ID provided in the output from the steps listed above for the MasterEDR job and then run:

```
[admin@host ~]# cd /data/oozie-admi/job-id-number-oozie-
admi-C/basecubejob--ssh/
[admin@host ~]# tail -f 16695.job-id-number-oozie-admi-
W@basecubejob@0.stdout
```

For example, if `job-id-number` is `0000632-121101135430892`:

```
[admin@host ~]# cd /data/oozie-admi/0000632-
121101135430892-oozie-admi-C/basecubejob--ssh/
[admin@host ~]# tail -f 16695.0000632-121101135430892-
oozie-admi-W@basecubejob@0.stdout
```

A similar update can be done for other jobs also if required. Set the new time from which you want to run the job.

Starting Oozie Jobs

1. Log in to the master Collector (Name) node and run the following commands:

```
[admin@Collector ~]# cli -m config
(config) # pmx subshell oozie
pm extension (oozie)> run job master_job
```

Stopping Oozie Jobs

1. Log in to the master Collector (Name) node and run the following commands:

```
[admin@Collector ~]# cli -m config
(config) # pmx subshell oozie
pm extension (oozie)> stop jobname master_job
```

Viewing Oozie Job Configuration

1. Run the following commands:

```
[admin@Collector ~]# cli -m config
(config) # pmx subshell oozie
pm extension (oozie)> show config job master_job
pm extension (oozie)> show config dataset rStats_output
```

Basic CLI Commands

Note: You can execute these commands in the configure mode (`configure terminal`).

Network Interfaces

Set IP Address/ Clear IP Address

```
interface <ifname> ip address <IP address> <netmask>
```

```
no interface <ifname> ip address
```

Enable or disable use of DHCP on the specified interface.

```
[no] interface <ifname> dhcp
```

Network Bonding

Create or Delete a bonded interface

```
bond <bonded-if> [mode <string>] [link-mon-time
<milliseconds>] [up-delay-time <milliseconds>] [down-
delay-time <milliseconds>]
```

```
no bond <bonded-if>
```

- `<bonded-if>` --- Name of bonded interface.
- `mode` --- Mode can be one of the following : balance-rr, backup, balance-xor,balance-xor-layer3+4, broadcast, link-agg, link-agg-layer3+4, balance-
- `link-mon-time` --- Link monitoring time

Add or Remove an interface from a specified bonded interface

```
interface <ifname> bond <bonded-if>
```

```
no interface <ifname> bond <bonded-if>
```

Display bonded interface configuration information

```
show bonds
```

```
show bonds <bonded-if>
```

Name resolution and Routing

```
[no] ip name-server <IP address>
```

```
[no] ip domain-list <domain>
```

```
ip route <network prefix> <network mask> <next hop IP
address or Interface>
```

```
no ip route <network prefix> <network mask> [<next hop IP
address>]
```

Set or clear the system hostname

```
hostname <hostname>
```

```
[no] hostname
```

Set or remove the default route

```
ip default-gateway <next hop IP address or Interface>
[<Interface>]
```

```
no ip default-gateway
```

Display the currently configured routing table

```
show routes
```

Display the currently configured host mapping

```
show hosts
```

Configuring and Viewing Event Logs

Set Logging Levels

```
logging local <log level>
```

Set Log rotation

```
logging files rotation criteria frequency
<daily/weekly/monthly>
```

```
logging files rotation max-num <max number of files to
keep>
```

Viewing Logs

```
show log
```

Or

```
show log continuous
```

Configuring SSH

Configuring the SSH server

```
[no] ssh server enable
```

Configuring the SSH client--Generate SSH keys

```
ssh client user <username> identity <key-type> generate
```

Add a user to authorized list

```
ssh client user <username> authorized-key sshv2 "<key>"
```

Remove a user from authorized list

```
no ssh user <username> authorized-key sshv2 "<key>"
```

Disable host-key check globally

```
no ssh client global host-key-check
```

Display SSH Client information

```
show ssh client
```

Configuring NTP, Clock, and Time zones

Add a NTP server or Peer

```
ntp server <IP address>

ntp peer <IP address>
```

Remove a NTP server or Peer

```
no ntp server <IP address>

no ntp peer <IP address>
```

Update from specific NTP server

```
ntpdate <IP address>
```

Show NTP settings

```
show ntp
```

Set system clock

```
clock set hh:mm:ss [<yyyy/mm/dd>]
```

Set system Zone

```
clock timezone <zone>
```

Display current Time, Date and Zone

```
show clock
```

Miscellaneous Configuration

Configuring System Banner

```
banner login <string>
```

Saving Configuration

```
write memory
```

Or

```
configuration write
```

Saving configuration to a file

```
configuration write to <name> [no-switch]
```

Restoring configuration from a saved file

```
configuration switch-to <name>
```

Exporting saved configuration

```
configuration upload <name> <URL>

configuration upload active <URL>
```

- First command uploads a `<named>` file, while second command uploads the currently active conf file.
- URL - can be of the type `scp://ad-min:password@hostname/config/db/<filename>`

Displaying Software Versions and Images

Display current software version

```
show version
```

Display all available images

```
show images
```

Monitoring Health of the Filesystem

Run the following command in CLI mode:

```
# hdfs fsck <dir path>
```

For example:

```
[admin@host ~]# hdfs fsck /data | tail -20
WARNING: org.apache.hadoop.metrics.jvm.EventCounter is deprecated.
Please use org.apache.hadoop.log.metrics.EventCounter in all the
log4j.properties files.

..................................................................
................................

..................................................................
.Status: HEALTHY
 Total size: 11658194365171 B (Total open files size: 2684354560 B)
 Total dirs: 68199
 Total files: 238068 (Files currently being written: 193)
 Total blocks (validated): 230599 (avg. block size 50556135 B)
(Total open file blocks (not validated): 130)
 Minimally replicated blocks: 230599 (100.0 %)
 Over-replicated blocks: 0 (0.0 %)
 Under-replicated blocks: 0 (0.0 %)
 Mis-replicated blocks:  0 (0.0 %)
 Default replication factor: 2
 Average block replication: 2.0000043
 Corrupt blocks:  0
 Missing replicas:  0 (0.0 %)
 Number of data-nodes:  7
 Number of racks:  1
FSCK ended at Tue Jul 14 09:54:51 GMT 2015 in 1575 milliseconds
```

```
The filesystem under path '/data' is HEALTHY
[admin@host ~]#
```

Note the attributes that are emphasized in the preceding sample. `HEALTHY` indicates that the filesystem is configured as required.

Monitoring Oozie Jobs

Verifying the Status of Oozie Jobs from CLI

1. Log into the NameNode and go to the pmx oozie subshell:

```
> en
# conf t
(config)# pmx subshell subshell-name
```

2. Show the workflow of all jobs:

```
pm extension (oozie)> show workflow pm extension (oozie)> show
workflow all jobs
```

   The resulting output may resemble:

```
pm extension (oozie)> show workflow
FAILED     KILLED     PREP       RUNNING    SUCCEEDED  all  pm
extension (oozie)> show workflow all jobs
Job ID  App Name    App Path    Console URL User    Group
Run Created Started Status  Last Modified   Ended
--------------------------------------------------------------
0003006-121209074445646-oozie-admi-W    CubeExporter    -
http://CISCO-COL1-147-11:8080/oozie?job=0003006-
121209074445646-oozie-admi-W    admin   users   2012-12-12
06:15    2012-12-12 06:15    RUNNING 2012-12-12 06:15    -
--------------------------------------------------------------
...
```

   And so on, for all oozie jobs, regardless of their status.

3. The status is the third value from the end. In the above example, the value is shown as `RUNNING`. Therefore we know that the CubeExporter job is running.

Look through the output to see which jobs are running, failed, or killed. If you would rather look at jobs of only one status, you can run any of the following three commands.

Identify all Running Jobs

```
pm extension (oozie)> show workflow RUNNING jobs
```

Identify all Failed Jobs

```
pm extension (oozie)> show workflow FAILED jobs
```

Identify all Killed Jobs

```
pm extension (oozie)> show workflow KILLED jobs
```

Monitoring an Oozie Job from the CLI

You can view job status and configuration of specific oozie jobs. First ensure you are in the `pmx oozie subshell` of the Collector node. See "Changing Modes" on page 28 for instructions on entering and switching between subshells.

1. Log into the pmx oozie subshell on the Collector node:

   ```
   > en
   # conf t
   (config)# pmx subshell subshell-name
   ```

2. Run the show job command with the `jobID` of the job you want to review:

   ```
   pm extension (oozie)> show job 0001577-120917211811867-oozie-
   admi-W
   ```

Note: You can also access the job logs from the file system under the `/data/oozie-admi` folder. There is one folder for each job.

Monitoring Computed Data

Cube Data in HDFS

Processed cubes and other data is stored in HDFS right after the records have been processed. The cube exporter job then takes the data from HDFS and into the Insta database.

```
[admin@host checkdone--ssh]# hdfs dfs -ls /data/output
```

The resulting output may resemble:

```
Found 13 items
drwxr-xr-x   - admin supergroup          0 2015-12-02 12:09
/data/output/AtlasRollupCubes
drwxr-xr-x   - admin supergroup          0 2015-12-02 12:01
/data/output/AtlasSubDevs
drwxr-xr-x   - admin supergroup          0 2015-12-02 12:01
/data/output/AtlasSubcrBytes
drwxr-xr-x   - admin supergroup          0 2015-12-02 12:08
/data/output/AtlasSubscriberDeviceMPH
drwxr-xr-x   - admin supergroup          0 2015-12-04 09:58
/data/output/BytesAggWeekly
drwxr-xr-x   - admin supergroup          0 2015-12-03 12:43
/data/output/CliMonthlyBytes
drwxr-xr-x   - admin supergroup          0 2015-12-03 12:46
/data/output/CliSegments
drwxr-xr-x   - admin supergroup          0 2015-12-03 12:47
/data/output/CliSegmentsMPH
drwxr-xr-x   - admin supergroup          0 2015-12-02 11:34
/data/output/CoreJobCubes
drwxr-xr-x   - admin supergroup          0 2015-12-03 17:07
/data/output/EDCGenerator
drwxr-xr-x   - admin supergroup          0 2015-12-02 10:59
/data/output/SubscriberIBCreator
```

```
drwxr-xr-x   - admin supergroup          0 2015-12-02 12:01
/data/output/TopN
drwxr-xr-x   - admin supergroup          0 2015-12-04 10:12
/data/output/TopSubcribers
```

```
Found 13 items
drwxr-xr-x   - admin supergroup          0 2012-09-18 17:11
/data/output/AtlasBaseCubes
drwxr-xr-x   - admin supergroup          0 2012-09-18 17:14
/data/output/AtlasRollupCubes
drwxr-xr-x   - admin supergroup          0 2012-09-18 17:13
/data/output/AtlasSubDevs
drwxr-xr-x   - admin supergroup          0 2012-09-18 17:13
/data/output/AtlasSubcrBytes
drwxr-xr-x   - admin supergroup          0 2012-09-25 00:00
/data/output/BytesAggWeekly
...
```

When cubes are calculated, they are stored in the appropriate folder. Folder names are created based on the timestamp of the bin. For example, this command lists cubes from one folder name:

```
[admin@host checkdone--ssh]# hdfs dfs -ls
/data/output/AtlasBaseCubes/2012/10/09/18
```

The resulting output may resemble:

```
Found 6 items
-rw-r--r--   2 admin supergroup   29386988 2012-10-09 18:13
/data/output/AtlasBaseCubes/2012/10/09/18/X.MAPREDUCE.0.0
-rw-r--r--   2 admin supergroup   29383694 2012-10-09 18:13
/data/output/AtlasBaseCubes/2012/10/09/18/X.MAPREDUCE.0.1
-rw-r--r--   2 admin supergroup   29383311 2012-10-09 18:13
/data/output/AtlasBaseCubes/2012/10/09/18/X.MAPREDUCE.0.2
-rw-r--r--   2 admin supergroup   29492142 2012-10-09 18:13
/data/output/AtlasBaseCubes/2012/10/09/18/X.MAPREDUCE.0.3
-rw-r--r--   2 admin supergroup          0 2012-10-09 18:13
```

```
/data/output/AtlasBaseCubes/2012/10/09/18/_DONE
-rw-r--r--   2 admin supergroup          0 2012-10-09 18:13
/data/output/AtlasBaseCubes/2012/10/09/18/_SUCCESS
```

Verifying HDFS Status

You can use the `hdfsdfsadmin` command provided by hadoop to view the status of the HDFS file system. This command reports information regarding the overall health of the file system, its utilization, and the status of the various data nodes that are part of the cluster:

```
[admin@host ~]# hdfs dfsadmin -report
```

The resulting output may resemble:

```
Configured Capacity: 3837463375872 (3.49 TB)
Present Capacity: 3641500550251 (3.31 TB)
DFS Remaining: 3533955174400 (3.21 TB)
DFS Used: 107545375851 (100.16 GB)
DFS Used%: 2.95%
Under replicated blocks: 0
Blocks with corrupt replicas: 0
Missing blocks: 0
-------------------------------------------------
Datanodes available: 4 (4 total, 0 dead)
Name: 10.84.35.151:50010
Decommission Status : Normal
Configured Capacity: 1082254041088 (1007.93 GB)
DFS Used: 27310859553 (25.44 GB)
Non DFS Used: 55276739295 (51.48 GB)
DFS Remaining: 999666442240(931.01 GB)
DFS Used%: 2.52%
DFS Remaining%: 92.37%
Last contact: Tue Oct 09 20:12:50 GMT 2012
Name: 10.84.35.152:50010
Decommission Status : Normal
```

```
Configured Capacity: 1082245816320 (1007.92 GB)
DFS Used: 27098452545 (25.24 GB)
Non DFS Used: 55222156735 (51.43 GB)
DFS Remaining: 999925207040(931.25 GB)
DFS Used%: 2.5%
DFS Remaining%: 92.39%
Last contact: Tue Oct 09 20:12:50 GMT 2012
...
```

Benchmarking the HDFS I/O

```
[admin@host ~]# hdfs jar/opt/hadoop/share/hadoop/mapreduce/hadoop-
mapreduce-client-jobclient-2.4.0-tests.jar TestDFSIO -write -
nrFiles 10 -size
```

The resulting output may resemble:

```
15/03/17 10:00:16 INFO fs.TestDFSIO: TestDFSIO.1.7
15/03/17 10:00:16 INFO fs.TestDFSIO: nrFiles = 10
15/03/17 10:00:16 INFO fs.TestDFSIO: nrBytes (MB) = 1000.0
15/03/17 10:00:16 INFO fs.TestDFSIO: bufferSize = 1000000
15/03/17 10:00:16 INFO fs.TestDFSIO: baseDir =
/benchmarks/TestDFSIO

15/03/17 10:00:17 INFO fs.TestDFSIO: creating control file:
1048576000 bytes, 10 files
15/03/17 10:00:18 INFO fs.TestDFSIO: created control files for: 10
files
15/03/17 10:00:18 INFO client.RMProxy: Connecting to
ResourceManager at /192.168.193.226:9001
15/03/17 10:00:18 INFO client.RMProxy: Connecting to
ResourceManager at /192.168.193.226:9001
15/03/17 10:00:19 INFO mapred.FileInputFormat: Total input paths to
process : 10
15/03/17 10:00:19 INFO mapreduce.JobSubmitter: number of splits:10
```

```
15/03/17 10:00:20 INFO mapreduce.JobSubmitter: Submitting tokens
for job: job_1426519537786_0001
15/03/17 10:00:20 INFO impl.YarnClientImpl: Submitted application
application_1426519537786_0001
15/03/17 10:00:20 INFO mapreduce.Job: The url to track the job:
http://Namenode-VIP:8088/proxy/application_1426519537786_0001/
15/03/17 10:00:20 INFO mapreduce.Job: Running job: job_
1426519537786_0001
15/03/17 10:00:29 INFO mapreduce.Job: Job job_1426519537786_0001
running in uber mode : false
15/03/17 10:00:29 INFO mapreduce.Job:  map 0% reduce 0%
15/03/17 10:00:51 INFO mapreduce.Job:  map 7% reduce 0%
15/03/17 10:00:52 INFO mapreduce.Job:  map 13% reduce 0%
15/03/17 10:00:53 INFO mapreduce.Job:  map 20% reduce 0%
15/03/17 10:00:54 INFO mapreduce.Job:  map 27% reduce 0%
15/03/17 10:00:55 INFO mapreduce.Job:  map 33% reduce 0%
15/03/17 10:00:56 INFO mapreduce.Job:  map 40% reduce 0%
15/03/17 10:00:59 INFO mapreduce.Job:  map 60% reduce 0%
15/03/17 10:01:01 INFO mapreduce.Job:  map 67% reduce 0%
15/03/17 10:02:15 INFO mapreduce.Job:  map 70% reduce 0%
15/03/17 10:03:18 INFO mapreduce.Job:  map 70% reduce 3%
15/03/17 10:03:26 INFO mapreduce.Job:  map 73% reduce 3%
15/03/17 10:03:56 INFO mapreduce.Job:  map 73% reduce 7%
15/03/17 10:03:57 INFO mapreduce.Job:  map 77% reduce 7%
15/03/17 10:04:03 INFO mapreduce.Job:  map 77% reduce 10%
15/03/17 10:04:14 INFO mapreduce.Job:  map 80% reduce 10%
15/03/17 10:04:18 INFO mapreduce.Job:  map 80% reduce 13%
15/03/17 10:04:20 INFO mapreduce.Job:  map 83% reduce 13%
15/03/17 10:04:21 INFO mapreduce.Job:  map 83% reduce 17%
15/03/17 10:04:27 INFO mapreduce.Job:  map 87% reduce 17%
15/03/17 10:04:28 INFO mapreduce.Job:  map 90% reduce 17%
15/03/17 10:04:30 INFO mapreduce.Job:  map 90% reduce 23%
15/03/17 10:04:34 INFO mapreduce.Job:  map 93% reduce 23%
```

```
15/03/17 10:04:35 INFO mapreduce.Job:  map 97% reduce 23%
15/03/17 10:04:36 INFO mapreduce.Job:  map 97% reduce 30%
15/03/17 10:05:00 INFO mapreduce.Job:  map 100% reduce 67%
15/03/17 10:05:02 INFO mapreduce.Job:  map 100% reduce 100%
15/03/17 10:05:06 INFO mapreduce.Job: Job job_1426519537786_0001
completed successfully
15/03/17 10:05:07 INFO mapreduce.Job: Counters: 49
       File System Counters
               FILE: Number of bytes read=388
               FILE: Number of bytes written=1068381
               FILE: Number of read operations=0
               FILE: Number of large read operations=0
               FILE: Number of write operations=0
               HDFS: Number of bytes read=2350
               HDFS: Number of bytes written=10485760081
               HDFS: Number of read operations=43
               HDFS: Number of large read operations=0
               HDFS: Number of write operations=12
       Job Counters
               Launched map tasks=10
               Launched reduce tasks=1
               Data-local map tasks=10
               Total time spent by all maps in occupied slots (ms)=
               4086718
               Total time spent by all reduces in occupied slots
               (ms)=330992
               Total time spent by all map tasks (ms)=2043359
               Total time spent by all reduce tasks (ms)=165496
               Total vcore-seconds taken by all map tasks=2043359
               Total vcore-seconds taken by all reduce tasks=165496
               Total megabyte-seconds taken by all map tasks
               =6277198848
               Total megabyte-seconds taken by all reduce tasks
```

```
                =508403712
        Map-Reduce Framework
                Map input records=10
                Map output records=50
                Map output bytes=772
                Map output materialized bytes=1026
                Input split bytes=1230
                Combine input records=0
                Combine output records=0
                Reduce input groups=5
                Reduce shuffle bytes=1026
                Reduce input records=50
                Reduce output records=5
                Spilled Records=100
                Shuffled Maps =10
                Failed Shuffles=0
                Merged Map outputs=10
                GC time elapsed (ms)=1223
                CPU time spent (ms)=139870
                Physical memory (bytes) snapshot=11203444736
                Virtual memory (bytes) snapshot=30397915136
                Total committed heap usage (bytes)=16594239488
        Shuffle Errors
                BAD_ID=0
                CONNECTION=0
                IO_ERROR=0
                WRONG_LENGTH=0
                WRONG_MAP=0
                WRONG_REDUCE=0
        File Input Format Counters
                Bytes Read=1120
        File Output Format Counters
                Bytes Written=81
```

```
15/03/17 10:05:07 INFO fs.TestDFSIO: ----- TestDFSIO ----- : write
15/03/17 10:05:07 INFO fs.TestDFSIO:            Date & time: Tue
Mar 17 10:05:07 UTC 2015
15/03/17 10:05:07 INFO fs.TestDFSIO:        Number of files: 10
15/03/17 10:05:07 INFO fs.TestDFSIO: Total MBytes processed:
10000.0
15/03/17 10:05:07 INFO fs.TestDFSIO:      Throughput mb/sec:
5.105979714963788
15/03/17 10:05:07 INFO fs.TestDFSIO: Average IO rate mb/sec:
5.5383758544921875
15/03/17 10:05:07 INFO fs.TestDFSIO:  IO rate std deviation:
2.065184667891268
15/03/17 10:05:07 INFO fs.TestDFSIO:     Test exec time sec:
288.625
15/03/17 10:05:07 INFO fs.TestDFSIO:
```

Make sure you clean up the test files in order to save storage space after the tests have been run:

```
[admin@host ~]# hdfs jar /opt/hadoop/share/hadoop/mapreduce/hadoop-
mapreduce-client-jobclient-2.4.0-tests.jar TestDFSIO –clean
```

The resulting output may resemble:

```
15/03/17 10:08:19 INFO fs.TestDFSIO: TestDFSIO.1.7
15/03/17 10:08:19 INFO fs.TestDFSIO: nrFiles = 1
15/03/17 10:08:19 INFO fs.TestDFSIO: nrBytes (MB) = 1.0
15/03/17 10:08:19 INFO fs.TestDFSIO: bufferSize = 1000000
15/03/17 10:08:19 INFO fs.TestDFSIO: baseDir =
/benchmarks/TestDFSIO
15/03/17 10:08:20 INFO fs.TestDFSIO: Cleaning up test files
```

Monitoring UI Nodes

The UI nodes are in charge of delivering the client side code (via HTTP) and answering queries from the different clients. Normally, the UI nodes cache data in order to answer requests for commonly accessed data more quickly.

When a particular data set is required, the UI node checks the local cache. If no match is found, a query to the database back end is dispatched. The log files from the UI node show the queries sent to the database, and the time taken by each query.

System Alerts

This topic describes the most common alerts on the system.

Cold Start Alert

The **coldStart** alert is generated when the system or server goes down and is restarted gracefully.

This alert requires manual intervention to clear. Contact Cisco Technical Support.

Data Alerts

Alerts about interruptions of data flow include the following:

- The **collectorNoData** alert is generated when the Collector node has not received data for a configurable period of time (default is 600 seconds).

  When resolved: **dataResume** alert is generated when the Collector starts receiving data. Contact Cisco Technical Support if alarm does not clear after 20 minutes.

- The **droppedFlowCrossed** alert is generated when the Collector drops or discards a larger percentage of data than the threshold setting (default is 10%) within a 15-minute bucket.

  When resolved: **droppedFlowAlarmCleared** alert is generated when the rate of data discard rises above the threshold.

  Verify the disk usage on the collector per steps in "Disk Space Low Alert" on the next page. To view how much disk space is free and verify that the disks are up and running, use the `df -k` command.

  Contact Cisco Technical Support if the alarm does not clear after 30 minutes.

- The **hdfsNamenodeStatusTrap** alert is triggered when the Collector cannot access HDFS.

Disk Space Low Alert

The **diskSpaceLow** alert is generated when disk space drops below a configured threshold (default 93%). When the disk space rises back above the threshold, the **diskSpaceOK** alert is generated.

1. Log into the node and start the command shell.
2. Check the **fsMountPoint** to find the name of the alarmed mount point and partition.
3. Log into the node and identify which partition is full using the command:

```
admin@host
> df -k
```

   If the `/var` partition is full, clear older log files on the node.

   If the `/data` partition is full, check for SAN capacity and connectivity issues.

This alarm clears when the available disk space rises back above the threshold.

Contact Cisco Technical Support if alarm does not clear after more than 10 minutes.

Link Down Alert

These alerts monitor the availability of the server's network interface:

- **linkDown**—interface of the server is down
- **linkUp—**interface is back up

To troubleshoot down links:

1. Check **HP OpenView** and **HP iLO** management tools. Monitor the NMS.
2. Check and verify physical network connectivity.
3. Contact Cisco Technical Support if alarm does not clear after 15 minutes.

Process Crash Alerts

The **procCrash** alert is generated when the Collector process crashes and the system automatically recovers.

When resolved: The Collector process is restarted, **procCrash** clears and the **procRelaunch** alert is generated. Contact Cisco Technical Support if the alarm does not clear after 15 minutes.

Paging Activity High Alert

The **pagingActivityHigh** alert is generated when the paging activity goes above the configured upper threshold (default threshold is 16000 page faults).

When resolved: **pagingActivityOK** alert is generated when the paging activity drops below the threshold.

Contact Cisco Technical Support if the alarm does not clear after 2 hours.

Process Liveness Failure Alert

The **procLivenessFailure** alert is generally caused by a software error. This trap is generated when a process accidentally hangs and is restarted. The alarm clears when the process is restarted and processes are restarted automatically.

Contact Cisco Technical Support if alarm does not clear after 15 minutes or if it keeps repeating.

Resource Overload Alerts

Resource overload alerts include:

- **cpuUtilHigh**—CPU utilization has risen above configured threshold.

  When resolved: **cpuUtilOK** alert is generated when the CPU utilization drops back below the threshold.

- **diskIOHigh**—Disk IO utilization has risen above configured threshold.

  When resolved: **diskIOOK** alert is generated when the disk IO utilization drops back below the threshold.

- **memUtilizationHigh**—Memory utilization has risen above configured threshold.

- **netUtilizationHigh**—Network utilization has risen above configured threshold.

  When resolved: **netUtilizationOK** alert is generated when the network utilization drops back below the threshold.

The system will generally recover from resource overload situations. However, if the alarm persists for more than 45 minutes, follow these steps:

1. Log into the node and access the CLI.
2. Run the command `free -g` to display memory usage. If memory is completely used, check which process is consuming memory.
3. Run the `top M` command to list processes ordered by memory usage. Provide this information to Cisco Technical Support for further assistance.
4. Identify processes that are utilizing high CPU resources. Provide this information to Cisco Technical Support for further assistance.

Unexpected Shutdown Alert

The **unexpectedShutdown** alert is generated when a node in the system is restarted ungracefully. This alert has a severity level of Critical.

This alarm must be manually cleared. Contact Cisco Technical Support for assistance.

Viewing Reports for Uncategorized Entities

If the system is showing a large percentage of user agents, URLs, TACs, Content Types, or Ports that are uncategorized, you can view a report listing the heavy hitters, in terms of highest tonnage and hit count, among the uncategorized entities.

To view these reports you need to access the CLI.

You can use the information in the reports to add categories for these uncategorized heavy hitters to the appropriate information database and assign them to categories. To add entries to information bases, see "Overriding Categorizations From Information Bases" on page 129.

Note: If there are certain user agents, URLs, TACs, Content Types or ports that make up a large percentage of the overall traffic and are not being categorized by the system, share the uncategorized reports with the Customer Support team so that they can be added in the next update of the information bases, in the next software release or patch.

Before You Begin

Before the system can run the weekly uncategorized entity reports, the following conditions must be met:

- Destination directories specified in the `serverFile_uncatReports` file are present on corresponding servers.
- The uncategorized aggregator job (UA) and `OfflineEngineUncatReports` jobs are running on the system.

Checking Specific Jobs

To check if UA and `OfflineEngineUncatReports` jobs are running:

1. Login and go to the pmx oozie subshell:

```
> en
# conf t
(config)# pmx subshell subshell-name
```

2. Run:

```
pm extension (oozie)# show coordinator RUNNING jobs
```

Or,

1. Log in and go to the `_shell`:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# _shell
```

2. Run:

```
admin@host# hadoop dfs -cat /data/UA/done.txt
admin@host# hadoop dfs -cat
/data/OfflineEngineUncatReports/done.txt
```

Both methods should show the last date that the jobs were run.

About the Uncategorized Aggregator Job

The uncategorized aggregator job runs once a week at calendar weekly boundary and creates a report for the last 7 days' uncategorized data. The job generates separate reports for tonnage in bytes and hit count for TACs, user agents, content types, devices, ports, service providers, mobile applications, URLs, traffic type category and protocols.

Note: The weekly-uncategorized data reports are available only for the previous four weeks, as per the current system time.

To check the last date on which the uncategorized aggregator job ran:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# _shell
admin@host# hdfs dfs –cat /data/UA/done.txt
```

Viewing Reports

To fetch and view a report for uncategorized entities:

1. Login and go to the pmx `aggregation_center` subshell:

   ```
   > en
   # conf t
   (config)# pmx subshell subshell-name
   ```

2. Run:

   ```
   pm extension (aggregation_center)> show report report-name
   dataset uncategorized_weekly_report time date
   ```

   Where:

   - `report-name` is one of the following values:
     - report_ct_bytes.map
     - report_ct_hits.map
     - report_db_device_comparison_bytes.map
     - report_db_device_comparison_hits.map
     - report_device_bytes.map
     - report_device_hits.map
     - report_mobile_app_bytes.map
     - report_mobile_app_hits.map
     - report_sp_bytes.map
     - report_sp_hits.map
     - report_tac_bytes.map
     - report_tac_hits.map
     - report_traffic_type_category_bytes.map
     - report_traffic_type_category_hits.map
     - report_traffic_type_protocol_bytes.map
     - report_traffic_type_protocol_hits.map
     - report_ua_bytes.map
     - report_ua_hits.map
     - report_uncat_ct_bytes.map
     - report_uncat_ct_hits.map
     - report_uncat_port_bytes.map
     - report_uncat_port_hits.map

    - report_url_bytes.map
    - report_url_hits.map
- `date` is in the format `YYYY-MM-DDT00:00Z`. For example, `2014-08-25`.

Examples of Reports

This section show examples for each report produced by the system to list the uncategorized entities which are heavy hitters.

All reports start with logging in and going to the `aggregation_center` subshell:

```
> en
# conf t
(config)# pmx subshell subshell-name
```

Uncategorized URL Heavy Hitters by Tonnage

```
pm extension (aggregation_center)> show report report_url_bytes.map
dataset uncategorized_weekly_report time 2013-10-11
```

The resulting output may resemble:

```
WARNING: org.apache.hadoop.metrics.jvm.EventCounter is deprecated.
Please use org.apache.hadoop.log.metrics.EventCounter in all the
log4j.properties files.
Report Fetched Successfully
APP         IS_WHITELIST_MEMBER(1)  HITS      TONNAGE
report_mobile_app_hits.map For DataSet uncategorized_weekly_report
and for Date 2014/08/11
----------------------------------------------------------
Adobe Flash Player     0       11608   1761228482.00
FOX Sports Mobile      0       4512    4797286013.00
Dragon Dictation       0       0       1428212818.00
TV Guide Mobile        0       175240  58780222540.00
LINE   0       272403  92614440455.00
Pinterest      1       550418  185564084583.00
Ace Budget     0       14527   5440438374.00
```

```
HeyTell         0       133468  45441872445.00
M-Manage Calendar       0       0       1491198280.00
Shopper         0       105885  36710206356.00
Groupon         1       749449  263403177608.00
Tumblr 1        55265   19764022843.00
MeetMoi: The Mobile MatchMaker 0       36390   9048543752.00
Countdown       0       48990   14865722974.00
Chess With Friends      0       16319   9110225341.00
Remember The Milk       0       74195   23879890501.00
Storm8 Games    0       216381  72864862138.00
iGun Pro        0       0       6336185935.00
Grindr 1        357812  124410105067.00
Pandora Radio   1       3588138 1257824715170.00
```

Uncategorized Content Type Heavy Hitters by Tonnage

```
pm extension (aggregation_center)> show report report_uncat_ct_
bytes.map dataset uncategorized_weekly_report time 2014-08-11
```

The resulting output may resemble:

```
WARNING: org.apache.hadoop.metrics.jvm.EventCounter is deprecated.
Please use org.apache.hadoop.log.metrics.EventCounter in all the
log4j.properties files.
Report Fetched Successfully
MIME                                                       HITS
TONNAGE
report_uncat_ct_bytes.map For DataSet uncategorized_weekly_report
and for Date 2014/08/11
----------------------------------------------------------
audio/amr-wb+  643753  234771172339.00
audio/EVRCB    650552  223021173784.00
video/vnd.dece.sd      638886  226238961590.00
application/vnd.smaf   614370  217668901502.00
application/vnd.rainstor.data  604876  214049781874.00
image/ktx      653739  226373788606.00
```

```
video/DV        670530  233784641907.00
audio/G722      617883  217526441995.00
audio/example   609644  211722517159.00
application/vnd.wfa.wsc        640439  222882812464.00
audio/PCMU      662477  234106328294.00
audio/vnd.nuera.ecelp7470      622262  215530816232.00
audio/EVRC0     648275  225218660912.00
audio/EVRC1     640518  221099073110.00
audio/PCMA      651399  227581978674.00
```

Uncategorized Content Type Heavy Hitters by Hits

```
pm extension (aggregation_center)> show report report_uncat_ct_
hits.map dataset uncategorized_weekly_report time 2014-08-11
```

The resulting output may resemble:

```
WARNING: org.apache.hadoop.metrics.jvm.EventCounter is deprecated.
Please use org.apache.hadoop.log.metrics.EventCounter in all the
log4j.properties files.
Report Fetched Successfully
MIME                                                         HITS
TONNAGE
report_uncat_ct_hits.map For DataSet uncategorized_weekly_report
and for Date 2014/08/11
----------------------------------------------------------
audio/amr-wb+  1159276 422773142078.00
audio/EVRCB    1171468 401603258676.00
video/vnd.dece.sd      1150489 407407640052.00
application/vnd.smaf   1106342 391972307777.00
application/vnd.rainstor.data  1089236 385449436723.00
image/ktx      1177197 407634971682.00
video/DV       1207497 421004577262.00
audio/G722     1112789 391700633523.00
audio/example  1097816 381261001125.00
application/vnd.wfa.wsc        1153280 401359528151.00
```

```
audio/PCMU     1192954 421570076026.00
audio/vnd.nuera.ecelp7470      1120534 388119672143.00
audio/EVRC0    1167364 405561532055.00
audio/EVRC1    1153433 398149303349.00
audio/PCMA     1173000 409818178858.00
audio/G726-32  1113417 382559705836.00
```

Uncategorized Ports Heavy Hitters by Tonnage

```
pm extension (aggregation_center)> show report report_uncat_port_
bytes.map dataset uncategorized_weekly_report time 2014-08-04
```

The resulting output may resemble:

```
WARNING: org.apache.hadoop.metrics.jvm.EventCounter is deprecated.
Please use org.apache.hadoop.log.metrics.EventCounter in all the
log4j.properties files.
Report Fetched Successfully
PORT   HITS              TONNAGE
report_uncat_port_bytes.map For DataSet uncategorized_weekly_report
and for Date 2014/08/04
----------------------------------------------------------
1304   3390432 1204814403192.00
1301   3378333 1202639295828.00
666    3401119 1175700715925.00
132    3319717 1173322825895.00
23     3429917 1199045015196.00
1444   3460942 1219619777938.00
1468   3411753 1171956809125.00
283    3401396 1182997562968.00
1313   3299489 1146583648087.00
1312   3403704 1190903927257.00
1092   3526087 1244895377201.00
1310   3255271 1121777584488.00
1318   3377473 1188720526033.00
```

```
120    3482138 1219279956243.00
1494   3249877 1124926834915.00
```

Uncategorized Ports Heavy Hitters by Hits

```
pm extension (aggregation_center)> show report report_uncat_port_
hits.map dataset uncategorized_weekly_report time 2014-08-04
```

The resulting output may resemble:

```
WARNING: org.apache.hadoop.metrics.jvm.EventCounter is deprecated.
Please use org.apache.hadoop.log.metrics.EventCounter in all the
log4j.properties files.
Report Fetched Successfully
PORT    HITS           TONNAGE
report_uncat_port_hits.map For DataSet uncategorized_weekly_report
and for Date 2014/08/04
----------------------------------------------------------
1304   3390432 1204814403192.00
1301   3378333 1202639295828.00
666    3401119 1175700715925.00
132    3319717 1173322825895.00
23     3429917 1199045015196.00
1444   3460942 1219619777938.00
1468   3411753 1171956809125.00
283    3401396 1182997562968.00
1313   3299489 1146583648087.00
1312   3403704 1190903927257.00
1092   3526087 1244895377201.00
1310   3255271 1121777584488.00
1318   3377473 1188720526033.00
120    3482138 1219279956243.00
1494   3249877 1124926834915.00
1401   3468027 1214147966002.00
1065   3297786 1165910968273.00
533    3473939 1190206934467.00
```

Categorized Traffic Type Categories by Tonnage with in_whitelist Indicator

```
pm extension (aggregation_center)> show report report_traffic_type_
category_bytes.map dataset uncategorized_weekly_report time 2014-
09-01
```

The resulting output may resemble:

```
WARNING: org.apache.hadoop.metrics.jvm.EventCounter is deprecated.
Please use org.apache.hadoop.log.metrics.EventCounter in all the
log4j.properties files.
Report Fetched Successfully
CATEGORY_NAME  IS_WHITELIST_MEMBER(1)  HITS       TONNAGE
report_traffic_type_category_bytes.map For DataSet uncategorized_
weekly_report and for Date 2014/09/01
----------------------------------------------------------
filesharing    0                       195878479  68644950906383
filesharing    1                       55247672   19354274088401
voip           1                       48838803   17087062794051
voip           0                       30724693   10814548472929
tunneling      0                       30615547   10766020138906
tunneling      1                       61412958   21525120328919
video          1                       24399662   8519959229607
web            1                       195778984  68747804598669
gaming         0                       97845250   34401450337665
gaming         1                       36862642   12849627109250
standard       0                       239428882  83967839654682
standard       1                       159402572  55821163251012
```

Categorized Traffic Type Categories by Hits with in_whitelist Indicator

```
pm extension (aggregation_center)> show report report_traffic_type_
category_hits.map dataset uncategorized_weekly_report time 2014-09-
01
```

The resulting output may resemble:

```
WARNING: org.apache.hadoop.metrics.jvm.EventCounter is deprecated.
Please use org.apache.hadoop.log.metrics.EventCounter in all the
log4j.properties files.
Report Fetched Successfully
CATEGORY_NAME  IS_WHITELIST_MEMBER(1)  HITS       TONNAGE
report_traffic_type_category_hits.map For DataSet uncategorized_
weekly_report and for Date 2014/09/01
----------------------------------------------------------
filesharing    0                       195878479  68644950906383
filesharing    1                       55247672   19354274088401
voip           1                       48838803   17087062794051
voip           0                       30724693   10814548472929
tunneling      0                       30615547   10766020138906
tunneling      1                       61412958   21525120328919
video          1                       24399662   8519959229607
web            1                       195778984  68747804598669
gaming         0                       97845250   34401450337665
gaming         1                       36862642   12849627109250
standard       0                       239428882  83967839654682
standard       1                       159402572  55821163251012
```

Categorized Traffic Type Protocols by Tonnage with in_whitelist Indicator

```
pm extension (aggregation_center)> show report report_traffic_type_
protocol_bytes.map dataset uncategorized_weekly_report time 2014-
09-01
```

The resulting output may resemble:

```
WARNING: org.apache.hadoop.metrics.jvm.EventCounter is deprecated.
Please use org.apache.hadoop.log.metrics.EventCounter in all the
log4j.properties files.
Report Fetched Successfully
```

```
PROTOCOL_NAME                 IS_WHITELIST_MEMBER(1)  HITS
TONNAGE
report_traffic_type_protocol_bytes.map For DataSet uncategorized_
weekly_report and for Date 2014/09/01
----------------------------------------------------------
husky                         0                       24519576
8611714947215
p2p                           1                       6131233
2154425736561
jlicelmd                      0                       24220901
8481744999288
issd                          0                       24539593
8655096298107
ivcollector                   0                       24637717
8647094109286
tvants                        0                       6120196
2163015934309
blackjack                     1                       24407411
8585510313278
bittorrent                    1                       6262842
2215215445073
rtp                           1                       6076485
2122783109922
texar                         0                       24510503
8640516000998
aeroflight-ads                0                       48827432
17136900847098
bootpc                        0                       24517119
8631319481998
skype-video                   1                       6068617
2106777986215
macromedia_rtmp               1                       12392219
4311495385838
```

```
wtp_wsp_connection_oriented  0                        5988490
2112259651054
yourfreetunnel               0                        6167033
2159483169112
```

Categorized Traffic Type Protocols by Hits with in_whitelist Indicator

```
pm extension (aggregation_center)> **show report report_traffic_type_
protocol_hits.map dataset uncategorized_weekly_report time 2014-09-
01**
```

The resulting output may resemble:

```
WARNING: org.apache.hadoop.metrics.jvm.EventCounter is deprecated.
Please use org.apache.hadoop.log.metrics.EventCounter in all the
log4j.properties files.
Report Fetched Successfully
PROTOCOL_NAME                IS_WHITELIST_MEMBER(1)  HITS
TONNAGE
report_traffic_type_protocol_hits.map For DataSet uncategorized_
weekly_report and for Date 2014/09/01
----------------------------------------------------------
husky                        0                        24519576
8611714947215
p2p                          1                        6131233
2154425736561
jlicelmd                     0                        24220901
8481744999288
issd                         0                        24539593
8655096298107
ivcollector                  0                        24637717
8647094109286
tvants                       0                        6120196
2163015934309
blackjack                    1                        24407411
```

```
8585510313278
bittorrent                    1                      6262842
2215215445073
rtp                           1                      6076485
2122783109922
texar                         0                      24510503
8640516000998
aeroflight-ads                0                      48827432
17136900847098
bootpc                        0                      24517119
8631319481998
skype-video                   1                      6068617
2106777986215
macromedia_rtmp               1                      12392219
4311495385838
wtp_wsp_connection_oriented   0                      5988490
2112259651054
yourfreetunnel                0                      6167033
2159483169112
```

Uncategorized URL Heavy Hitters by Hit Count

```
pm extension (aggregation_center)> show report report_url_hits.map
dataset uncategorized_weekdaily_report time 2013-10-10T00:00Z
```

The resulting output may resemble:

```
Report Fetched Successfully
URL                         HITS       TONNAGE
report_url_hits.map For DataSet uncategorized_weekdaily_report and
for Date 2013/10/10
----------------------------------------------------------
c2.cduniverse.ws            2842       1316020400
widget.youappi.com          5678       1908140283
```

```
www.morritastube.com        2829       1065407349
meet24.com                  9959       2412706289
slots.sharkparty.com        11353      5231791141
...
```

Uncategorized User Agent Heavy Hitters by Tonnage

```
pm extension (aggregation_center)> show report report_ua_bytes.map
dataset uncategorized_weekly_report time 2013-10-10T00:00Z
```

The resulting output may resemble:

```
Report Fetched Successfully
report_ua_bytes.map For DataSet uncategorized_weekly_report and for
Date 2013/10/10
----------------------------------------------------------
UA                                                  HITS   TONNAGE
BornRich/23 CFNetwork/609 Darwin/13.0.0             5235   2700450303
Animals%20Puzzle/3 CFNetwork/548.1.4 Darwin/11.0.0
                                                    1749   3381479708
31/5.2.6.1 (Linux; U; Android 2.3.5; en-us; PC36100 Build/GRJ90)
                                                    360    2284914186
15/4 (Linux; U; Android 4.0.3; en-us; PG86100 Build/IML74K)
                                                    796    3337577916
AppyGeek.iPhone/2.4.1 CFNetwork/609 Darwin/13.0.0
                                                    12809  4316407884
...
```

Uncategorized User Agent Heavy Hitters by Hits

```
pm extension (aggregation_center)> show report report_ua_hits.map
dataset uncategorized_weekly_report time 2013-10-11T00:00Z
```

The resulting output may resemble:

```
Report Fetched Successfully
report_ua_hits.map For DataSet uncategorized_weekly_report and for
Date 2013/10/11
```

```
------------------------------------------------------------------
UA                                                HITS    TONNAGE
GMM/3.0 (LGL85C GRJ22); gzip                      1014    151739896
DXStatisticAppInfo                                303     1172478338
CarMaze/1.4 CFNetwork/548.0.4 Darwin/11.0.0       2847    1040677865
DogWhistler/1.1.1 CFNetwork/609 Darwin/13.0.0     4226    2315680280
8K3DQ2NBT4.com.vevo.iphone/5842 iPhone OS/6.0.1   3164    1298624698
...
```

Uncategorized TAC Heavy Hitters by Tonnage

```
pm extension (aggregation_center)> show report report_tac_bytes.map
dataset uncategorized_weekly_report time 2013-10-10T00:00Z
```

The resulting output may resemble:

```
Report Fetched Successfully
report_tac_bytes.map For DataSet uncategorized_weekly_report and
for Date 2013/10/10
----------------------------------------------------------
TAC        HITS         TONNAGE
350176     10706085     3746556562178.00
351308     10753822     3808925836088.00
351002     10768013     3789770208197.00
1216200    10715887     3755896252546.00
520250     10688524     3738700196803.00
...
```

Uncategorized TAC Heavy Hitters by Hit Count

```
pm extension (aggregation_center)> show report report_tac_hits.map
dataset uncategorized_weekly_report time 2013-10-11T00:00Z
```

The resulting output may resemble:

```
Report Fetched Successfully
report_tac_hits.map For DataSet uncategorized_weekly_report and for
Date 2013/10/10
```

```
-----------------------------------------------------------
TAC        HITS        TONNAGE
350176     5951243     2082657141356.00
351308     5978177     2117437827993.00
351002     5985072     2106367056306.00
1216200    5956928     2087884304997.00
520250     5941658     2078276199339.00
...
```

Uncategorized Device Heavy Hitters by Tonnage

```
pm extension (aggregation_center)> show report report_device_
bytes.map dataset uncategorized_weekly_report time 2013-10-
11T00:00Z
```

The resulting output may resemble:

```
Report Fetched Successfully
DEVICE        IS_WHITELIST_MEMBER(1)   HITS       TONNAGE
report_device_bytes.map For DataSet uncategorized_weekly_report and
for Date 2013/10/11
-----------------------------------------------------------
Opera Tablet on Android        1       15948      8130598344.00
Gradiente GC-1+                0       5951233    2075265582467.00
HTC Aria G7/Liberty (A6380)    0       5803037    2044011069754.00
Windows PCs Mobile Hotspot     1       14638741   5125692626167.00
Desktop Linux                  0       104216     33094411498.00
...
```

Uncategorized Device Heavy Hitters by Hits

```
pm extension (aggregation_center)> show report report_device_
hits.map dataset uncategorized_weekly_report time 2013-10-11T00:00Z
```

The resulting output may resemble:

```
Report Fetched Successfully
DEVICE         IS_WHITELIST_MEMBER(1)  HITS        TONNAGE
```

```
report_device_hits.map For DataSet uncategorized_weekly_report and
for Date 2013/10/11
----------------------------------------------------------
Opera Tablet on Android        1       15948       8130598344.00
Gradiente GC-1+                0       5951233     2075265582467.00
HTC Aria G7/Liberty (A6380)    0       5803037     2044011069754.00
Windows PCs Mobile Hotspot     1       14638741    5125692626167.00
Desktop Linux                  0       104216      33094411498.00
...
```

Uncategorized Service Providers Heavy Hitters by Tonnage

```
pm extension (aggregation_center)> show report report_sp_bytes.map
dataset uncategorized_weekly_report time 2013-10-11T00:00Z
```

The resulting output may resemble:

```
Report Fetched Successfully
SP          IS_WHITELIST_MEMBER(1)     HITS     TONNAGE
report_sp_bytes.map For DataSet uncategorized_weekly_report and for
Date 2013/10/11
----------------------------------------------------------
trivialtechnology.com          0       0       18923864722
okcupid.com                    0       0       34348687407
meetmoi.com                    0       0       5142415458
kiloo.com                      1       0       31030207356
chillingo.com                  0       26233   24862292041
...
```

Uncategorized Service Providers Heavy Hitters by Hits

```
pm extension (aggregation_center)> show report report_sp_hits.map
dataset uncategorized_weekly_report time 2013-10-11T00:00Z
```

The resulting output may resemble:

```
Report Fetched Successfully
SP             IS_WHITELIST_MEMBER(1)  HITS     TONNAGE
```

```
report_sp_hits.map For DataSet uncategorized_weekly_report and for
Date 2013/10/11
----------------------------------------------------------
trivialtechnology.com         0        0       18923864722
okcupid.com                   0        0       34348687407
meetmoi.com                   0        0       5142415458
kiloo.com                     1        0       31030207356
chillingo.com                 0        26233   24862292041
...
```

Uncategorized Mobile Application Heavy Hitters by Tonnage

```
pm extension (aggregation_center)> show report report_mobile_app_
bytes.map dataset uncategorized_weekly_report time 2013-10-
11T00:00Z
```

The resulting output may resemble:

```
Report Fetched Successfully
APP          IS_WHITELIST_MEMBER(1)    HITS    TONNAGE
report_mobile_app_bytes.map For DataSet uncategorized_weekly_report
and for Date 2013/10/11
----------------------------------------------------------
Adobe Flash Player              0     11608   1761228482.00
FOX Sports Mobile               0     4512    4797286013.00
Dragon Dictation                0     0       1428212818.00
TV Guide Mobile                 0     175240  58780222540.00
LINE                            0     272403  92614440455.00
...
```

Uncategorized Mobile Application Heavy Hitters by Hits

```
pm extension (aggregation_center)> show report report_mobile_app_
hits.map dataset uncategorized_weekly_report time 2013-10-11T00:00Z
```

The resulting output may resemble:

```
Report Fetched Successfully
APP            IS_WHITELIST_MEMBER(1)  HITS    TONNAGE
report_mobile_app_hits.map For DataSet uncategorized_weekly_report
and for Date 2013/10/11
----------------------------------------------------------
Adobe Flash Player                0    11608   1761228482.00
FOX Sports Mobile                 0    4512    4797286013.00
Dragon Dictation                  0    0       1428212818.00
TV Guide Mobile                   0    175240  58780222540.00
LINE                              0    272403  92614440455.00
...
```

Uncategorized Content Type Heavy Hitters by Tonnage

```
pm extension (aggregation_center)> show report report_uncat_ct_
bytes.map dataset uncategorized_weekly_report time 2013-10-
11T00:00Z
```

The resulting output may resemble:

```
Report Fetched Successfully
MIME                             HITS          TONNAGE
report_uncat_ct_bytes.map For DataSet uncategorized_weekly_report
and for Date 2013/10/11
----------------------------------------------------------
audio/amr-wb+                    643753        234771172339.00
audio/EVRCB                      650552        223021173784.00
video/vnd.dece.sd                638886        226238961590.00
application/vnd.smaf             614370        217668901502.00
application/vnd.rainstor.data    604876        214049781874.00
...
```

Uncategorized Content Type Heavy Hitters by Hits

```
pm extension (aggregation_center)> show report report_uncat_ct_
hits.map dataset uncategorized_weekly_report time 2013-10-10T00:00Z
```

The resulting output may resemble:

```
Report Fetched Successfully
MIME                                 HITS      TONNAGE
report_uncat_ct_hits.map For DataSet uncategorized_weekly_report
and for Date 2013/10/10
----------------------------------------------------------
audio/amr-wb+                        1159276   422773142078.00
audio/EVRCB                          1171468   401603258676.00
video/vnd.dece.sd                    1150489   407407640052.00
application/vnd.smaf                 1106342   391972307777.00
application/vnd.rainstor.data        1089236   385449436723.00
```

Exporting Uncategorized Data Reports

The twenty-four weekly uncategorized data reports are transferred every week to the destination specified in the `/data/work/serverFile_uncatReports` file after they are generated. This file is created at the time of initial system setup.

The `/data/work/serverFile_uncatReports` file contains the following details:

```
# cat /data/work/serverFile_uncatReports
192.168.147.18, admin, admin@123, /data/uncat_reports
```

In this example, the reports are copied to the server 192.168.147.18 under the `/data/uncat_reports` folder. The reports are copied to destination server after they are generated by system for the beginning of each week.

To verify the reports in the file, log in to the server on which the reports are stored and list the files:

```
# cd /data/uncat_reports
# ls -rlth
```

The resulting output may resemble:

```
-rw-r--r-- 1 admin root    0 Sep 17 11:43 1409529600.report_ct_
bytes.map
-rw-r--r-- 1 admin root    0 Sep 17 11:43 1409529600.report_ct_
hits.map
-rw-r--r-- 1 admin root    0 Sep 17 11:44 1409529600.report_db_
```

```
device_comparison_bytes.map
-rw-r--r-- 1 admin root    0 Sep 17 11:44 1409529600.report_db_
device_comparison_hits.map
-rw-r--r-- 1 admin root 4.5K Sep 17 11:44 1409529600.report_device_
bytes.map
-rw-r--r-- 1 admin root 4.5K Sep 17 11:44 1409529600.report_device_
hits.map
-rw-r--r-- 1 admin root 7.3K Sep 17 11:44 1409529600.report_mobile_
app_bytes.map
-rw-r--r-- 1 admin root 7.3K Sep 17 11:44 1409529600.report_mobile_
app_hits.map
-rw-r--r-- 1 admin root  72K Sep 17 11:44 1409529600.report_sp_
bytes.map
-rw-r--r-- 1 admin root  72K Sep 17 11:44 1409529600.report_sp_
hits.map
-rw-r--r-- 1 admin root    0 Sep 17 11:44 1409529600.report_tac_
bytes.map
-rw-r--r-- 1 admin root    0 Sep 17 11:44 1409529600.report_tac_
hits.map
-rw-r--r-- 1 admin root  666 Sep 17 11:44 1409529600.report_
traffic_type_category_bytes.map
-rw-r--r-- 1 admin root  666 Sep 17 11:44 1409529600.report_
traffic_type_category_hits.map
-rw-r--r-- 1 admin root 9.8K Sep 17 11:44 1409529600.report_
traffic_type_protocol_bytes.map
-rw-r--r-- 1 admin root 9.8K Sep 17 11:45 1409529600.report_
traffic_type_protocol_hits.map
-rw-r--r-- 1 admin root  85K Sep 17 11:45 1409529600.report_ua_
bytes.map
-rw-r--r-- 1 admin root  80K Sep 17 11:45 1409529600.report_ua_
hits.map
-rw-r--r-- 1 admin root    0 Sep 17 11:45 1409529600.report_uncat_
ct_bytes.map
```

```
-rw-r--r-- 1 admin root    0 Sep 17 11:45 1409529600.report_uncat_
ct_hits.map
-rw-r--r-- 1 admin root    0 Sep 17 11:45 1409529600.report_uncat_
port_bytes.map
-rw-r--r-- 1 admin root    0 Sep 17 11:45 1409529600.report_uncat_
port_hits.map
-rw-r--r-- 1 admin root  42K Sep 17 11:45 1409529600.report_url_
bytes.map
-rw-r--r-- 1 admin root  40K Sep 17 11:45 1409529600.report_url_
hits.map
```

Note: 1409529600 is the epoch time of 01 September 2014 on which these reports were generated.

For example,

```
# cat 1409529600.report_url_bytes.map
asfhj.dfdfkh48.com 5104 408160000
asfhj.dfdfkh46.com 5220 522000000
asfhj.dfdfkh44.com 5336 6403200000
asfhj.dfdfkh42.com 5452 7632800000
asfhj.dfdfkh40.com 5568 8908800000
asfhj.dfdfkh54.com 4756 94960000
asfhj.dfdfkh50.com 4988 298920000
asfhj.dfdfkh52.com 4872 194680000
yatraABCD.com 864 213900
```

Stopping Automatic Export of Reports

To stop the `OfflineEngineUncatReports`:

1. Log in and go to the pmx oozie subshell:

   ```
   > en
   # conf t
   (config)# pmx subshell subshell-name
   ```

2. Run:

```
pm extension (oozie)> stop jobname OfflineEngineUncatReports
```

Adding New Uncategorized Report Destinations

To add a new destination for the reports of uncategorized entities:

1. Create the directory path on the destination server.
2. Add new destinations for the reports on both the master and then the standby Collector nodes:

```
# vi /data/work/serverFile_uncatReports
192.168.147.18, admin, admin@123, /data/uncat_reports_new
192.168.151.79, admin, admin@123, /data/reports/
192.168.100.2,  admin, admin@123, /data/reports/
192.168.151.81, admin, admin@123, /data/reports/
```

3. Log in and go to the pmx oozie subshell:

```
> en
# conf t
(config)# pmx subshell subshell-name
```

4. Run the `OfflineEngineUncatReports` job with the new entries:

```
pm extension (oozie)> run job OfflineEngineUncatReports
```

The reports will be copied to all the destinations specified in the `/data/-work/serverFile_uncatReports` file for each week.

Adding New Components

Adding a New Gateway for EDR and Bulkstats

This section describes the procedure to add a new ASR gateway to the MURAL system so that it can send EDR and bulkstats data to MURAL. For this process to succeed, the MURAL system must be in a functional state. The current release of Mural has been verified to work with both ASR 5000 and ASR 5500.

Basic assumptions to be kept in mind while adding a new dc gateway:

- DC, ASR, and gateway are synonymous terms and hold the same meaning for the system. Gateway/DC name is a unique key for a gateway configuration.
- In bulkstats hierarchy, gateway name and dc name will be the same.

Prerequisites

Before configuring the Collector for a new ASR, ensure that you have the information for the following attributes:

- GATEWAY NAME = GATEWAY name that will be visible on the MURAL UI for the Bulk Stats or EDR data sent from this gateway. For example, **GMPLAB1**.
- GATEWAY EDR_FILE_PATH = Local directory on the Collector where this new ASR will drop the EDR record files. For example, **/California/edr111/**.
- GATEWAY BULKSTATS_FILE_PATH = Local directory on the Collector where this new ASR will drop the bulkstats record files. For example, **/California/bs111**.
-
- GATEWAY TIMEZONE = Timezone where gateway is physically located and in which the new ASR will send the time stamp in the file name. For example, **America/Recife**.

- REGION NAME = Name of the region to which this GATEWAY belongs. For example, **USA**.
- AREA NAME = Name of the area to which this GATEWAY belongs. For example, EAST.
- EDR_COLLECTOR_FILENAME_PATTERN = Pattern of the file to be configured in the Collector for EDR flow and HTTP files.
- BULKSTAT_COLLECTOR_FILENAME_PATTERN = Pattern of the file to be configured in the Collector for bulkstats files.
- COLLECTOR IP = Collectors where the gateway needs to be configured.
- EDR_FILENAME_PATTERN = Pattern of the EDR filename. For example, ***_MURAL-edr_*_%MM%DD%YYYY%hh%mm%ss.***.
- BULKSTAT_FILENAME_PATTERN = Pattern of the Bulkstat filename. For example, **GMPLAB1_%YYYY%MM%DD%hh%mm%ss**.
- TYPE = Type of the cluster. For example, **HA**.
- SCHEMA_VERSION = Version of the schema. For example, **15**.
- GATEWAY_IP = IP address of the gateway. For example, **10.10.10.255**.
- EDR_COLLECTOR_FILENAME_PATTERN = %DC_*_*_%MM%DD%YYYY%hh%mm%ss.*
- BULKSTAT_COLLECTOR_FILENAME_PATTERN = *_%YYYY%MM%DD%hh%mm%ss
- COLLECTOR_IP = 10.10.10.101,10.10.10.102

Points to Remember

- All input directories are created on the `/data/collector` path. Therefore, ASR send the EDR files to the `/data/collector/<GATEWAY.EDR_FILE_PATH>` directory and the Bulkstats file to the `/data/collector/<GATEWAY.BULKSTATS_FILE_PATH>` directory.
- `GATEWAY.EDR_FILE_PATH` and `GATEWAY.BULKSTATS_FILE_PATH` must always begin with a backslash (/).

- Files coming for BulkStats must contain the Gateway Name delimited by an underscore (_). For example, **GMPLAB1_20131006121500**.
- Filename pattern must be in sync with Collector configs.
- ASR must send the gateway name in place of %DC as specified in filename pattern in collector configuration.
- If the file names are going to come with ".gz" or ".txt" at the end with no other special charcter in between timestamp & ".gz",then it is mandatory to append ".*" in the filename format while adding gateway. .But if "_" is post-fixed after timestamp and then followed by .gz then "_*" will take care of ".gz" extn as well.
- All incoming EDR files should contain the string as per their type in file name i.e.flow edr file should contain string "flow" (delimited by under-score) and http edr files shouldl contain string "http: (delimited by under-score).
- The input paths for EDR and Bulkstat must be under /data/collector
- Bulk Stats input path must end with the ASRGATEWAY.NAME
- After all configs in this MOP are completed and data is processed, the ASRGATEWAY.NAME and DC.NAME will be visible on UI after data is pro-cessed for them
- time zone name must be in the format as present in third column of fol-lowing file on the collector server /usr/lib64/python*/site-pack-ages/pytz/zoneinfo/zone.tab

Update Information Bases

Apply Configurations to the Collector

Once the above mentioned variable are present, then update the IBs/collector for the new gateway on master Collector node.

Log into the master Collector node and go to the `aggregation_center` subshell:

```
> en
# conf t
(config)# pmx subshell subshell-name
```

- To add a gateway, use following command:

```
pm extension (aggregation_center)> add gateway name <gateway-
name> region <gateway-region> location <gateway-area> schema_
version <bulkstat-schema-version> ip <gateway-IP> timezone
<gateway-timezone> edr-filename-pattern <incoming-EDR-
fileName-pattern> bulkstat-filename-pattern <incoming-BS-
fileName-pattern> edr-collector-filename-pattern <collector-
config-edr-filename-format> bulkstat-collector-filename-
pattern <collector-config-bs-filename-format> collector-ip
<collector-IPs> type <gateway-type> edr-file-path <incoming-
EDR-files-path-on-collector> bulkstat-file-path <incoming-BS-
files-path-on-collector>
```

  Note: See "Modifying Gateway Attributes" on page 123 for more information on gateway attributes and a sample of the output generated by this command.

- To view new gateway that has been added:

```
pm extension (aggregation_center)> show gateways
```

  Note: If gateway configuration show erroneous entry then it can be corrected by running the add gateway command, but replacing the incorrect values with the corrected ones.

- Push gateway configuration to all collectors:

```
pm extension (aggregation_center)> push gateway configuration
```

Verify Setup and Completion

1. Send traffic from the new ASR to the Virtual IP of the Collector cluster:
    a. Push data for both EDR and bulkstats to the directories.
    b. Wait a couple of hours so that the **mapReduce** job can process the Collector's data and the new data stats can be pushed to Insta.
2. Check the UI:
    a. Access UI as before and click **Today** to see traffic for the new DC added on the UI.
    b. Go to the bulkstats and KPI pages. From the gateway drop down menu, select the new gateway and verify that its data is shown on the UI.

Adding a UI Node to the Existing Cluster

This topic describes how to add another UI node to the existing DPI UI/Rubix cluster when DPI and small applications run on separate clusters.

Before You Begin

The new node which is being added should be on 3.5 release with all the relevant patches as per the latest Release Notes.

1. To verify the release and patch information, log into the GMS and go to the config terminal:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
```

2. Verify the version number:

```
host [cluster : master|standby](config)# show version
Product release:   3.7.1
```

3. Go to the pmx patch subshell:

```
host [cluster : master|standby](config)# pmx subshell patch
```

4. Verify the most recent patch:

```
pm extension (patch)> show patch_history
```

Building a New UI Node

1. Insert a new blade into the UCS Chassis.

2. Update the xml file and add details for the new UI node which is to be added to the existing UI cluster. Load the `mural.xml` to the GMS UI and edit it with the details of the new UI node.

   Note: The xml file is the same as what was used during bringing up the setup with GMS.

3. Edit the new UI/Rubix node entry under three tabs: **Server_Details**, **Nodes**, and **Clusters**.

Configuring the new node on the GMS

1. Navigate to the GMS interface from a browser on a machine from which the GMS is reachable:

   ```
   http://192.168.147.18/configure
   ```

   Where `192.168.147.18` is the IP Address assigned to the management interface of GMS server.

2. On the GMS UI, from the second dropdown select the "active" xml and click Load config file from the server. The configuration file is loaded into the GMS.

3. Add the details for the new UI node entry under 3 tabs: **Server_Details**, **Nodes**, and **Clusters**.

4. Provide these details:

   - **Nodes** tab—enter the node information, including management and control IP details.
   - **Clusters** tab—add the new UI node to the existing cluster
   - SSH to the GMS using the management IP and start the installation on all the new UI node.

5. Validate the XML for any errors. If validation is successful, save the XML on the server.

   a. SSH to the GMS using the management IP and start the installation on all the new UI nodes:

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   host [cluster : master|standby](config)# gms config
   mural.xml activate
   host [cluster : master|standby](config)# image fetch
   ```

```
http://192.168.0.17/release/atlas/atlas3.4.rc2/mfgcd-
atlas3.9.rc1.iso
host [cluster : master|standby](config)# image mount
mfgcd-atlas3.9.rc1.iso
```

The resulting output may resemble:

```
Copying linux...
Copying rootflop.img...
Copying image.img...
```

Note: The XML file is the same file as what was used during bringing up the setup with GMS.

Finishing the New UI Node

1. Reboot the new Rubix blade from KVM manager and press **F12** so as to boot it from network boot.
2. Once blades start booting from network, GMS will push the image on the blade using PXE boot and manufacture process will eventually start on blade.
3. Wait 30 minutes for the blade to be manufactured with image 3.5.

Applying Patches on the new UI Node

1. Log in to the GMS node and run the patch install script for this new node Management IP:

   ```
   > en
   # _shell
   # cd /data/
   # ls -l tmp_patch
   ```

   All the patches for the current release are listed. If patches do not exist, copy those to the /data/ directory and run the following command:

   ```
   # ./patch_install.sh node <NewUINodeIP>
   ```

2. Check the status of patches applied by the script on the new node:

```
> en
# conf t
(config) # pmx subshell patch
pm extension (patch)> show patch_history
```

Configuring the New UI Node

1. Establish an SSH connection to the GMS:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
```

2. Start installation on the Rubix nodes:

```
host [cluster : master|standby](config)# install appliance
cluster cluster-name RUBIX-CLUS-GMS
```

   Where *RUBIX-CLUS-GMS* is the name of the UI (Rubix) cluster which is used while configuring GMS. It comes up automatically when you press tab after typing `cluster-name`.

3. Monitor the installation status on the UI nodes (shows the percentage completed):

```
host [cluster : master|standby](config)# install appliance
show installation-status cluster RUBIX-CLUS-GMS
```

   The resulting output may resemble:

```
Node successfully installed.
```

4. Run the installation on the small application rubix cluster:

```
host [cluster: role] > en
host [cluster: role] # conf t
host [cluster: role](config) # install appliance cluster
cluster-name SMALL-RUBIX-CLUS
```

   Where:

`Cluster-name SMALL-RUBIX-CLUS` is the name of the second small application UI (Rubix) cluster that is used while configuring through GMS. It comes up automatically when you press the **Tab** key after typing the cluster-name.

5. Monitor the installation status on the this UI node cluster (displays the percentage completed):

   ```
   host [cluster: role](config) # install appliance show
   installation-status cluster SMALL-RUBIX-CLUS
   ```

   When complete, the output returns the message, `Node successfully installed`.

Pushing IBs and Certificates to New Node

1. Log into the master Collector node and go to the `aggregation_center` subshell:

   ```
   > en
   # conf t
   (config)# pmx subshell subshell-name
   ```

2. Push IBs to the new UI node:

   ```
    pm extension (aggregation_center)> add ib_destination
   Control_IP_Of_NewRubixNode
   pm extension (aggregation_center)> push all ibs Control_IP_Of_
   NewRubixNode
   ```

3. Run the `quit` command twice:

   ```
   pm extension (aggregation_center)> quit
   > quit
   ```

4. Log in to the standby Collector node and add the new UI node:

   ```
   host [cluster: standby]> en
   host [cluster: standby]# conf t
   host [cluster: standby](config)# pmx
   ```

```
pm extension> subshell aggregation_center
pm extension (aggregation center)> add ib_destination Control_
IP_Of_NewRubixNode
```

5. If single certificate files were installed on the UI nodes during the original system setup, restore these files on the new UI node.

6. Log in to any one of the old UI nodes and go to `_shell`:

```
host [cluster : master]> en
host [cluster : master]# _shell
```

7. Copy the keystore to the new UI node:

```
admin@host# scp /opt/tms/apache-tomcat/apache-tomcat-
7.0.27/keystore admin@control-ip-of-new-rubix-
node:/opt/tms/apache-tomcat/apache-tomcat-7.0.27/keystore
```

   Where `control-ip-of-new-rubix-node` is the IP address of the new Rubix node.

Starting Processes on New Node

1. Log into the master UI node:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
```

2. Start the atlas process:

```
host [cluster : master|standby] (config) # pm process rubix
restart
host [cluster : master|standby] (config) # rubix modify-app
atlas enable
host [cluster : master|standby] (config) # rubix modify-app
atlas modify-instance 1 enable
```

3. Log in to the standby UI node and repeat steps 1 and 2 to start all the applicable tomcat instances as configured.

4. Log in to the third UI node (newly added) and repeat steps 1 and 2 to start

all the applicable tomcat instances as configured.

5. Log in to the master UI Node of the second UI small apps cluster and start the tomcat instances:

   ```
   host [cluster : master|standby](config)# rubix modify-app
   process-name enable
   host [cluster : master|standby](config)# rubix modify-app
   process-name modify-instance 1 enable
   ```

   Where `process-name` is replaced with:

   - atlas
   - reportAtlas
   - bulkstats
   - rge
   - httperror
   - ruleEngine
   - launcher

   Note: Wait for two minutes between starting each process.

6. Log into the standby UI node and repeat the above steps to restart the tomcat instances.

7. Log in to the standby UI node and repeat preceding step to start all the applicable tomcat instances as configured.

Adding a Compute Node to the Existing Cluster

This topic describes how to add a Compute node to the existing Compute cluster.

Building a New Compute Node

1. Insert a new blade into the UCS chassis.
2. Add the new Compute node to the existing Compute node cluster:
   - Load `mural.xml` to the GMS UI and edit it with the details of the new Compute node.

     Note: The XML file is either the file which was originally used to configure the system, or the one which was last modified and saved for running another MOP.
3. Edit the compute node entry under three tabs: **Server_Details**, **Nodes**, and **Clusters**. Go to the GMS UI and do the following:
   a. Under **Slot**, click **Add** and provide the details for the compute node (slot number, host name, MAC IP address, and interface name).

   b. Under **Storage**, add the storage type WWIDs.

   Finding the WWID of LUNs

   1. Log into the EMC
   2. Click the **Storage** tab
   3. Click **LUNs**
   4. Highlight the destination LUN
   5. Click **Properties**

   The EMC Web UI shows the unique ID.

   For the WWID, remove the separator ':' from the unique ID and prefix the complete unique ID with 3, for example:

   ```
   360060160c7102f004887f4015d49e212
   ```

   c. Under **Nodes**, add the node (chassis logical name, slot number, and node hostname).

   d. Under **Node_IPdetails**, provide the management and Control IP address details as required in the setup.

   e. Under **Clusters**, add the new compute node as the member node to the compute node cluster.

4. Save the new xml file as `DN_recovery.xml`.

5. Validate the XML for any errors. If validation is successful, save the XML on server.

6. SSH to the GMS using the management IP and start the installation on all the new Compute nodes:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# gms config DN_
recovery.xml activate
```

   Note: The `DN_recovery.xml` file is the file you created after updating the original XML file with the details of the new Compute node.

7. Reboot the new Compute blade from KVM manager and press **F12** so as to boot it from network boot.

8. Once blades start booting from network, GMS will push the image on the blade using PXE boot and manufacture process will eventually start on blade.

   Wait 30 minutes for the blade to be manufactured with image 3.5.

9. Log in to the GMS node and run the patch install script for this new node Management IP:

```
> en
# _shell
# cd /data/
# ./patch_install.sh node <NewComputeNodeIP>
```

10. Stop processes that run on Name nodes:

    a. Stop jobs on the master Collector (NameNode):

    ```
    > en
    # conf t
    # pmx subshell oozie
    pm extension (oozie)> stop jobname all
    pm extension (oozie)> show coordinator RUNNING jobs
    ```

    The output of the `show` command must not display any job in the `Running` state.

    b. Run the following command to obtain the data start time as configured on the system:

    ```
    pm extension (oozie)> show config dataset atlas_edrflow_1
        Attributes:
    -----------
    doneFile     : _DONE
    endOffset    : 1
    frequency    : 5
    outputOffset : 0
    path         :
    /data/collector/1/output/edrflow/%Y/%M/%D/%H/%mi
    pathType     : hdfs
    startOffset  : 12
    startTime    : 2015-04-28T07:50Z
    ```

    Note the value against the “startTime” parameter to use it later.

    c. Terminate Collector on all master and standby Collectors:

    ```
    > en
    # conf t
    # pm process collector terminate
    ```

    d. Check the file name format configured in collector and note them for later use:

```
> en
# _shell
#  cli -t "en" "sho run" | grep filename-format | awk '
{print $5" -> "$NF}'
bulkStats -> *_%YYYY%MM%DD%hh%mm%ss
edrflow -> %DC_*_*_%MM%DD%YYYY%hh%mm%ss_*.gz
edrhttp -> %DC_*_*_%MM%DD%YYYY%hh%mm%ss_*.gz
```

11. Establish an SSH connection to the GMS and start installation on the Compute node:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# install appliance
cluster cluster-name ComputeNode-CLUS
```

12. Monitor the installation status on the Compute node by executing this command to show the percentage completed:

```
host [cluster : master|standby](config)# install appliance
show installation-status cluster ComputeNode-CLUS
```

When complete, the output shows the message:

```
Node successfully installed
```

13. Install the Namenode cluster:

```
> en
# conf t
(config) # install appliance cluster cluster-name <Namenode-
CLUS>
```

14. Monitor the installation status on the Name node to show the percentage completed:

```
(config) # install appliance show installation-status cluster
<Namenode-CLUS>
```

Checking the New Compute Node

Verify that the new Compute node was successfully added to the Hadoop Cluster.

1. Log in to the master NameNode and check the status of the new Compute node which has come up:

```
admin@host# hdfs dfsadmin -report
```

The resulting output may resemble:

```

Configured Capacity: 2223700070400 (2.02 TB)
Present Capacity: 2107313029120 (1.92 TB)
DFS Remaining: 2017122156544 (1.83 TB)
DFS Used: 90190872576 (84.00 GB)
DFS Used%: 4.28%
Under replicated blocks: 135
Blocks with corrupt replicas: 0
Missing blocks: 0

-------------------------------------------------
Datanodes available: 3 (3 total, 0 dead)

Live datanodes:
Name: 192.168.151.182:50010 (UCS-DN1-151-182)
Hostname: UCS-DN1-151-182
Rack: /Chassis-1
Decommission Status : Normal
Configured Capacity: 744050839552 (692.95 GB)
DFS Used: 45286723584 (42.18 GB)
Non DFS Used: 38530887680 (35.88 GB)
DFS Remaining: 660233228288 (614.89 GB)
DFS Used%: 6.09%
DFS Remaining%: 88.73%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Last contact: Fri Dec 26 08:19:52 UTC 2014


Name: 192.168.151.139:50010 (UCS-DN3-151-139)
Hostname: UCS-DN3-151-139
Rack: /Chassis-2
Decommission Status : Normal
Configured Capacity: 739824615424 (689.02 GB)
DFS Used: 1849409536 (1.72 GB)
Non DFS Used: 39434940416 (36.73 GB)
DFS Remaining: 698540265472 (650.57 GB)
DFS Used%: 0.25%
DFS Remaining%: 94.42%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Last contact: Fri Dec 26 08:19:52 UTC 2014


Name: 192.168.151.164:50010 (UCS-DN2-151-164)
Hostname: UCS-DN2-151-164
Rack: /Chassis-2
Decommission Status : Normal
Configured Capacity: 739824615424 (689.02 GB)
DFS Used: 43054739456 (40.10 GB)
```

```
Non DFS Used: 38421213184 (35.78 GB)
DFS Remaining: 658348662784 (613.13 GB)
DFS Used%: 5.82%
DFS Remaining%: 88.99%
Configured Cache Capacity: 0 (0 B)
Cache Used: 0 (0 B)
Cache Remaining: 0 (0 B)
Cache Used%: 100.00%
Cache Remaining%: 0.00%
Last contact: Fri Dec 26 08:19:53 UTC 2014
```

Note: The above output should list the new Compute node with a `Normal` status.

2. Log into the newly added Compute node and check that the processes are running:

```
# ps -ef| egrep -i "datanode|_nodemanager" |grep -v grep |wc -
l
# mount | grep hadoop-admin
/dev/mapper/mpathep1 on /data/hadoop-admin type ext3
(rw,relatime,errors=continue,barrier=1,data=ordered)
```

Before running the mount command, ensure that the following processes are running:

- /usr/java/default/bin/java -Dproc_datanode
- /usr/java/default/bin/java -Dproc_nodemanager

Start Processes on Collectors (Name Nodes)

1. Add the gateways through the `aggregation_center` subshell:

   Before running the commands to add gateways, obtain the gateway name, DC name, Region, Location, IP, timezone, edr-filename-pattern, type, edr-file-path, bulkstat-filename-pattern and bulkstat-file-path from the `gate-way.conf` file.

The `gateway.conf` file is available at `/data/configs/gateway/`. It contains a block enclosed with curly braces for each gateway. Within that block , the “parameter” : “value” can be found for all the mentioned parameters.

Note: The `/data/collector` string must be removed from the edr-file-path parameter value and then the remainder string must be used in the add gateway command.

For example:

```
{
"Name": "DC1",
"Associated Region": "R1",
"IP": "192.168.151.87",
"Dc": "DC1",
"file_pattern": "*_*_*_%MM%DD%YYYY%hh%mm%ss_*.gz",
"Location": "A1",
"Schema Version": "17",
"timezone": "UTC",
"input_dir": "/data/collector/DC1/edr/",
"Type": "HA",
"output_dir": {
 "http": "/data/collector/edrhttp",
 "flow": "/data/collector/edrflow",
 "asn": "/data/collector/edrAsn"
 }
},
 {
 "Name": "DC1","Associated Region": "R1","IP":
"192.168.151.87","Dc": "DC1","file_pattern": "*_%YYYY%MM%DD%h
h%mm%ss","Location": "A1","Schema Version": "17","timezone":
"UTC","input_dir": "/data/collector/DC1/bs/","Type": "HA","out-
put_dir": {"DC1": "/data/collector/bulkstats_files/DC1"}},
```

The `add gateway` command in the succeeding steps use values from this block for the gateway DC1.

For parameters “edr-collector-filename-pattern” and “bulkstat-collector-filename-pattern” to be given in the `add gateway` command, use the file-name format values saved in the step 10 (d) in "Building a New Compute Node" on page 105.

For the “collector-ip” parameter in the `add gateway` command, provide a comma separated list of the collector IPs (use Data Network IPs if present otherwise the management network IP) that will be receiving the data from this gateway.

```
> en
# conf t
# pmx subshell  aggregation_center
pm extension (aggregation center)> add gateway name DC7 region
R7 location A7 schema_version 18 ip 192.168.151.93 timezone
UTC edr-filename-pattern *_*_*_%MM%DD%YYYY%hh%mm%ss_*.gz
bulkstat-filename-pattern *_%YYYY%MM%DD%hh%mm%ss edr-
collector-filename-pattern %DC_*_*_%MM%DD%YYYY%hh%mm%ss_*.gz
bulkstat-collector-filename-pattern *_%YYYY%MM%DD%hh%mm%ss
collector-ip 192.168.193.225,192.168.192.178 edr-file-path
/DC7/edr/ bulkstat-file-path /DC7/bs/ type HA
```

Repeat this step for each unique gateway name and its corresponding parameter values present in the gateway.conf file.

2. Update the timezone to UTC for BulkStats in the `gateway.conf` file on the master Collector node and Namenode:

```
# _shell
# vi /data/configs/gateway/gateway.conf
# cli -m config
(config) # pmx subshell aggregation_center
pm extension (aggregation center)> push gateway configuration
```

3. Generate and push the IBs:

```
pm extension (aggregation center)> generate all ibs
pm extension (aggregation center)> push all ibs
pm extension (aggregation center)> quit
pm extension > subshell bulkstats
pm extension (bulkstats)> generate all ibs
pm extension (bulkstats)> push all ibs
pm extension (bulkstats)> quit
```

If anomaly is enabled:

```
pm extension > subshell anomaly
pm extension (anomaly) > update all ibs
```

4. Add the ASR user on both the master and standby collectors again:

```
> en
# conf t
(config) # username username password password
(config) # wri mem
```

Repeat this step on the newly added collector cluster nodes.

5. Check if the collector is running or not on both the master and standby Collectors:

```
> en
# _shell
# cli -t "en" "conf t" "show pm process collector" | grep -i
status
Current status:  running
```

6. Log in to master GMS node and run the setOozieTime script for both the existing and old Collector (Name) nodes:

```
> en
# _shell
# cd /opt/deployment/Mural_setStartTime/
# ./setOozieTime --dataStartTime <data_start_Time> --node <IP_
GCU_NODE>  --password admin@123
```

Note: Use the data start time as noted in the step 10 (b) of the "Building a New Compute Node" on page 105 section.

Repeat the step with IP of the standby Collector (Name) node.

7. Start the jobs on the master Collector (Namenode):

```
> en
# conf t
# pmx subshell oozie
# run job all
```

Manually Manufacturing the Master GMS Blade

This topic describes a method for manufacturing the master GMS blade. Perform the procedures in this topic only if you are not using the standard method found in the *MURAL Installation Guide*.

The master General Management System (GMS) blade hosts the master GMS node, a platform component that enables centralized installation and monitoring of all the blades on the system.

Follow these steps to manufacture (install the operating system software on) the master GMS blade.

Before you begin:

- Configure Serial over LAN (SOL) on all the blades during EMC setup.
- Locate your CIQ, and refer to it for such details as UCS access credentials and KVM SOL IP address.

To manufacture the master GMS blade, perform the following steps:

1. Download the ISO image included with the software package to the machine from which you will access the Cisco UCS blades.

   The ISO image filename is **/data/mfgcd-guavus-x86_64-20140731-005012.iso**

   To verify the MD5 checksum of the image, run the **md5sum *filename*** command.

   ```
   # md5sum /data/mfgcd-guavus-x86_64-20140731-005012.iso
   48fc39f77bc7ab8847ca2f7a684d2ace /data/mfgcd-guavus-x86_64-20140731-
   005012.iso
   ```

2. Open the Cisco UCS - KVM Launch Manager in a browser and enter your login credentials.

   Note: For best performance, access the KVM Launch Manager in Firefox with Java version 6 or greater.

The UCS - KVM Launch Manager application opens and displays all blades available on the chassis.

3. Click the **Launch** button for the first node (**Server1** in the following figure).

   Click **OK** to download and open the **kvm.jnlp** file.

   Click **OK** to clear the keyboard access warning message that appears.

   The console for the port opens.

4. Navigate to **KVM Console > Virtual Media**, click **Add Image**, and specify the path of the ISO image that you downloaded in Step 1.
5. Click the check box in the **Mapped** column for the added ISO image, which is then mounted.
6. Reboot the blade to use the newly mounted image. Access the **KVM** tab and select **Ctrl-Alt-Del** from the **Macros > Static Macros** drop-down menu.
7. When the boot screen appears, press **F6** to select the boot menu.
8. Select **Enter Setup** to open the setup utility.
9. On the **Boot Options** tab, verify that the value in the **Boot Option #1** field is ***CD/DVD** as shown in the following figure. If you change the value, press the **F10** key to save and exit; if the value is already correct, press the **Esc** key to exit.
10. To change the boot order to set CD/DVD as the first priority, from the left menu select **Virtual Media > Boot Order** and under **Server/Boot Order**, select **CD/DVD Drive**, as shown in the following figure.
11. To open the serial console, from the left menu of the HP iLO interface, select **Remote Console**, and then from the **Launch** tab, click the **Launch** button under **Java Integrated Remote Console**.

    The remote console opens.

12. To open the Virtual Media applet, select Virtual Drives > CD/DVD and attach the ISO image, as shown in the following figure.

13. Verify that the Virtual Media light, located at the bottom of the remote console window turns green, as shown in the following figure.

14. From the remote console, reboot the server.

15. Select Virtual CD/DVD so the blade boots with the mounted ISO image.

16. At the **#** prompt, run the **manufacture** command to manufacture the master GMSGCU blade.

    The following command is appropriate for a single disk configured as RAID 1, as indicated by the **-L 1D** argument. The master GMSGCU blade has a second disk that functions as the mirrored disk in the RAID configuration.

```
# manufacture.sh -v -t -f /mnt/cdrom/image.img –m 1D –L 1D --cc no
--cs no --cl no –a
```

    Output similar to the following traces the manufacturing process.

```
Running  /etc/in it .d/rcS .d/S34automf g
— Automatic manufacture is not enabled. Type ‘automf g’ to start it.

BusyBox v1.00 (2010.12.03—23:16+0000) Built—in shell (ash)
Enter ‘help’ for a list of built—in commands.

—sh: can’t access tty: job control turned off

Processing  /etc/profile... Done

# manufacture.sh —v —t —f /mnt/cdrom/image.img
======  Starting manufacture at YYYYMMDD—hhmmss
======  Called as: /sbin/manufacture.sh —v —t —f /mnt/cdrom/image.img
===============================================
Manufacture script starting
===============================================
----------------------------------
Model selection
----------------------------------
```

```
Product model (‘?‘ for info) ( DESKTOP VM VM_2D 1D 2D 3D 4D 2D_EXT4)
[2D]: 2D

== Using model: 2D

----------------------------------
Kernel type selection
----------------------------------
Kernel type (uni smp) [smp]:
== Using kernel type: smp
----------------------------------
Layout selection
----------------------------------
Layout (STD) [2D]:
== Using layout: 2D
----------------------------------
Partition name—size list selection
----------------------------------

Partition name—size list [VAR 40960  SWAP 40960 ]:
== Using partition name—size list: VAR 40960  SWAP 40960
----------------------------------
Device list selection
----------------------------------
Device list [/dev/sda   /dev/sdb]
== Using device list: /dev/sda   /675dev/sdb

----------------------------------
Interface list selection
----------------------------------
Interface list [eth0  eth1]:
== Using interface list: eth0  eth1

----------------------------------
Interface naming selection
----------------------------------
Interface naming [none]:
```

```
== Using interface naming: none
== Smartd enabled

----------------------------------
CMC server settings
----------------------------------
Enable CMC server (yes no) [yes]: no
Enable CMC server (yes no) [yes]: no
== CMC server enabled: no

----------------------------------
CMC client settings
----------------------------------
Enable CMC client (yes no) [yes]: no
== CMC client enabled: no
== CMC client auto—rendezvous enabled: no
== CMC server address for rendezvous: (none)

----------------------------------
Cluster settings
----------------------------------
Enable cluster (yes no) [no]: no
== Cluster enable: no
Cluster ID:

Number Start End Size File system Mame Flags
1 O.OZMiB 30000MiB 30000MiB ext3 primary
Writing partition table to DISICZ
Disk devsdb: S7Z3Z6MiB
Sector size (logicalphysical): 512B,51ZB
Partition Table: gpt
Number Start End Size File system Mame Flags
1 O.OZMiB 5?Z3Z6MiB 5?ZJZ6MiB ext3 primary

==== Making filesystems
== Creating  ext3  filesystem on /dev/sda2 for BOOT1
== Creating  ext3  filesystem on /dev/sda3 for BOOT2
== Creating  ext3  filesystem on /dev/sda1 for BOOTMGR
== Creating  ext3  filesystem on /dev/sda8 for CONFIG
== Nothing  to do on /dev/sda9  for  HA
== Creating  ext3  filesystem on /dev/sda5 for ROOT1
== Creating  ext3  filesystem on /dev/sda6 for ROOT2
== Making swap on /dev/sda7  for SWAP
Setting up swapspace version 1, size = 42952404 kB
== Creating  ext3  filesystem on /dev/sda10 for VAR

== CMC server address for rendezvous: (none)

----------------------------------
Cluster settings
----------------------------------
== Cluster  enable: no
Cluster ID:
== Cluster ID: (none)
Cluster  description:
== Cluster  description: (none)
Cluster interface:
== Cluster  interface: (none)
Cluster master virtual IP address [0.0.0.0]:
== Cluster  master  virtual  IP address: 0.0.0.0
Cluster master virtual IP masklen [0]:
== Cluster master virtual IP masklen:  0
Cluster shared secret :
== System successfully imaged
—— Writing Host ID: 3b0455ef813d
== Zeroing the destination partition disk /dev/sda9 with dd
== Calling imgverify to verify manufactured system
== Using layout: ZD
Using dev list:  /dev/sda /dev/sdb
== Verifying image location 1
=== Mounting partitions
=== Checking manifest
=== Unmounting  partitions
=== Image location 1 verified successfully.
```

```
== Verifying image location 2
=== Mounting partitions
=== Checking manifest
=== Unmounting partitions
=== Image location 2 verified successfully.
== Done
======  Ending manufacture at YYYYMMDD—hhmmss
—— Manufacture done.
#
```

17. In the UCS-KVM Launch Manager, navigate to **KVM Console > Virtual Media** and click the check box in the **Mapped** column of the ISO image file, to remove the check and unmount the file.

18. Run the **reboot** command to reboot the blade with the new ISO image.

```
# reboot
```

19. Use SSH to log in to the master GMSGCU blade as user **admin**.

Modifying and Configuring Components

Modifying Gateway Attributes

Before creating or modifying gateway attributes, familiarize yourself with these values, their meanings, and formatting guidelines:

Names

- **`gateway-name`**—After all configurations are completed and data is processed, this name is visible on the UI for the bulkstats data sent from this gateway.
- **`gateway-timezone`**—Timezone where gateway is physically located and the timezone in which ASR is going to send the timestamp in the filename. Must be in the format as present in third column of following file on the collector server `/usr/lib64/python*/site-packages/pytz/zoneinfo/zone.tab`
- **`region-name`**—Region name to which this gateway belongs.

Filename Formats

ASR should send gateway name in-place of `%DC` as specified in filename pattern in collector configuration.

Filename patterns should be in sync with Collector configurations.

- **`edr-filename-format`**—incoming-EDR-fileName-pattern. You can print existing file name formats for both EDR and bulkstats.

File Paths

Paths should always start with "/" and the bulkstats input path must end with the *`gateway-name`*. Input directories are always created under the `/data/collector` path.

- **`gateway-edr-file-path`**—Local directory on collector where this new ASR will drop the EDR record files.

Incoming Files

If the file names are going to come with ".gz" or ".txt" at the end with no other special character in between timestamp & ".gz", then it is mandatory to append ".*" in the filename format while adding gateway. But if "_" is postfixed after timestamp and then followed by .gz then "_*" will take care of ".gz" extension as well.

- **EDR**—should contain the string as per their type in file name i.e. flow edr file should contain string "flow" (-delimited by underscore) and http edr files should contain string "http: (delimited by underscore)
- **Bulkstats**—should contain gateway name delimited by "_". For example, `GMPLAB1_20131006121500`.

Other Attributes

- **`bulkstat-schema-version`**—The version number of bulkstats schema that was used to configure the gateway.
- **`gateway-ip`**—The ip address designated for the gateway.
- **`gateway-type`**—Indicate the kinds of files or data which will be flowing through this gateway.

Adding a Gateway

1. Log into the master Collector node and go to the pmx `aggregation_center` subshell:

```
> en
# conf t
(config)# pmx subshell subshell-name
```

2. Add a gateway by running:

```
pm extension (aggregation_center)> add gateway name <gateway-
name> region <gateway-region> location <gateway-area> schema_
version <bulkstat-schema-version> ip <gateway-IP> timezone
<gateway-timezone> edr-filename-pattern <incoming-EDR-
fileName-pattern> bulkstat-filename-pattern <incoming-BS-
```

```
fileName-pattern> edr-collector-filename-pattern <collector-
config-edr-filename-format> bulkstat-collector-filename-
pattern <collector-config-bs-filename-format> collector-ip
<collector-IPs> type <gateway-type> edr-file-path <incoming-
EDR-files-path-on-collector> bulkstat-file-path <incoming-BS-
files-path-on-collector>
```

Where `gateway-edr-file-path` is:

- `/California/edr111/edr` if the ASR is in Charging mode
- `/California/edr111/redr` if the ASR is in reporting mode

For example:

```
pm extension (aggregation_center)> add gateway name GMPLAB1
 region EAST
 location USA
 schema_version 15
 ip 10.10.10.255
 timezone America/Recife
 edr-filename-pattern *_MURAL-edr_*_%MM%DD%YYYY%hh%mm%ss.*
 bulkstat-filename-pattern bulkstats_%YYYY%MM%DD%hh%mm%ss
 type HA
 edr-file-path /California/edr111/edr
 bulkstat-file-path /California/bs111
```

Note: In the above example the absolute path configured in `/data/-collector/gateway/gateway.conf` and the collector configuration will be `/data/collector/California/edr111/edr` and `/data/-collector/Carlifornia/bs111`.

The resulting output may resemble:

```
Adding IBs....
***************Adding ib dcRegionArea*******************
Adding in dcRegionArea.map
[dcRegionArea.map]:
generated dc.id
generated region.id
generated area.id
generated dcRegionArea.id.map
Summary:
========
Successful IBs : 1 out of 1
Failed IBs : No id generation failures.
pushing ib [dc.id] to all IB destinations
pushing ib to ip 192.168.151.173
...
dc.id                          100%  133     0.1KB/s   00:00
...
Summary:
========
Successful Transfers : 4 out of 4
Failed Transfers : No transfer failures.
pushing ib [region.id] to all IB destinations
pushing ib to ip 192.168.151.173
...
region.id                      100%   86     0.1KB/s   00:00
*******************Added ib dcRegionArea*********************
********************Adding ib gateway************************
Adding in gateway.map
[key.map]:
generated key.id.map
[gateway.map]:
generated gateway_bs.id
generated version_bs.id
generated dc_bs.id
...
Summary:
========
Successful IBs : 5 out of 5
```

```
Failed IBs : No id generation failures.
pushing all ibs to all IB destinations
pushing ib to ip 192.168.151.173
...
gatewayIDVersion.map           100%  113     0.1KB/s   00:00
schemaIDToKeyID.map            100% 8018     7.8KB/s   00:00
Gateway_Schema_ASR5K.map       100% 1779KB   1.7MB/s   00:00
Gateway_Schema_GMPLAB1.map     100% 1779KB   1.7MB/s   00:00
Gateway_Schema_GMPLAB2.map     100% 1779KB   1.7MB/s   00:00
Gateway_Schema_NewDelhi.map    100% 1779KB   1.7MB/s   00:00
Gateway_Schema_gmplab1.map     100% 1779KB   1.7MB/s   00:00
Gateway_Schema_gmplab2.map     100% 1779KB   1.7MB/s   00:00
gatewaySchemaMetric.map        100%   10MB  10.4MB/s   00:00
key.id.map                     100% 1375     1.3KB/s   00:00
...
_DONE                          100%    0     0.0KB/s   00:00
Summary:
========
Successful Transfers : 4 out of 4
Failed Transfers : No transfer failures.
**********************Added ib gateway***********************
Adding Gateway configs....
***********************Gateway*******************************
Name: gmplab2
Associated Region: EAST
Location: USA
Schema Version: 14
IP: 10.10.10.255
Timezone: America/Recife
Flow-EDR/Http-EDR Filename Pattern: %DC_MURAL-edr_
*_%MM%DD%YYYY%hh%mm%ss*.gz
Bulkstat Filename Pattern: *_%YYYY%MM%DD%hh%mm%ss
Type: HA
```

```
Flow-EDR/Http-EDR Filename Path:
/data/collector/California/edr111/edr
Bulkstat Filename Path: /data/collector/Graham/bs111
*****************Successfully Added********************
```

If any of the command entries are not correct then one can correct as per my earlier comment. <The one I get from Tony.>

Overriding Categorizations From Information Bases

You can specify for the system to override the automatic classification of incoming data records for device type (based on user agent or TAC substring), application (based on user agent), or service provider used from the URL. You can also use the same procedure to add categories for entities that the system is currently classifying as uncategorized. Any SP, App, Protocol, or Device can also be renamed through the steps mentioned in this section and the same procedure can be followed to update their categories as well.

Any subscriber ID can be marked as Top Subscriber, to view its data in the UI.

The system categorizes entities according to the definitions contained in the information bases. Information bases are updated whenever the data mining team determines that there has been a major change in the available information and the information that is made available in the next release or patch release. You might want to override the information base and change classifications in order to:

- Override any device or App based on the User Agent. If multiple user agents exist for a particular device or app, any common string in all the UAs can be added to override.
- Override any SP based on URL. If multiple URLs exist for same SP, a common string can be added to override.
- Override any Device based on TAC.
- Override Categories for any SP, App, or Protocol.
- Rename any SP, App, and Protocol Category.
- Rename any SP, App, Protocol, and Devices.
- Add any subscriber ID to be treated as Top Subscriber.

Note: If there are certain user agents, URL and TACs that make up a large percentage of the overall traffic and are not being categorized by the system, share the uncategorized reports with the Customer Support team so that they can be added in the next update of the information bases.

The following is an example of a list of categories to override:

```
SP category overriding
SP: craigslist.org
Default category: Business
Overriding Category: Shopping

App category overriding
App: Background Intelligent Transfer Service (BITS)
Default category: Business
Overriding Category: Productivity

Protocol category overriding
Protocol: HTTPS
Default category: Web
Overriding Category: Tunneling



UA Mobile App Overriding
Actual UA: Microsoft BITS/7.5
UA: BITS
Default Category: Business
Default APP: Background Intelligent Transfer Service
(BITS)
Overriding Category: Productivity
Overriding APP: BITS

UA Device Model Overriding
Actual UA: Mozilla/5.0 (Linux; U; Android 2.3.3; en-us;
LG-VS700
Build/GINGERBREAD) AppleWebKit/533.1 (KHTML, like Gecko)
Version/4.0 Mobile Safari/533.1
UA: LG-VS700
Default Device Manufacturer : LG
Default Device Model : LG VS700 Gelato Q
Overriding Device Manufacturer : LG
Overriding Device Model : Gelato

TAC Device Model Overriding
TAC : 98005800
Default Device Manufacturer : RIM
Default Device Model : RIM BlackBerry 9650 Bold Tour 2
Overriding Device Manufacturer : BlackBerry
Overriding Device Model : BB Bold


SP category renaming
Default category: Finance
New Name: Commerce

App category renaming
Default category: Music
Overriding Category: Songs

Protocol category renaming
Default category: Tunneling
Overriding Category: VPN

SP renaming
Default name: 9msn.com
New Name: ninemsn.com

App renaming
Default name: AFL 2012
New name: AFLLive

Protocol renaming
```

```
Default name: HTTP-Miscellaneous
New Name: HTTP

Device renaming
Default name: Samsung I9300 Galaxy S III
New Name: Samsung Galaxy S3
```

Bulk Update Feature for Overriding and Renaming Categorizations

You can also use the bulk update feature to add entries in the override and rename IBs as adding entries manually is tiresome and prone to error. The bulk update feature takes an offline CSV file as the input. You can use CLI to point to this CSV file and the system validates the entries and discards any row that has invalid entries.

Perform the following steps:

1. Execute the following commands:

```
>en
#_shell
# pmx subshell aggregation_center
(aggregation_center) # update all ibs from image
```

2. Copy the required sample files available at `/opt/catalogue/atlas` to any location in the `/data` folder.

   The following sample files are available:

   - override_app_from_ua
   - override_device_from_ua
   - override_protocolcat_from_protocol
   - override_urlcat_from_sp
   - override_appcat_from_app
   - override_sp_from_url
   - override_device_from_tac
   - rename_template

Note: Existing entries in the override and rename IBs available at `/data/ib/inbox` will not be impacted by the bulk update.

3. Execute the following commands to modify the required sample files:

   - To override app from ua, execute the following command:

     ```
     pm extension (aggregation center)> override add app from
     ua in bulk File path: /data/override_app_from_ua
     ```

   - To override device from ua, execute the following command:

     ```
     pm extension (aggregation center)> override add device
     from ua in bulk File path: /data/override_device_from_ua
     ```

   - To override protocolcat from protocol, execute the following command:

     ```
     pm extension (aggregation center)> override add
     protocolcat from protocol in bulk File path:
     /data/override_protocolcat_from_protocol
     ```

   - To override urlcat from sp, execute the following command:

     ```
     pm extension (aggregation center)> override add urlcat
     from sp in bulk File path: /data/override_urlcat_from_sp
     ```

   - To override appcat from app, execute the following command:

     ```
     pm extension (aggregation center)> override add appcat
     from app in bulk File path: /data/override_appcat_from_
     app
     ```

   - To override sp from url, execute the following command:

     ```
     pm extension (aggregation center)> override add sp from
     url in bulk File path: /data/override_sp_from_url
     ```

   - To override device from tac, execute the following command:

```
pm extension (aggregation center)> override add device
from tac in bulk File path: /data/override_device_from_
tac
```

- To add enteries in bulk in rename IB, execute the command:

```
pm extension (aggregation center)>rename in bulk File
path: /data/rename_template
```

4. Execute the following commands in the aggregation_center subshell:

```
(aggregation_center) # generate all ibs
(aggregation_center) # push all ibs
(aggregation_center) # quit
```

Before You Begin

Before you begin, you must create the list of service providers, user agents, type approval codes (TACs), and service providers (SPs) that you want to override or add to the IB.

For information about viewing the uncategorized UAs, URLs and TACs, see "Viewing Reports for Uncategorized Entities" on page 69.

Overriding Categories

For category overriding, user must know the exact name of SP, MobileApp, or Protocol and category that user wants to see in the UI. SP, App, and Protocol that is added are case insensitive, but spaces must be exact as returned by the URL Cat.

To override or add category mappings in information bases:

1. Log into the master Collector node (NameNode).
2. Go to the PMX shell and `aggregation_center` subshell:

```
> en
# conf t
(config)# pmx subshell subshell-name
```

3. Edit the information base for SP category:

```
pm extension (aggregation_center)> override add urlcat from sp

Service Provider : facebook.com
URL Category : Facebook
Added override property
```

To add this entry in the SP list:

```
pm extension (aggregation center)> edit ib category.list add
URL Category: category-name
```

For example, *category-name* can be **Facebook**.

4. Edit the information base for App category:

```
pm extension (aggregation center)> override add appcat from
app
Mobile Application Name : BlackBerry
Mobile Application Category : BlackBerry Broswer
Added override property
```

To add this entry in App Category list:

```
pm extension (aggregation center)> edit ib
mobileappcategory.list add
Mobile Application Category: BlackBerry Browser
```

5. Edit the information base for Protocol category:

```
pm extension (aggregation center)> override add protocalcat
from protocol
Protocol : Facetime
Protocol Category : IM
updating categoryTT.list
Added override property
```

To add this entry in the Protocol category list:

```
pm extension (aggregation center)> edit ib categoryTT.list add
Mobile TT Application Category: IM
```

6. Generate and push all information bases:

```
pm extension (aggregation_center)> generate all ibs
.
.
.
Summary: ========
Successful IBs : 27 out of 27
Failed IBs : No id generation failures.
```

7. Push all information bases:

```
pm extension (aggregation_center)> push all ibs
Checking if there are pending transfers...
Sending 131 ibs to 3 receivers (using 3 threads, skipping 0)
***
 running fetch all ibs from inbox on standby node :
192.168.151.82



Acceess restricted to authorised users only. Welcome to medium
pack MURAL3_5 system

Summary:
========
Failed Transfers     : 0 out of 3
Successful Transfers : 3 out of 3
```

Note: You are not required to restart jobs as the system picks up the updated information bases the next time the job runs (for the next hour's data).

Renaming SP, App, Protocol, or Device

To rename various SP, App, Protocol, or Device, add new names to the corresponding static whitelist. Older values can be case insensitive but spaces or extra characters should be exactly the same.

To rename a device:

1. Run the following command in the `(aggregation center)` subshell:

```
pm extension (aggregation center)> rename add device
Rename For Device
Old Value: Samsung Galaxy Nexus
New Value: Nexus
updating model.list_static
Added rename property
```

2. Add model entry in the model static list:

```
pm extension (aggregation center)> edit ib model.list_static
add
Device: Nexus
```

3. Add or modify its entry in the device_man_dg_os.map file. If it already exists use "modify" action; else use "add" action to create an entry.

```
pm extension (aggregation center)> show ib device_man_dg_
os.map
1       [Samsung Galaxy Nexus][Samsung][SmartPhone][Android]
2       [Apple iPhone 6]      [Apple]  [SmartPhone][iOS]
pm extension (aggregation center)> edit ib device_man_dg_
os.map modify record 1
Device Model:Nexus
Manufacturer:Samsung
Device Group:SmartPhone
OS:Android
IB [device_man_dg_os.map] updated.
```

Or, run the following command:

```
pm extension (aggregation center)> edit ib device_man_dg_
os.map add
Device Model:Nexus
Manufacturer:Samsung
Device Group:SmartPhone
OS:Android
IB [device_man_dg_os.map] updated.
```

To rename a service provider:

1. Run the following command:

```
pm extension (aggregation center)> rename add sp
Rename For Service Provider
Old Value: craigslist.org
New Value: clist.com
Added rename property
```

   To add its entry in SP static whitelist:

```
pm extension (aggregation center)> edit ib sp.list_static add
Service Provider: clist.com
```

To rename an application:

1. Run the following command:

```
pm extension (aggregation center)> rename add app
Rename For Application
Old Value: Firefox
New Value: Mozilla FireFox
Added rename property
```

   To add its entry in Mobile App Static whitelist:

```
pm extension (aggregation center)> edit ib mobileappname.list
add
Mobile Application Name: Mozilla FireFox
```

To rename a protocol:

1. Run the following command:

```
pm extension (aggregation center)> rename add protocol
Rename For Protocol
Old Value: gadugadu
New Value: gadu gadu
Added rename property
```

To add its entry in Protocol Static whitelist:

```
pm extension (aggregation center)> edit ib protocolTT.list_
static add
Mobile Application Name: gadu gadu
```

After renaming the entries, perform the following steps:

1. Run the following command to check these entries:

```
pm extension (aggregation center)> show ib rename_ib
1       DEVICE
2       Samsung Galaxy Nexus
3       Nexus
4
5       PROTOCOL
6       gadugadu
7       gadu gadu
8
9       APP
10      Firefox
11      Mozilla FireFox
12
13      SP
14      craigslist.org
15      clist.com
16
```

2. Generate and push all information bases:

```
pm extension (aggregation center)> generate all ibs
.
.
.
Summary: ========
Successful IBs : 27 out of 27
Failed IBs : No id generation failures.
pm extension (aggregation center)> push all ibs
Checking if there are pending transfers...
Sending 131 ibs to 3 receivers (using 3 threads, skipping 0)
***
running fetch all ibs from inbox on standby node :
192.168.151.82
Acceess restricted to authorised users only. Welcome to medium
pack MURAL3_5 system
Summary:========
Failed Transfers     : 0 out of 3
Successful Transfers : 3 out of 3
```

Note: You do not need to restart jobs as the system picks up the updated information bases the next time the job runs (for the next hour's data).

Overriding Mobile Devices by User Agents

To override a mobile device based on the user agent (UA), user must know either entire UA or any substring that is part of UA.

1. If all UA containing “Tbox” should be annotated as a particular device, type “Tbox”. For example:

```
> en
# conf t
(config) # pmx
Welcome to pmx configuration environment.
pm extension> subshell aggregation_center
pm extension (aggregation center)> override add device from ua
```

```
User Agent : Tbox
Device : Telstra Box
Manufacturer : Telstra
Device Group : Mobile Router
OS : Others
updating device_man_dg_os.map
Added override property
User Agent          Device       Manufacturer  Device Group
OS
----------------------------------------------------------
Tbox                Telstra Box  Telstra       Mobile Router
Others
```

2. To add the manufacturer, OS, and device in the corresponding list, run the following commands:

```
pm extension (aggregation center)> edit ib model.list_static
add
Device: Telstra Box
```

```
pm extension (aggregation center)> edit ib manufacturer.list
add
Manufacturer: Telstra
```

```
pm extension (aggregation center)> edit ib os.list add
OS: Others
```

3. Add or modify the new entry in the device_man_dg_os.map file. If it already exists use the “modify” action; else use the “add” action to create an entry:

```
pm extension (aggregation center)> show ib device_man_dg_
os.map
1       [Samsung Galaxy Nexus][Samsung][SmartPhone][Android]
2       [Apple iPhone 6]      [Apple]  [SmartPhone][iOS]
```

```
pm extension (aggregation center)> edit ib device_man_dg_
os.map modify record 1
```

Or, run the following command:

```
pm extension (aggregation center)> edit ib device_man_dg_
os.map add
```

Note: After editing the file, generate and push all the IBs.

Overriding Devices by TAC Identifier

1. To override a mobile device based on its TAC identifier, add its entry in override IB:

```
> en
# conf t
(config) # pmx
Welcome to pmx configuration environment.
pm extension> subshell aggregation_center
pm extension (aggregation center)> override add device from
tac
TAC : 99999999Device : Telstra Box
Manufacturer : Telstra
Device Group : Mobile Router
OS : Others
updating device_man_dg_os.map
Added override property
pm extension (aggregation center)> override show device from
tac
TAC  Device  Manufacturer  Device Group  OS
---------------------------------------------------------
99999999  Telstra Box     Telstra            SmartPhone
Others
```

2. Add manufacturer, OS and device in their corresponding list:

```
pm extension (aggregation center)> edit ib model.list_static
add
Device: Telstra Box
```

```
pm extension (aggregation center)> edit ib manufacturer.list
add
Manufacturer: Telstra
```

```
pm extension (aggregation center)> edit ib os.list add
OS: Others
```

3. Add or modify the new entry in the device_man_dg_os.map file. If it already exists use the “modify” action; else use the “add” action to create an entry:

```
pm extension (aggregation center)> show ib device_man_dg_
os.map
1       [Samsung Galaxy Nexus][Samsung][SmartPhone][Android]
2       [Apple iPhone 6]      [Apple]  [SmartPhone][iOS]
pm extension (aggregation center)> edit ib device_man_dg_
os.map modify record 1
```

Or, run the following command:

```
pm extension (aggregation center)> edit ib device_man_dg_
os.map add
```

Note: After editing the IB, generate and push all the IBs.

Overriding Service Provider Based on URL

To override default category of service provider, edit the the sp_url_cat.map information base.

Note: You can only update categories for service providers that are identified as generating heavy traffic.

1. Run the following commands:

```
> en
# conf t
(config) # pmx
Welcome to pmx configuration environment.
pm extension> subshell aggregation_center
pm extension (aggregation center)> override add sp from url
URL : facebook
Service Provider : facebook.com
Added override property
```

2. To verify the update was made, run the show ib command and look for the entry for the service provider:

```
pm extension (aggregation center)> override show sp  from url
URL       Service Provider
---------------------------------------------------------
facebook  facebook.com
```

Note: After editing the IB, generate and push all the IBs.

Overriding Mobile Applications Based on User Agents

To override a mobile device based on the user agent, user must know either the entire UA or any substring that is part of UA.

For example, if all UA containing “AFL” should be annotated as a particular application, then provide “AFL” in the UA field.

1. Run the following commands:

```
pm extension (aggregation center)> override add app from ua
User Agent : AFL
Mobile Application Name : AFLApp
Mobile Application Type : Browser
Mobile Application Category : Games
Added override property
```

```
pm extension (aggregation center)> override show app from ua
User Agent          Mobile Application Name  Mobile
Application Type  Mobile Application Category
-----------------------------------------------------------
AFL                 AFLApp                    Browser
       Games
```

2. Add application and category entries in the corresponding lists:

```
pm extension (aggregation center)> edit ib mobileappname.list
addMobile Application Name: AFLApp
```

```
pm extension (aggregation center)> edit ib
mobileappcategory.list add
Mobile Application Category: Games
```

Note: After editing the IB, generate and push all the IBs.

Adding Top Subscribers

To see a subscriber as a part of Top Subscriber list, add its entry in the subscribersib.list IB. Ensure that this subscriber ID is in sync with the subscriber ID used by the MURAL system.

1. Run the following command:

```
pm extension (aggregation center)> edit ib subscribersib.list
add
Subscriber: 1000000009
```

Note: After editing the IB, generate and push all the IBs.

Modifying Behaviour for Cell Sector Annotation

You can categorize the Cell Sectors, which are not present in the IB configuration into Other cells,and and GGSN and SGSN which are not present in the IB configuration into Miscellaneous.

The values of Cell Sectors, GGSN and SGSN to be retained in data are specified in the following IBs:

- **cellLocationInfo.list** for Cell Sectors
- **ipGGSN.map** for GGSN
- **ipSGSN.map** for SGSN

Perform the following steps for categorizing the Cell Sectors, GGSN and SGSN:

1. Execute the following command to create the backup of *EDR.json*:

```
# cp -p /opt/etc/oozie/EDR/app/EDR.json
/opt/etc/oozie/EDR/app/EDR.json.org
```

2. Execute the following command to check the current values assigned to **groupSGSNAsMisc**, **groupGGSNAsMisc** and **emitUsingMCCMNC**:

```
# grep -e "groupSGSNAsMisc" -e "groupGGSNAsMisc" -e
"emitUsingMCCMNC" /opt/etc/oozie/EDR/app/EDR.json
        "groupGGSNAsMisc":"false"
        "groupSGSNAsMisc":"false"
        "emitUsingMCCMNC":"false"
```

3. Execute the following command to set the value of groupGGSNAsMisc as true:

```
# sed -i
's/"groupGGSNAsMisc":"false"/"groupGGSNAsMisc":"true"/'
/opt/etc/oozie/EDR/app/EDR.json
```

4. Execute the following command to set the value of groupSGSNAsMisc as true:

```
# sed -i
's/"groupSGSNAsMisc":"false"/"groupSGSNAsMisc":"true"/'
/opt/etc/oozie/EDR/app/EDR.json
```

5. Execute the following command to set the value of emitUsingMCCMNC as true:

```
# sed -i
's/"emitUsingMCCMNC":"false"/"emitUsingMCCMNC":"true"/'
/opt/etc/oozie/EDR/app/EDR.json
```

6. Execute the following command to validate the value of **groupSGSNAsMisc**, **groupGGSNAsMisc** and **emitUsingMCCMNC**:

```
# grep -e "groupSGSNAsMisc" -e "groupGGSNAsMisc" -e
"emitUsingMCCMNC" /opt/etc/oozie/EDR/app/EDR.json
        "groupGGSNAsMisc":"true"
        "groupSGSNAsMisc":"true"
        "emitUsingMCCMNC":"true"
```

Adding the Cell Sectors, GGSN and SGSN in IBs

If you want to retain the values of Cell Sectors, GGSN and SGSN, add their values in cellLocationInfo.list, ipGGSN.map and ipSGSN.map respectively.

To add the values, perform the perform the following steps:

1. To add the values of Cell Sectors, GGSN and SGSN to be retained incellLocationInfo.list, ipGGSN.map and ipSGSN.map respectively, execute the following commands:

```
> en # conf t (config) # pmx subshell aggregation_center
pm extension (aggregation center)> edit ib
cellLocationInfo.list add MCC,MNC: 101,01
pm extension (aggregation center)> edit ib ipGgsn.map add
GGSN IP: 27.23.157.1
GGSN: GGSN1
pm extension (aggregation center)> edit ib ipSgsn.map add
SGSN IP: 2.2.2.1
SGSN: SGSN1
```

   NOTE: If the IB is empty, all the cell sectors will be emitted as "Other Cells".

2. To update the IBs, execute the following commands:

```
> en # conf t (config) # pmx subshell aggregation_center
pm extension (aggregation center)> generate all ibs
pm extension (aggregation center)> push all ibs
```

Changing Data Retention Period

This topic describes the steps you should take to change the data retention period on various nodes. The procedures vary depending on the kind of data: Collector raw data, Collector output data and Cached Rubix data.

Changing Data Retention for Collector Raw Input Data

Data retention for collector input data is governed by the backup-file-expiryperiod configuration under collector config. This is configured for all adaptors and gateways.

1. To check the current value , log in to the master collector node and run:

```
> en
# conf t
(config) # _shell
# cli -t "en" "conf t" "show running-config full" |grep
backup-file-expiry-period
collector modify-instance 1 modify-adaptor bulkStats
modifyfile-
if bulkStatsFile1 backup-file-expiry-period 168
collector modify-instance 1 modify-adaptor bulkStats
modifyfile-
if bulkStatsFile2 backup-file-expiry-period 168
collector modify-instance 1 modify-adaptor edrflow modifyfile-
if flowFile backup-file-expiry-period 24
collector modify-instance 1 modify-adaptor edrhttp modifyfile-
if httpFile backup-file-expiry-period 24
```

   Here, the retention period for the configured adaptors and gateways is 24 hours for EDR and 168 hours for BulkStats.

2. To change the data retention period for both Bulkstats and EDR to 48 hours, run the following commands:

```
# cli -m config
(config) # collector modify-instance 1 modify-adaptor
```

```
bulkStats modify-file-if bulkStatsFile1 backup-file-expiry-

period 48
(config) # collector modify-instance 1 modify-adaptor
bulkStats modify-file-if bulkStatsFile2 backup-file-
expiryperiod
48
(config) # collector modify-instance 1 modify-adaptor edrflow
modify-file-if flowFile backup-file-expiry-period 48
(config) # collector modify-instance 1 modify-adaptor edrhttp
modify-file-if httpFile backup-file-expiry-period 48
(config) # pm process collector restart
```

Note: Change the data retention period on the standby collector and name node as well.

Changing Data Retention for Collector Output Data

Data retention for collector output data is governed by the cleanupOffset in the CleanupCollector jobs of EDR and Bulkstats data.

1. Execute the following commands to enter the oozie subshell:

```
> en
# conf t
(config)# pmx subshell subshell-name
```

2. For EDR

   Execute the following command to check the current data retention period:

```
pm extension (oozie)> show config job CleanupCollector full
```

   The resulting output may resemble to the following:

```
Attributes:
-
frequencyUnit : hour
```

```
jobEnd : 2099-12-31T00:00Z
jobFrequency : 1
jobLocation :
/opt/ooziecore/genericjobs/CleanupDataset
jobStart : 2014-08-24T08:00Z
jobType : CleanupDataset
Actions:
-
CleanupDatasetAction:
cleanupDatasets : ['atlas_edrhttp_1', 'atlas_
edrflow_1']
atlas_edrhttp_1
-
doneFile : .*_DONE
endOffset : 1
frequency : 5
outputOffset : 0
path :
/data/collector/1/output/edrhttp/%Y/%M/%D/%H/%mi
pathType : hdfs
startOffset : 12
startTime : 2014-08-24T07:00Z
atlas_edrflow_1
-
doneFile : .*_DONE
endOffset : 1
frequency : 5
outputOffset : 0
path :
/data/collector/1/output/edrflow/%Y/%M/%D/%H/%mi
pathType : hdfs
startOffset : 12
startTime : 2014-08-24T07:00Z
```

```
cleanupOffset : 24
leaveDonefile : false {Default: true}
```

For Bulkstats

Execute the following command to check the current retention period:

```
pm extension (oozie)> show config job CleanupRawRecords full
```

The resulting output may resemble to the following:

```
Attributes:
-
frequencyUnit : day
jobEnd : 2099-12-31T00:00Z
jobFrequency : 1
jobLocation :
/opt/ooziecore/genericjobs/CleanupDataset
jobStart : 2014-09-07T00:00Z
jobType : CleanupDataset
Actions:
-
CleanupDatasetAction:
cleanupDatasets : input_bulkstats
input_bulkstats
-
doneFile : .*_DONE {Default: _DONE}
endOffset : 1 {Default: 1}
frequency : 5 {Default: 5}
outputOffset : 0 {Default: 0}
path :
/data/collector/1/output/bulkStats/%Y/%M/%D/%H/%mi {Default:
/data/collector/bs/%Y/%M/%D/%H/%mi/*}
pathType : hdfs {Default: hdfs}
startOffset : 1 {Default: 1}
startTime : 2013-01-01T00:00Z {Default:
```

```
2013-01-01T00:00Z}
cleanupOffset : 7
leaveDonefile : false {Default: true}
```

3. To change the data retention period, run the following command:

   ```
   pm extension (oozie)> show config job CleanupCollector
   ```

   The output may resemble as follows:

   ```
    Attributes:
       -
           frequencyUnit : hour
           jobEnd        : 2099-12-31T00:00Z
           jobFrequency  : 1
           jobLocation   :
   /opt/ooziecore/genericjobs/CleanupDataset
           jobStart      : 2015-04-29T05:00Z
           jobType       : CleanupDataset

       Actions:
       -
       CleanupDatasetAction:
           cleanupDatasets : ['input_bulkstats_2', 'input_
   bulkstats', 'atlas_edrflow_2', 'atlas_edrflow_1', 'atlas_
   edrhttp_2', 'atlas_edrhttp_1']
   ```

   Perform the following steps:

   a. For Bulkstats, define the period in hours for all bulkstats datasets that exist in CleanupCollector:

      ```
      pm extension (oozie)> set dataset input_bulkstats_2
      attribute cleanupOffset 240
      ```

      Repeat the command for `input_bulkstats`.

   b. For EDR, define the period in hours for all EDR (http / flow) datasets that exist in CleanupCollector:

```
pm extension (oozie)> set dataset atlas_edrflow_1
attribute cleanupOffset 24
```

Repeat the command for the following:

- atlas_edrflow_1
- atlas_edrhttp_2
- atlas_edrhttp_1

Note: Change the data retention period on the standby and name nodes as well.

4. Restart the Cleanup jobs on the master Collector node:

```
pm extension (oozie)> stop jobname CleanupCollector
pm extension (oozie)> run job CleanupCollector
```

Changing Data Retention for data stored in Insta

Data older than configured number of hours on Insta gets automatically cleaned at regular intervals by jobs which run every day.

1. Check the jobs on the Insta node:

   For EDR

```
(config) # show jobs 0
Job 0:
Status: pending
Enabled: yes
Continue on failure: no
Schedule type: daily
Time of day: 23:00:00
Absolute start: (no limit)
Absolute end: (no limit)
Last exec time: Wed 2014/09/17 23:00:01 +0000
Next exec time: Thu 2014/09/18 23:00:00 +0000
Commands:
```

```
Command 0: insta delete-with-retention instance-id 0
bin-class 60min bin-type * aggr * retention-period 4320
Last output:
All args good. Data cleanup initiated. See logs for more
details
```

For Bulkstats

```
(config) # show jobs 1
Job 1:
Status: pending
Enabled: yes
Continue on failure: no
Schedule type: daily
Time of day: 23:00:00
Absolute start: (no limit)
Absolute end: (no limit)
Last exec time: Wed 2014/09/17 23:00:00 +0000
Next exec time: Thu 2014/09/18 23:00:00 +0000
Commands:
Command 1: insta delete-with-retention instance-id 1
bin-class 5min bin-type * aggr * retention-period 8760
Last output:
All args good. Data cleanup initiated. See logs for more
details
```

2. Change the data retention period to 4320:

   For EDR

```
(config) # job 0 command 0 "insta delete-with-retention
instance-id 0 bin-class 60min bin-type * aggr *
retentionperiod
4320"
(config) # write memory
```

For Bulkstats

```
(config) # job 1 command 1 "insta delete-with-retention
instance-id 1 bin-class 5min bin-type * aggr * retentionperiod
4320"
(config) # write memory
```

Note: Change the data retention period on the standby Insta node as well.

Changing Data Retention for Cached Data in Rubix

1. Stop the EDR and Bulkstats UI processes on both the master and standby nodes:

   ```
   > en
   # conf t
   (config) # rubix modify-app process-name modify-instance 1
   disable
   (config) # rubix modify-app process-name disable
   ```

   Here, `process-name` is:

   - `bulkstats`
   - `edr`

2. On the Rubix nodes data is prefetched from Insta based on the settings for `VariableRetentionMap`, `timeseriesLevelMap` and `SCHEDULERVARIABLEGRANULARITYMAP` parameters in the Rubix applic- ations.

   To view the data retention period of these parameters:

   ```
   # cli -t "en" "conf t" "sh ru f" | egrep "
   VariableRetentionMap|timeseriesLevelMap|SCHEDULERVARIABLEGRANU
   LARITYMAP"|egrep "atlas|bulkstats"
   rubix modify-app atlas set adv-attribute
   SCHEDULERVARIABLEGRANULARITYMAP value 1h:169;1d:60;1M:3
   rubix modify-app atlas set adv-attribute VariableRetentionMap
   value 1h:169;1d:60;1M:3
   ```

```
rubix modify-app atlas set adv-attribute timeseriesLevelMap
value 1h:169;1d:60;1M:3
rubix modify-app bulkstats set adv-attribute
SCHEDULERVARIABLEGRANULARITYMAP value 15m:98;1h:169;1d:60;1M:3
rubix modify-app bulkstats set adv-attribute
VariableRetentionMap value 15m:98;1h:169;1d:60;1M:3
rubix modify-app bulkstats set adv-attribute
timeseriesLevelMap value 15m:98;1h:169;1d:60;1M:3
```

In this output, 15m, 1h, 1d, and 1M correspond to various granularities and the value for each granularity signify the number of points cached for it.

For example,

For `atlas app VariableRetentionMap value 1h:169;1d:60;1M:3`, number of hourly cached points is 169 (last 7days), daily cached points is 60 (last 2 months) and monthly cached pointed is 3 (last 3 months).

For Bulkstats, 15min points are cached for last 24 hours, hourly points for last 7 days, daily for last 60 days and monthly for last 3 months.

3. The cached data is stored in disk cache for atlas and bulkstats based on the `diskRetentionPeriod` parameter. To set a new value for this parameter:

```
# cli -t "en" "conf t" "show running-config full" |grep
"diskRetentionPeriod"|egrep "atlas|bulkstats"
rubix modify-app atlas set adv-attribute
diskRetentionPeriod value 1h:169;1d:60;1M:3
rubix modify-app bulkstats set adv-attribute
diskRetentionPeriod value 15m:98;1h:169;1d:60;1M:3
```

4. If disk cache is enabled, clean disk cache for atlas and bulkstats:

```
> en
# _shell
# rm -rf /data/diskstore/$(cli -t "en" "conf t" "sh runn" |
grep atlas | grep clusterName | cut -d' ' -f11)/*
```

```
# touch /data/diskstore/$(cli -t "en" "conf t" "sh runn" |
grep atlas | grep clusterName | cut -d' ' -f11)/disk_mounted
# rm -rf /data/diskstore/$(cli -t "en" "conf t" "sh runn" |
grep bulkstats | grep clusterName | cut -d' ' -f11)/*
# touch /data/diskstore/$(cli -t "en" "conf t" "sh runn" |
grep bulkstats | grep clusterName | cut -d' ' -f11)/disk_
mounted
```

5. To change data retention period on rubix nodes, the properties `Vari-ableRetentionMap`, `timeseriesLevelMap`, `SCHEDULERVARIABLEGRANULARITYMAP` and `diskRetentionPeriod` need to be modified for atlas and bulkstats:

```
# cli -m config
(config) # rubix modify-app atlas set adv-attribute
VariableRetentionMap value 1h:240;1d:75;1M:4
(config) # rubix modify-app atlas set adv-attribute
SCHEDULERVARIABLEGRANULARITYMAP value 1h:240;1d:75;1M:4
(config) # rubix modify-app atlas set adv-
attributetimeseriesLevelMap value 1h:240;1d:75;1M:4
(config) # rubix modify-app atlas set adv-attribute
diskRetentionPeriod value 1h:240;1d:75;1M:4
(config) # rubix modify-app bulkstats set adv-attribute
VariableRetentionMap value 15m:196;1h:240;1d:75;1M:4
(config) # rubix modify-app bulkstats set adv-attribute
SCHEDULERVARIABLEGRANULARITYMAP value
15m:196;1h:240;1d:75;1M:4
(config) # rubix modify-app bulkstats set adv-attribute
timeseriesLevelMap value 15m:196;1h:240;1d:75;1M:4
(config) # rubix modify-app bulkstats set adv-attribute
diskRetentionPeriod value 15m:196;1h:240;1d:75;1M:4
(config)# write memory
```

Here, retention period for atlas is changed to 10 days hourly points, 75 days of daily points and 4 monthly points, for bulkstats – 2 days of 15min points, 10 days hourly points, 75 days of daily points and 4 monthly points.

6. Repeat steps 4 and 5 on standby rubix nodes as well.
7. Restart EDR and Bulk Stats UI processes on both Master and Standby Nodes for the changes to take effect:

```
> en
# conf t
(config) # rubix modify-app process-name enable
(config) # rubix modify-app process-name modify-instance 1
enable
```

Here, `process-name` is:

- bulkstats
- atlas

Modifying the Database

Upgrading the Bulkstats Schema Version

A new software release may include a new Bulkstats schema version. This section describes how to upgrade the Bulkstats schema.

Before You Begin

- Refer to the *MURAL Release Notes* for details on upgrading your software.
- Be sure the existing system is up and running.

Performing the Upgrade

To upgrade the bulkstats schema version, you will need to access the `subschema.txt` file, update the ASR version number in the IBs in the Collector, update the schema version number against each gateway that will use the updated schema.

1. Get the `subschema.txt` file to be applied on the ASR gateway for the new version. The file is available on the master Collector (NameNode) server under the following path after the software upgrade has been completed:

   ```
   ls -l /opt/catalogue/bulk_stats/subschema.txt
   -rw-r--r-- 1 admin root 593009 Dec  7 14:54
   /opt/catalogue/bulk_stats/subschema.txt
   ```

   You can copy the file from this path on the Collector node and feed it to all the ASRs that will use the updated schema version.

2. Update the schema version number for each ASR gateway. You will need the new schema version number that is to be used with upgraded release, which you can find in the release notes. The sample output in this section uses schema version 16.

Note: You must do this step after the Insta nodes have come up following the upgrade and before starting data processing (before starting the oozie jobs on the upgraded system).

3. Log in to the master Collector node and edit the IBs to update the schema version number against each gateway that will use the upgraded schema:

   a. To update bulkstats schema version, check the `gateway.conf` file in `/data/configs/gateway` on the master Collector for already added gateway with different schema version.

   b. Execute the below command with updated schema version number with all the other values the same as in the example:

   ```
   pm extension (aggregation_center)> add gateway name
   <gateway-name> region <gateway-region> location <gateway-
   area> schema_version <bulkstat-schema-version> ip
   <gateway-IP> timezone <gateway-timezone> edr-filename-
   pattern <incoming-EDR-fileName-pattern> bulkstat-
   filename-pattern <incoming-BS-fileName-pattern> edr-
   collector-filename-pattern <collector-config-edr-
   filename-format> bulkstat-collector-filename-pattern
   <collector-config-bs-filename-format> collector-ip
   <collector-IPs> type <gateway-type> edr-file-path
   <incoming-EDR-files-path-on-collector> bulkstat-file-path
   <incoming-BS-files-path-on-collector>
   ```

   Note: See "Modifying Gateway Attributes" on page 123 for more information on gateway attributes and a sample of the output generated by this command.

   For example:

   ```
   pm extension (aggregation_center)> pm extension
   (aggregation center)>add gateway name GMPLAB3 region SEZ
   location INDIA schema_version 18 ip 10.10.10.255 timezone
   ```

```
UTC edr-filename-pattern *_*_*_*_%MM%DD%YYYY%hh%mm%ss_
*.gz bulkstat-filename-pattern *_%YYYY%MM%DD%hh%mm%ss
edr-collector-filename-pattern %DC_*_*_
*_%MM%DD%YYYY%hh%mm%ss_*.gz bulkstat-collector-filename-
pattern *_%YYYY%MM%DD%hh%mm%ss collector-ip
10.10.10.101,10.10.10.102 type HA edr-file-path
/GMPLAB1/edr/ bulkstat-file-path /GMPLAB1/bs/
Gateway/DC already exists with following configuration


*************************Gatewa
y********************************

Name: GMPLAB3

Dc: GMPLAB3

Associated Region: SEZ

Location: INDIA

Schema Version :  15

IP: 10.10.10.255

Timezone: UTC

Flow-EDR/Http-EDR/Asn-EDR Filename Pattern:
*_%YYYY%MM%DD%hh%mm%ss

Bulkstat Filename Pattern: *_MURAL-edr_
*_%MM%DD%YYYY%hh%mm%ss*.*
```

```
Type: HA

Flow-EDR/Http-EDR Filename Path:
/data/collector/GMPLAB3/bs

Bulkstat Filename Path: /data/collector/GMPLAB3/edr



*********************************************************
*******



Do you want to add/update the gateway/DC?(Yes/No): Yes



*********************************************************
*******

*********************Successfully
Updated*************************
```

4. Log into the master Collector node and go to the pmx `bulkstats` subshell:

```
> en
# conf t
(config)# pmx subshell subshell-name
```

5. Push all IBs:

```
pm extenstion (bulkstats)# push all ibs
```

6. Generate all IBs:

```
pm extenstion (bulkstats)# generate all ibs
```

The resulting output may resemble to the following:

```
[key.map]:
    generated key.id.map
[gateway.map]:
    generated gateway_bs.id
    generated version_bs.id
    generated dc_bs.id
    generated region_bs.id
    generated area_bs.id
    generated gatewayRegionArea.id.map
[schema.map]:
    generated schema.id.map
[metric.map]:
    generated metric.id.map
[gatewayIDVersion.map]:

Summary:
========

Successful IBs : 5 out of 5
Failed IBs : No id generation failures.
pm extension (bulk stats)> push all ibs
Checking if there are pending transfers...
Sending 32 ibs to 1 receiver (using 1 thread, skipping 0)
*
Summary:
========
Failed Transfers     : 0 out of 1
Successful Transfers : 1 out of 1

pm extension (bulk stats)>
```

7. To view new schema version that has been upgraded:

```
pm extension (bulkstats)> show gateways
```

8. Check whether all Collector IPs are added for pushing this information:

```
pm extension (aggregation_center)> show collector IPs
```

9. If all Collector IPs are not added, then add Collector IPs:

```
pm extension (aggregation_center)> set collector IPs comma-
separated-ip-list
```

   Where *comma-separated-ip-list* is two or more IP addresses, separated by a comma.

   For example:

```
pm extension (aggregation_center)> set collector IPs
192.168.1.1,192.168.2.2
```

10. Push gateway configuration to all collectors:

```
pm extension (aggregation_center)> push gateway configuration
```

Backup Strategy

If any of these steps fail, do not proceed. Instead, contact Technical Support to resolve the issue.

Backing Up Data and IBs

Backing-up the system configuration enables you to re-populate system information on an alternate node if a node failure occurs. This section describes the procedure of backing up and restoring the configuration of databases and information bases, both of which are essential to the functionality of the system.

Information Bases

Procedures for how to perform these backups can be found in "Backing Up and Restoring Information Bases" on page 181 of this guide.

IBs are stored in two folder locations:

- `/data/ib/inbox`
- `/data/ib/work`

Whenever IBs are edited for one of the following four scenarios, take a backup of the IBs:

1. Patch Application
2. Release Upgrade
3. IB/Data Overriding
4. URL Categorization update

Databases

Procedures for how to perform backups of the PGSQL Database can be found in "Backing-Up the PGSQL Database" on page 191 of this guide.

Procedures for how to perform backups of the Insta Database can be found in "Backing Up and Restoring a Large (TB) Insta Database" on page 167 of this guide.

Offline Reports

Backup all the content in the folder `/data/rge/` on the master UI node.

Note: Backup of PGSQL DB data and reports data should be taken at the same time.

To restore, copy all the backed-up files and folders to the `/data/rge/` folder on the restored master UI RGE node.

Backup GMS Configuration Files

Backing-up the GMS configuration files enables you to re-populate system information on an alternate node if a node failure occurs.

1. Log into the Collector node GMS server and backup the XML file and config files:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# _shell
admin@host# mkdir -p /data/GMS_backup/
admin@host# cp -R /config/gms/* /data/GMS_backup/
```

Note: The above folder `/data/GMS_backup` can be backed up on external storage, such as a tape drive.

Backing Up and Restoring a Large (TB) Insta Database

This section describes how to back up and restore the database on the Insta node. It also includes best practices.

Restoring a large database would be done in one of two secenarios:

1. LUNs have gone down
2. Insta node failure

Before You Begin

1. Stop all types of cube exporter jobs on the master Collector/NameNode of the setup on which you are taking a backup of the Insta node.

   a. Log into the master Collector node and go to the oozie subshell:

   ```
   > en
   # conf t
   (config)# pmx subshell subshell-name
   ```

   b. Stop all jobs:

   ```
   pm extension (oozie)> stop jobname all
   ```

   c. Verify jobs have stopped:

   ```
   pm extension (oozie)> show coordinator RUNNING jobs
   ```

   If it returns this output then all oozie jobs have been successfully stopped:

   ```
   no jobs matching your criteria!
   ```

   Wait for this message to appear before proceeding.

2. On the UI nodes, stop all UI tomcat instances. Log into each UI node one by one and repeat this step:

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   ```

```
host [cluster : master|standby](config)# pm process rubix
terminate
```

3. Stop the Insta process from the master Insta node.

```
host [cluster : master]> en
host [cluster : master]# _shell
admin@host# cc shutdownsystem y
```

4. On the master Insta node, verify that the node is designated as 1:

```
[admin@host ~]# cli -t "en" "conf t" "show run full" | grep
"module"
```

The resulting output may resemble:

```
   insta adapters infinidb module 1
   insta adapters infinidb module 1 ip 192.168.147.15
   insta adapters infinidb module 2
   insta adapters infinidb module 2 ip 192.168.147.16
   insta adapters infinidb modulecount 2
```

5. If Insta master and module 1 are the same, then follow the steps mentioned below:

   a. Verify that there is enough disk space on the destination node to which you will copy the data backup. If the local disk on that server does not have enough space, use a temporarily mounted LUN to hold the data until it is restored to destination database.

      To verify the amount of space that the data to be backed up requires:

      - On the master node:

        ```
        > en
        # _shell
        [admin@host ~]# cd /data/Calpont
        [admin@host Calpont]# du -sh .
        756.2G
        ```

- On the standby node:

```
> en
# _shell
[admin@host ~]# cd /usr/local/Calpont
[admin@host Calpont]# du -s * | grep -v "data." |
awk '{u+=$1}END{ print u}'||exit $?;
```

The resulting output may resemble:

```
224868
**  Total Data Size = Size on instaBaseMaster +
instaBaseStandby
```

b. Create the directories on the machine where the database backup is being saved to store the database:

```
> en
# _shell
# mkdir -p /data/tempMount/destDirTimeStamp/bkupActive/
# mkdir -p
/data/tempMount/destDirTimeStamp/bkupActive/data1
# mkdir -p
/data/tempMount/destDirTimeStamp/bkupActive/data2
# mkdir -p /data/tempMount/destDirTimeStamp/bkupStandBy/
```

Note: Create as many `dataX` directories as there are `dbroots` (LUNs) being used for Insta.

To obtain the count of dbroots, run the following command on the master Insta node:

```
# cli -t "en" "conf t" "show run full" | grep
"dbrootcount" | awk '{print $NF}'
9
```

Copying the Database

1. Copy `/data/Calpont/` from an active location to a storage destination.

   On the master Insta node, for the database is being backed up:

   ```
   > en
   # _shell
   [admin@host ~]# rsync -avz --exclude=data* /data/Calpont/
   root@dest_IP:/data/tempMount/destDirTimeStamp/bkupActive/
   [admin@host ~]# time scp -c arcfour -r /data/Calpont/data1/*
   root@dest_IP:/data/tempMount/destDirTimeStamp/bkupActive/
   data1/
   [admin@host ~]# time scp -c arcfour -r /data/Calpont/data2/*
   root@dest_IP:/data/tempMount/destDirTimeStamp/bkupActive/
   data2/
   ```

   Note: Create as many `dataX` directories as there are `dbroots` (LUNs) being used for Insta.

2. Log into the standby Insta node and copy the data in `/usr/local/Calpont/` from the standby node to the storage destination:

   ```
   [admin@host ~]# rsync -avz --exclude=data* /usr/local/Calpont/
   root@dest_IP:/data/tempMount/destDirTimeStamp/bkupStandBy/
   ```

3. Start the `infinidb` on the master Insta node whose database is being backed up:

   ```
   [admin@host ~]# cc startsystem y
   [admin@host ~]# cli -m config
   host [cluster : master](config)# insta infinidb get-systeminfo
   ```

4. Go to the pmx oozie subshell:

   ```
   > en
   # conf t
   (config)# pmx subshell subshell-name
   ```

5. Start the jobs and the tomcat instances for this setup.

```
pm extension (oozie)> run job all
```

6. Restart Rubix on all UI nodes. Log in to each UI node one by one and repeat this step:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby]# pm process rubix restart
```

If Insta master and module 1 are different, then follow the steps mentioned below:

a. Verify that there is enough disk space on the destination node to which you will copy the data backup. If the local disk on that server does not have enough space, use a temporarily mounted LUN to hold the data until it is restored to destination database.

Verifying Size of Data to be Backed up

- On the master node:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# _shell
[admin@host ~]# cd /usr/local/Calpont
[admin@host Calpont]# du -sh .
```

The resulting output may resemble:

```
756.2G
```

- On the standby node:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# _shell
[admin@host ~]# cd /data/Calpont
[admin@host Calpont]# du -s * | grep -v "data." |
awk '{u+=$1}END{ print u}'||exit $?;
```

The resulting output may resemble:

```
224868
```

** Total Data Size = Size on instaBaseMaster + instaBaseStandby

b. Create the directories on the machine where the database backup is being saved to store the database:

```
> en
# _shell
# mkdir -p /data/tempMount/destDirTimeStamp/bkupActive/
# mkdir -p /data/tempMount/destDirTimeStamp/bkupActive/
data1
# mkdir -p /data/tempMount/destDirTimeStamp/bkupActive/
data2
# mkdir -p /data/tempMount/destDirTimeStamp/bkupStandBy/
```

Note: Create as many `dataX` directories as there are `dbroots` (LUNs) being used for Insta.

Copying the Database

1. Copy `/usr/local/Calpont/` from an active location to storage destination.

   On the master Insta node, whose database is being backed up:

```
> en
# _shell
[admin@host ~]# rsync -avz --exclude=data* /usr/local/Calpont/
root@dest_IP:/data/tempMount/destDirTimeStamp/bkupActive/
[admin@host ~]# time scp -c arcfour -r
/usr/local/Calpont/data1/* root@dest_IP:/data/tempMount/
destDirTimeStamp/bkupActive/data1/
[admin@host ~]# time scp -c arcfour -r /usr/local/Calpont/
data2/* root@dest_IP:/data/tempMount/destDirTimeStamp/
bkupActive/data2/
```

Note: Create as many `dataX` directories as there are `dbroots` (LUNs) being used for Insta.

2. Log into the standby Insta node and copy the data in `/data/Calpont/` from the standby node to the storage destination:

```
[admin@host ~]# rsync -avz --exclude=data* /data/Calpont/
root@dest_IP:/data/tempMount/destDirTimeStamp/bkupStandBy/
```

3. Start the infinidb on this Insta node:

```
[admin@host ~]# cc startsystem y
[admin@host ~]# cc getsysteminfo
host(config)# insta infinidb get-systeminfo
```

4. Log into the pmx oozie subshell:

```
> en
# conf t
(config)# pmx subshell subshell-name
```

5. Start the jobs and tomcat instances for this setup:

```
pm extension (oozie)> run job all
```

6. Restart Rubix on all UI node. Log in to each UI node one by one and repeat this step:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby]# pm process rubix restart
```

Restoring a Large (TB) Database

Execute the following section before restoring backed up Insta Data.

Stopping Dependent Processes

1. Stop all the Oozie jobs on the master Collector node:

   ```
   admin@instanode# cli -m config
   admin@instanode(config)# pmx
   pm extension> subshell oozie
   pm extension (oozie)> stop jobname all
   ```

2. Verify that all Oozie jobs are stopped:

   ```
   pm extension (oozie)> show coordinator RUNNING jobs
   ```

   Verify that no RUNNING jobs are displayed. If still there are jobs running, wait a short time and try again to verify that no jobs are running before proceeding to the next step.

3. On the UI nodes, stop all UI tomcat instances. Log in to each UI node one by one and repeat this step.

   ```
   > en
   # conf t
   (config) # pm process rubix terminate
   ```

Installing the New XML File

1. Establish an SSH connection to the GMS server using management IP and start the installation on the Insta cluster:

   ```
   host [cluster : master]> en
   host [cluster : master]# conf t host host [cluster : master]
   (config)# gms config Insta_Storage_Failure.xml activate
   host [cluster : master](config)# install appliance cluster
   cluster-name cluster-name force-format
   ```

   Where the *cluster-name* is the name of the Insta cluster and it will automatically come up when you press tab after typing cluster-name.

   For example,

```
host [cluster : master](config)# install appliance cluster
cluster-name INSTA-CLUS-GMS force-format
```

2. Monitor the status of the installation status on the Collector blades by executing this command to show the percentage completed:

```
host [cluster : master](config)# install appliance show
installation-status cluster INSTA-CLUS-GMS
```

3. Wait for the `Node successfully installed...` message to appear before proceeding.

Installing and Configuring Backed Up Data

Install and configure the Insta nodes in high availability mode using the same configurations as the setup from where the backup was taken. Criteria includes the following:

- Same number of instances
- Same number and size of dbroots
- Same database name
- Same `cubeXML`

To restore a large Insta database:

1. Uninstall `infinidb` from the master node:

```
host [cluster : master]> en
host [cluster : master]# conf t
host [cluster : master](config)# insta infinidb uninstall
host [cluster : master](config)# insta infinidb get-status-
info
```

The output may resemble as follows:

```
“Infinidb Install status : UNINIT”
```

2. Format `dbroots` and schema from the new active:

```
host [cluster : master](config)# insta infinidb disk-mgmt
format dbroot 1
host [cluster : master](config)# insta infinidb disk-mgmt
format dbroot 2
```

Note: Repeat this step for all the dbroots that are configured in your setup.

```
host [cluster : master](config)# insta infinidb disk-mgmt
clean schema
(config)# write mem
```

Note: If the formatting of dbroots and schema is not successful, contact the Technical Support.

3. Mount `dbroot` folder for backup on new active:

```
host [cluster : master]# _shell
[admin@host ~]# mount /dev/mapper/dbroot1p1
/data/Calpont/data1/
[admin@host ~]# mount
/dev/mapper/dbroot2p1/data/Calpont/data2/
```

Note: Repeat this step for all the dbroots that are configured in your setup.

To verify if the LUNs got mounted successfully, run the `df -kh` command. The output displays the details of the mounted LUNs.

4. Log into the server where the backup is saved and copy the data from `bkupActive/` to the new master Insta node:

```
[admin@host ~]# rsync -avz --exclude=data*
/data/tempMount/destDirTimeStamp/bkupActive/
root@newActive-IP:/data/Calpont/
[admin@host ~]# time scp -c arcfour -r /data/tempMount/
destDirTimeStamp/bkupActive/data1/*
root@newActive-IP:/data/Calpont/data1/
[admin@host ~]# time scp -c arcfour -r /data/tempMount/
```

```
destDirTimeStamp/bkupActive/data2/*
root@newActive-IP:/data/Calpont/data2/
```

Note: Repeat this step for all the dbroots that are configured in your setup.

5. On the new master Insta node, umount the `/data/Calpont/dataX` folders before infinidb installation from new Insta active:

```
[admin@host ~]# umount -l /data/Calpont/data1
[admin@host ~]# umount -l /data/Calpont/data2
```

Note: Repeat this step for all the dbroots that are configured in your setup.

6. Log into the server where the backup is saved and copy the data from `/usr/local/Calpont/` to the new Insta standby node:

```
[admin@host ~]# rsync -avz
/data/tempMount/destDirTimeStamp/bkupStandBy/
root@insta-new-standby-ip:/usr/local/Calpont/
```

Where `insta-new-standby-ip` is th ip address for the new standby Insta node.

Note: If it fails due to a write permission error, go to the standby node _ `shell` and make the root filesystem rw.

```
host [cluster : standby]> en
host [cluster : standby]# _shell
[admin@host ~]# mount -o remount,rw /
```

7. On the new master Insta node, install `infinidb`:

```
host [cluster : master](config)# insta infinidb install
host [cluster : master](config)# write mem
```

8. Verify the status of the installation:

```
host [cluster : master](config)# insta infinidb get-status-
info
```

Proceed to the next step only if the resulting output is as follows:

```
Infinidb Install status : INSTALLED
and Infinidb Adaptor status : Adaptor Running
```

9. Restart the insta process:

```
host [cluster : master](config)# pm process insta restart
```

Note: This must show all the instances in `RUNNING` state; this may take up to 45 minutes.

To verify the insta process was restarted:

```
host [cluster : master](config)# write mem
host [cluster : master](config)# insta infinidb get-status-
info
```

10. Check the timestamp upto which data has been restored into Insta on the Master Insta node:

```
> en
# _shell
# idbmysql database_mural -e "select * from bin_metatable"
+----------+--------------------+------------+------------+--
-------+-------------+
| binclass | aggregationinterval | mints      | maxts      |
binType | maxexportts |
+----------+--------------------+------------+------------+--
-------+-------------+
| 60min    |                  -1 | 1426147200 | 1429066800 |
 NULL    |  1429066800 |
| 60min    |               86400 |          0 |          0 |
NULL    |           0 |
| 60min    |              604800 |          0 |          0 |
NULL    |           0 |
| 60min    |             2419200 |          0 |          0 |
NULL    |           0 |
+----------+---------------------+------------+------------+--
-------+-------------+
```

Convert the maxts epoch timestamp to readable format for each db:

For example, for DPI:

```
# date -d@1429066800
Wed Apr 15 03:00:00 UTC 2015
```

11. Perform the following steps to set the CubeExporter Job time to the next hour. For example, in this step the next hour is “2015-04-15T04:00Z”:

    a. Run the following commands:

    ```
    > en
    # conf t
    # pmx subshell oozie
    pm extension (oozie)> set job CubeExporter attribute
    jobStart 2015-04-15T05:00Z
    pm extension (oozie)> rollback job CubeExporter
    ```

    b. Set the starttime for BS exporter job to the current time minus 3 day start boundary; so if it is 2015-04-17 today (when all jobs were stopped and recovery is being done), set the start time to 2015-04-14T00:00:

    ```
    pm extension (oozie)> set job BulkStatExporter_15min
    attribute jobStart 2015-04-14T00:00
    pm extension (oozie)> rollback job BulkStatExporter_15min
    ```

    c. Set the starttime for HET exporter job to the current time minus 2 days start boundary; so if it is 2015-04-17 today (when all jobs were stopped and recovery is being done), set the start time to 2015-04-15T00:00:

```
pm extension (oozie)> set job ErrorExport attribute
jobStart 2015-04-15T00:00Z
pm extension (oozie)> rollback job ErrorExport
pm extension (oozie)> run job all
```

12. Restart Rubix on all UI nodes. Log in to each UI node one by one and repeat this step:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# pm process rubix
restart
```

Backing Up and Restoring Information Bases

This topic describes how to:

- Back up information bases (IBs), including manual entries (whitelist entries) from an existing Collector node
- Restore IBs to a new Collector cluster
- Generate and push the Bulkstats, EDR, and Anomaly IBs

Before You Begin

Each of these processes have requirements:

- The backup process requires that the complete setup is up and running for all the specific nodes according to their role.
- The restore process requires that all the nodes are up and running.

If any of these steps fail, do not proceed. Instead, contact Technical Support to resolve the issue.

Backing Up an Information Base

To backup an information base:

1. Log into the master Collector node:

```
 host [cluster : master]> en
host [cluster : master]# _shell
```

2. Take the backup of all the IBs:

```
[admin@host ~]# mkdir -p /data/ib_backup
[admin@host ~]# cd /data/ib/
[admin@host ib]# cp -R work /data/ib_backup
[admin@host ib]# cp -R inbox /data/ib_backupadmin@host ~]#
mkdir -p /data/gateway_backup
[admin@host ~]# cp /data/configs/gateway/* /data/gateway_
backup/
```

3. Check the directory which contains the backup of all the IB's:

```
[admin@host ib]# ls -lrt /data/ib_backup/
```

The resulting output may resemble:

```
total 8
drwxr-xr-x 2 admin root 4096 Jan 31 13:56 work
drwxr-xr-x 2 admin root 4096 Jan 31 13:56 inbox
```

```
admin@host ib]# ls -l /data/gateway_backup/
```

The resulting output may resemble:

```
total 8
-rw-r--r-- 1 admin root   25 Aug 24  2014 collector_ip.conf
-rw-r--r-- 1 admin root 1990 Aug 24  2014 gateway.conf
```

4. Copy the `/data/ib_backup` and `/data/gateway_backup` directories to a SAN storage or a disk drive where you want to save the backup of all the manual IBs in the system.

Restoring an Information Base

Restoring IBs to a Collector Cluster

1. Configure the Collector cluster per the *MURAL Installation Guide*.
2. Download the `/data/ib_backup` and `/data/gateway_backup` folders, which you saved to the SAN storage or a disk drive in the last step, and save it to the `/data` directory on the master Collector node.
3. Run this command to look at the files downloaded in the previous step:

```
[admin@host ~]# cd /data
[admin@host data]# ls -lrt ib_backup/
```

The resulting output may resemble:

```
total 8
drwxr-xr-x 2 admin root 4096 Jan 31 13:56 work
drwxr-xr-x 2 admin root 4096 Jan 31 13:56 inbox
```

4. Copy the IB files (identified in the previous step) to the respective

directories on the new Collector cluster:

```
[admin@host data]# cp -R ib_backup/inbox /data/ib/
[admin@host data]# cp -R ib_backup/work /data/ib/
admin@host data]# cp gateway_backup/* /data/configs/gateway/
```

Verify that all IBs have been copied from backup correctly.

Restoring IBs - for Bulkstats

Log into the master Collector node and perform the following steps from CLI:

Note: The master Collector node is the one which shows `master` in the prompt when you log into the CLI shell.

1. Add the gateways as per the restore IB and the gateway.conf files through `aggregation_center` subshell on the Collector + Namenode master node:

   Before running the commands to add gateways, obtain the gateway name, DC name, Region, Location, IP, timezone, edr-filename-pattern, type, edr-file-path, bulkstat-filename-pattern and bulkstat-file-path from the gateway.conf file.

   The `gateway.conf` file is available at `/data/configs/gateway/`. It contains a block enclosed with curly braces for each gateway. Within that block , the “parameter” : “value” can be found for all the mentioned parameters.

   Note: The `/data/collector` string must be removed from the edr-file-path parameter value and then the remainder string must be used in the add gateway command.

   For example:

   ```
   {
   "Name": "DC1",
   "Associated Region": "R1",
   "IP": "192.168.151.87",
   "Dc": "DC1",
   "file_pattern": "*_*_*_%MM%DD%YYYY%hh%mm%ss_*.gz",
   "Location": "A1",
   ```

```
"Schema Version": "17",
"timezone": "UTC",
"input_dir": "/data/collector/DC1/edr/",
"Type": "HA",
"output_dir": {
 "http": "/data/collector/edrhttp",
 "flow": "/data/collector/edrflow",
 "asn": "/data/collector/edrAsn"
 }
},
 {
 "Name": "DC1",
"Associated Region": "R1",
"IP": "192.168.151.87",
"Dc": "DC1",
"file_pattern": "*_%YYYY%MM%DD%h h%mm%ss",
"Location": "A1",
"Schema Version": "17",
"timezone": "UTC",
"input_dir": "/data/collector/DC1/bs/",
"Type": "HA",
"output_dir": {"DC1": "/data/collector/bulkstats_files/DC1"}},
```

The `add gateway` command in the succeeding steps use values from this block for the gateway DC1.

For parameters "edr-collector-filename-pattern" and "bulkstat-collector-filename-pattern" to be given in the `add gateway` command, use the filename format values saved in the step 10 (d) in "Backing Up and Restoring Information Bases" on page 181.

For the "collector-ip" parameter in the `add gateway` command, provide a comma separated list of the collector IPs (use Data Network IPs if present otherwise the management network IP) that will be receiving the data from this gateway.

```
> en
# conf t
# pmx subshell  aggregation_center
pm extension (aggregation center)> add gateway name DC7 region
R7 location A7 schema_version 18 ip 192.168.151.93 timezone
UTC edr-filename-pattern *_*_*_%MM%DD%YYYY%hh%mm%ss_*.gz
bulkstat-filename-pattern *_%YYYY%MM%DD%hh%mm%ss edr-
collector-filename-pattern %DC_*_*_%MM%DD%YYYY%hh%mm%ss_*.gz
bulkstat-collector-filename-pattern *_%YYYY%MM%DD%hh%mm%ss
collector-ip 192.168.193.225,192.168.192.178 edr-file-path
/DC7/edr/ bulkstat-file-path /DC7/bs/ type HA
```

Repeat this step for each unique gateway name and its corresponding parameter values present in the gateway.conf file.

2. Update the timezone to "UTC" for BulkStats in the gateway.conf file on the master Collector + Namenode cluster:

```
# _shell
# vi /data/configs/gateway/gateway.conf
# cli -m config
```

3. Go to the pmx bulkstats subshell.

```
> en
# conf t
(config)# pmx subshell subshell-name
```

4. On the same bulkstats subshell, continue by generating IBs:

```
pm extension (subshell-name)> generate all ibs
```

Example output of this command is:

```
[key.map]:
    generated key.id.map
[gateway.map]:
    generated gateway.id
    generated version.id
    generated dc.id
    generated region.id
    generated area.id
    generated gatewayRegionArea.id.map
[schema.map]:
    generated schema.id.map
[metric.map]:
    generated metric.id.map
[gatewayIDVersion.map]:
Summary:
========
Successful IBs : 5 out of 5
Failed IBs : No id generation failures.
```

5. Push IBs to the required nodes:

   a. If you made numerous changes to the IBs, or they are going to several destinations, you should push all IBs (to all destinations) using the following command:

   ```
    pm extension (bulkstats)# push all ibs
   ```

   Note: Depending upon the destinations and the size of the IBs being pushed, this step may take about 15 minutes to complete. Moreover, IBs are automatically synced on the standby node.

   b. If you only modified a select few IBs, then pushing all IBs is not necessary. Run the following command instead:

   ```
   pm extension (bulkstats)# push IB to IB_destination
   ```

Note: If the push is interrupted before it completes, you need to restart it.

Restoring IBs - for Anomaly

1. Log into the master Collector and change the port to 11111:

   Note: Go to `_shell` and invoke a new CLI shell before changing the port. Otherwise the port change does not take effect.

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# _shell
   [admin@host]# cli -m config
   [admin@host]# sm service-info modify ps-server-1 port new-
   port-number
   [admin@host]# write mem
   ```

2. From the pmx `anomaly`subshell, generate IBs and push them to required nodes.

   ```
   (config) # pmx
   pm extension > subshell anomaly
   pm extension (anomaly)> update all ibs
   ```

   Note: IBs will be automatically synced on standby node automatically

3. Now take the backup of latest IBs again, and ensure that the backups are moved to a tape or SAN disk so that they are available in case the current system goes down.

Restoring IBs - for EDR

1. Log into the master Collector and change the port to 11111:

   Note: Go to `_shell` and invoke a new CLI shell before changing the port. Otherwise the port change does not take effect.

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# _shell
   [admin@host]# cli -m config
   [admin@host]# sm service-info modify ps-server-1 port new-
   port-number
   [admin@host]# write mem
   ```

2. From the pmx `aggregation_center` subshell, generate IBs and push them to required nodes.

   Note: If any IB fails or times out, then generate IBs again.

   ```
   pm extension (subshell-name)> generate all ibs
   pm extension (subshell-name)> push all ibs
   pm extension (subshell-name)> push gateway configuration
   pm extension (subshell-name)> quit
   pm extension> quit
   host [cluster : master](config)# write memory
   ```

   The server starts copying the various files required for data annotation to the added `ib_destinations`.

   Note: Depending upon the destinations and the size of the IBs being pushed, this step may take about 15 minutes to complete. Moreover, IBs are automatically synced on the standby node.

3. Now take the backup of latest IBs again, and ensure that the backups are moved to a tape or SAN disk so that they are available in case the current system goes down.

Backing Up Raw Data

All of the input feeds coming into the input directory of the Collector are processed and moved automatically to the backup-directory configured for that particular adapter on the same SAN drive mounted on the Collector node.

By default, the files are deleted from the backup directories after 24 hours.

This section describes how to backup and restore raw records received on the Collector node. The process for backing up EDR flow and bulkstats files is similar.

Before You Begin

1. Log into the master Collector node to verify the location of the input directories configured in the system:

   ```
   # cli -t "en" "conf t" "show runn full" | grep "input-
   directory" | awk -F ' ' '{print $9}'
   ```

   This command shows all input directories configured in the system per adaptor (for EDRs) and ASR gateway (for bulkstats). The resulting output may resemble:

   ```
   /data/collector/bulkstats_files/GMPLAB1/
   /data/collector/bulkstats_files/GMPLAB2/
   /data/collector/edrflow
   /data/collector/edrhttp
   ```

2. Verify that the raw data is being received by the Collector in the configured input directories:

   ```
   # watch "ls -lrt /data/collector/edrflow"
   ```

Backing Up Data

1. Log into the master Collector node to check the location of backup directories configured for various adapters in the system:

   ```
   host [cluster : master]> en
   host [cluster : master]# _shell
   ```

```
[admin@host ~]# cli -t "en" "conf t" "show runn full" | grep
"backup-directory" | awk -F ' ' '{print $9}'
```

This command shows the location of all the backup directories per adapter (for EDR http and flow) and per gateway (for bulkstats):

```
/data/collector/bulkstats_files_backup/GMPLAB1/
/data/collector/bulkstats_files_backup/GMPLAB2/
/data/collector/edrflow_backup
/data/collector/edrhttp_backup
```

2. Back up the one day old raw data for any adapter to a new folder locally or onto a new SAN drive.

   For example, to back up http files:

```
[admin@host ~]# mkdir -p /data/http_rawdata
[admin@host ~]# cp -R /data/collector/edrhttp_backup
/data/http_rawdata
```

3. To fetch the bulkstats file for the previous day, run the following commands to fetch it directly from HDFS:

```
# mkdir -p /data/bulkstats_rawdata
# cd /data/bulkstats_rawdata
# hdfs dfs -get
/data/collector/1/output/bulkStats/<YEAR>/<MONTH>/<DAY>/ .
```

   Here, `<YEAR>/<MONTH>/<DAY>` can be `2015/03/18`.

4. Use `tar` or `gzip` to compress this new folder, and then copy it to the required backup location.

Backing Up and Restoring PGSQL DB

This document describes how to back up and restore Pstgres (PGSQL) database data on a master GMS node, Collector (Name) node or the Insta node in case PGSQL database needs to be freshly configured due to corrupt data.

First, verify that the PGSQL database is running on the master node (GMS, Name (Collector) node, or Insta node) by running the following command:

```
(config) # show pm process pgsqld
State:
      Current status:  running
```

If any of these steps fail, do not proceed. Instead, contact Technical Support to resolve the issue.

Backing-Up the PGSQL Database

To back-up an PGSQL database at Insta node:

1. Log into the master UI node and go to the CLI mode.
2. Stop all java processes running on the system:

   ```
   (config) # pm process rubix terminate
   ```

3. On the standby UI node, repeat the previous step.
4. In case of Starter and Medium Pack setups, log in to the master Name node and stop the Hive jobs that are currently running. To check which Hive jobs are running, use the `show coordinator RUNNING jobs` and `show coordinator PREP jobs` commands.

   ```
   # pmx
   pm extension> subshell oozie
   pm extension (oozie)> stop jobname AnomalyHiveHourly
   pm extension (oozie)> stop jobname AnomalyHiveDaily
   pm extension (oozie)> stop jobname AnomalyHiveAggDaily
   pm extension (oozie)> stop jobname AnomalyHiveMonthly
   pm extension (oozie)> stop jobname AnomalyCleanupHiveHourly
   ```

```
pm extension (oozie)> stop jobname AnomalyCleanupHiveDaily
pm extension (oozie)> stop jobname AnomalyCleanupHiveAggDaily
pm extension (oozie)> stop jobname AnomalyCleanupHiveMonthly
pm extension (oozie)> stop jobname bsHiveJob
pm extension (oozie)> stop jobname bsCleanupHive
pm extension (oozie)> stop jobname edrFlowHiveJob
pm extension (oozie)> stop jobname edrHttpHiveJob
pm extension (oozie)> stop jobname edrflowCleanupHive
pm extension (oozie)> stop jobname edrhttpCleanupHive
```

5. To take a backup of PGSQL database on Name node (required only in case of Standard pack setup), perform step 4 to stop all the Hive jobs.
6. To take a backup of PGSQL database on GMS node, stop the running GMS Server running on master node:

```
(config) # pm process gms_server terminate
(config) # show pm process gms_server
State:
Current status:  stopped
```

7. On the master (GMS, Insta, or Name Node (Collector)) node, take a backup of the database:

```
(config) # pgsql file-path /data/pgsql_bkup dbname all backup
```

8. Save the backup as a tar file:

```
# cd /data
# tar -zcvf pgsql_bkup.tgz /pgsql_bkup
```

Note: Move this file to a safe location or an external storage available.

The pgsql backup of each module – GMS, Insta, Name node must be saved with a unique name as all of these mainatain different types of data. At the time of restore, use the backup DB file corresponding to the node whose pgsql DB has failed.

Restart Processes on the UI Nodes

Note: Perform this procedure only after the data is backed up on the Insta node.

1. Log into the master UI node and go to the CLI mode.
2. Restart EDR tomcat on the master UI node:

```
(config) # pm process rubix restart
```

3. Log into the standby UI node, repeat the previous step.

Restart Jobs on Master Name (Collector) Node

Note: Perform this procedure only after the data is backed up on the Name (Collector) node.

To restart jobs on the master Collector node:

1. Log in to the master Collector node and go to the CLI mode.
2. Restart all the HIVE jobs:

```
# pmx
pm extension> subshell oozie
pm extension (oozie)> run job AnomalyHiveHourly
pm extension (oozie)> run job AnomalyHiveDaily
pm extension (oozie)> run job AnomalyHiveAggDaily
pm extension (oozie)> run job AnomalyHiveMonthly
pm extension (oozie)> run job AnomalyCleanupHiveHourly
pm extension (oozie)> run job AnomalyCleanupHiveDaily
pm extension (oozie)> run job AnomalyCleanupHiveAggDaily
pm extension (oozie)> run job AnomalyCleanupHiveMonthly
pm extension (oozie)> run job bsHiveJob
pm extension (oozie)> run job bsCleanupHive
pm extension (oozie)> run job edrFlowHiveJob
pm extension (oozie)> run job edrHttpHiveJob
pm extension (oozie)> run job edrflowCleanupHive
pm extension (oozie)> run job edrhttpCleanupHive
```

Restart GMS Process on Master GMS Node

Note: Perform this procedure only after the data is backed up on the master GMS node.

1. Run the following commands:

```
(config) # pm process gms_server restart
(config) # show pm process gms_server
State:
Current status:  running
```

Restoring a PGSQL Database

Caution: Data such as filters and users that were created after the last backup of the pgsql DB, will be lost.

To restore a PGSQL database on an iNSTA node:

1. Log into the master UI node and go to the CLI mode.
2. Stop all Java processes on the system:

```
(config) # pm process rubix terminate
```

3. Log in to the standby UI node, repeat the previous step.
4. In case of Starter and Medium Pack setups, log in to the master Name node (Collector node) and stop Hive jobs:

```
# pmx
pm extension> subshell oozie
pm extension (oozie)> stop jobname AnomalyHiveHourly
pm extension (oozie)> stop jobname AnomalyHiveDaily
pm extension (oozie)> stop jobname AnomalyHiveAggDaily
pm extension (oozie)> stop jobname AnomalyHiveMonthly
pm extension (oozie)> stop jobname AnomalyCleanupHiveHourly
pm extension (oozie)> stop jobname AnomalyCleanupHiveDaily
pm extension (oozie)> stop jobname AnomalyCleanupHiveAggDaily
pm extension (oozie)> stop jobname AnomalyCleanupHiveMonthly
pm extension (oozie)> stop jobname bsHiveJob
pm extension (oozie)> stop jobname bsCleanupHive
pm extension (oozie)> stop jobname edrFlowHiveJob
pm extension (oozie)> stop jobname edrHttpHiveJob
pm extension (oozie)> stop jobname edrflowCleanupHive
pm extension (oozie)> stop jobname edrhttpCleanupHive
```

5. To restore an PGSQL database on Name node in Standard pack setup, repeat the previous step on the master name node.

6. To restore an PGSQL database on GMS node, stop the running GMS Server running on the master node:

```
(config) # pm process gms_server terminate
(config) # show pm process gms_server
State:
Current status:  stopped
```

7. Check if the PGSQL data directory is clean and the PGSQL process is running on the master (GMS, Insta, or Name) Node:

```
(config) # pm process pgsqld terminate
(config) # _shell
# cd /data/pgsql
# rm -rf *
# cli -m config
(config) # pm process pgsqld restart
(config) # show pm process pgsqld
Current status:  running
```

   Repeat the preceding commands on the standby GMS, NameNode(Collector), and Insta node.

8. Restore the backed up PGSQL DB on the master (GMS, Insta, or Name) node:

```
(config) # _shell
# cd /data
# tar -xvzf /data/pgsql_bkup.tgz
# cli -m config
(config) # pgsql file-path <path to backup file that has to be
restored> dbname <name of db/all> restore
```

For example:

```
(config)# pgsql file-path /data/pgsql_bkup dbname all restore
```

9. On the master Namenode (Collector node), restart all Hive Jobs that were stopped earlier:

```
# pmx
pm extension> subshell oozie
pm extension (oozie)> run job AnomalyHiveHourly
pm extension (oozie)> run job AnomalyHiveDaily
pm extension (oozie)> run job AnomalyHiveAggDaily
pm extension (oozie)> run job AnomalyHiveMonthly
pm extension (oozie)> run job AnomalyCleanupHiveHourly
pm extension (oozie)> run job AnomalyCleanupHiveDaily
pm extension (oozie)> run job AnomalyCleanupHiveAggDaily
pm extension (oozie)> run job AnomalyCleanupHiveMonthly
pm extension (oozie)> run job bsHiveJob
pm extension (oozie)> run job bsCleanupHive
pm extension (oozie)> run job edrFlowHiveJob
pm extension (oozie)> run job edrHttpHiveJob
pm extension (oozie)> run job edrflowCleanupHive
pm extension (oozie)> run job edrhttpCleanupHive
```

10. Log into the master UI node and restart processes:

```
(config) # pm process rubix restart
```

11. Log into the standby UI node and repeat the previous step.

12. Verify that you can access the UIs:

From a browser window, go to the URL `https://<domainName>:21443/`.

For example,

```
https://demo.sanmateo.com:21443/
Username: admin
Password: admin123
```

13. Go to the **Configure** tab in the UI and check that all old users, whose accounts were created on the setup from which backup is taken, are still available.

14. Restart the GMS process on the master GMS Node that was stopped earlier:

```
(config) # pm process gms_server restart
(config) # show pm process gms_server
  State:
      Current status:  running
```

Troubleshooting

Troubleshooting User Interface Access Problems

This topic describes the most common causes of trouble in accessing the MURAL user interface from a browser.

UI Not Accessible

In this scenario, the user interface is unavailable, the URL is not accessible, or you are unable to log into the UI.

Checking Connectivity to UI

1. Check to see that the UI node URL and Management IP is reachable. If IP is reachable but URL is not, then it could be a DNS issue or mapping of the URL to IP on your local PC might be missing. If IP is also not reachable, follow steps for network connectivity troubleshooting.
2. Check that the URL has the correct **IP: port number**.

Troubleshooting Connectivity Issues

1. Log into the UI node, and go to the _shell prompt.
2. Issue a `ps` command and verify that the rubix app `atlas` is running. If launcher is configured, then launcher must also be in the running state:

```
ps - ef | grep tomcat
```

3. If the atlas app does not show the pid, then run the following command:

```
# cli -m config
(config)# rubix modify-app atlas enable
(config)# rubix modify-app atlas modify-instance 1 enable
```

4. After two minutes check that atlas app is getting listed under running processes. Go to the `_shell` prompt and run:

```
ps - ef | grep atlas
```

5. If the `atlas` app is still not shown under running processes, contact Cisco Technical Support.

Cannot Log In

In this scenario, the UI opens, but you cannot log in or the UI is stuck with spinning gears.

1. Check that the PGSQL process is running on the master iNSTA node:

```
host [cluster : master]> en
host [cluster : master]# conf t
host [cluster : master](config)# show  pm process pgsqld
```

2. If the status is displayed as "not running", start the PGSQL process:

```
host [cluster : master](config)# pm process pgsqld restart
host [cluster : master](config)# show  pm process pgsql
```

   The resulting output may resemble:

```
Current status:  running
```

3. If the UI is still not accessible:
   a. Log into both UI nodes.
   b. Save the UI log files to the FTP server.
   c. Contact Technical Support.

UI does not display data due to issues related with IBs

In this scenario, the UI opens, but data is not displayed correctly after updating IBs. This issue may be encountered, when:

- New IB entries that were manually added are not updated successfully during the upgrade.
   1. Do one of the following as required:
      - Create a map file containing new entries and an empty line at the end. The structure of the new entry must be similar to the existing entries in the map file. If an empty line is not found in

the map file, the "Empty Line expected after last field." message is displayed when the IB is updated.

- Create a list file containing new entries and an empty line at the end. The structure of the new entry must be similar to the existing entries in the list file.

2. Edit the IB:

```
host [cluster : master](config)# edit ib <name-of-ib> add
bulk <path-of-the-file>
```

For example,

```
host [cluster : master](config)# edit ib exchangeDC.map
add bulk /data/exchange
```

3. Verify the new entries in the *file-name*.checkstyle file. The new entries have `ADD` at the beginning.

- IB is not available at the required location.

1. Ensure that an IB is available at the correct location. If it is not, copy the IB manually to the correct location. For example, dcLocation.map IB must be available on the `/data/atlasData` directory.

2. Restart Rubix on the UI node.

If the UI still does not display the correct data, contact Technical Support with relevant logs.

Troubleshooting Abrupt Failure of Jobs

This topic provides information about preliminary diagnosis and analysis if data processing has stopped on the Reflex MURAL system.

Note: If any deviation from the expected behavior is observed while debugging various nodes, note and report them to Cisco Technical Support along with all the logs that are collected.

If there are any problems with the jobs, data on the RGE is not processed as scheduled.

Checking the Bin Times

One of the main functions of the Collector node is assigning time-stamps to the data being processed. This section will guide you through steps to verify this process is working properly.

1. Generate a list of the current database names being used by running:

```
host [cluster : master]> en
host [cluster : master]# _shell
[admin@host dir]# cli -t "en" "conf t" "show run full" | grep
-I "cubes-database"
```

   Database names are against each instance number: 0 for EDR and 1 for bulkstats.

   The resulting output may resemble:

```
insta instance 0 cubes-database database mural
insta instance 1 cubes-database bulkstats
```

2. Run the following commands and make sure the `mints` and `maxts` are not all zero for each DB:

```
host [cluster : master]> en
host [cluster : master]# _shell
```

```
[admin@host dir]# idbmysql <edr_database_name> -e "select *
from bin_metatable";
```

For example,

```
[admin@host dir]# idbmysql database mural -e "select * from
bin_metatable"
+----------+----------------+------------+-----------+------+
| binclass | aggregationinterval | mints  | maxts  | bintype |
+----------+----------------+------------+-----------+------+
| 60min    |             -1 | 1358326800 | 1358359200 | NULL |
| 60min    |          86400 |          0 |          0 | NULL |
| 60min    |         604800 |          0 |          0 | NULL |
| 60min    |        2419200 |          0 |          0 | NULL |
+----------+----------------+------------+-----------+------+
[admin@host dir]# idbmysql bulkstats -e "select * from bin_
metatable";
+----------+----------------+------------+-----------+------+
| binclass | aggregationinterval | mints  | maxts  | bintype |
+----------+----------------+------------+-----------+------+
| 5min     |             -1 |          0 |          0 | NULL |
| 5min     |            900 | 1358327700 | 1358366400 | NULL |
| 5min     |           3600 | 1358326800 | 1358362800 | NULL |
| 5min     |          86400 |          0 |          0 | NULL |
| 5min     |         604800 |          0 |          0 | NULL |
| 5min     |        2419200 |          0 |          0 | NULL |
+----------+----------------+------------+-----------+------+
```

a. In this example, output from the first table (EDR) which needs to be considered are the `mints` and `maxts` values in the row which has `aggregationinterval` = `-1` and `binclass` = `60min`.

   For example, `mints` = 1358326800, `maxts` = 1358359200

b. Similarly, output from the second table (bulkstats) which needs to be considered are the `mints` and `maxts` values in the row which has `aggregationinterval = 900` and `binclass = 5min` will be considered.

For example, `mints` = 1358327700, `maxts` = 1358366400

3. Check the function of the **date** command using the **mints** and **maxts** values identified in step 2a and 2b, above:

   ```
   [admin@host dir]# date -d @1358326800
   Wed Jan 16 09:00:00 UTC 2013
   [admin@host dir]# date -d @1358359200
   Wed Jan 16 18:00:00 UTC 2013
   ```

   Note: This shows we have processed data from 09:00 to 18:00 on Jan 16 in Insta DB for EDR.

   ```
   [admin@host dir]# date -d @1358327700
   Wed Jan 16 09:15:00 UTC 2013
   [admin@host dir]# date -d @1358366400
   Wed Jan 16 20:00:00 UTC 2013
   ```

   Note: This shows we have processed data from 09:15 to 20:00 on Jan 16 in Insta DB for bulkstats.

4. If things are operating as expected on Insta, the maxts of both databases should show as 1 hour less than the maxts shown on UI.

After performing all steps in this section on "Troubleshooting Abrupt Failure of Jobs" on page 202, consider:

- If no anomaly is observed, then the Insta nodes are operating as expected and you need to proceed to the next step of troubleshooting the abrupt failure of jobs.
- If one or more anomaly is observed, notify Cisco Technical Support and provide the log files as explained in "System Logs" on page 33.

Debugging Collector Nodes

If data processing has stopped on the MURAL system, this procedure will help identify whether the issue is on Collector nodes.

Checking the Status of Processes

1. Log into the master Collector node and go to `_shell`:

   ```
   host [cluster : master]# cli -m config
   (config)# _shell
   ```

2. Check the Collector processes by running:

   ```
   [admin@host dir]# cli -t "en" "config t" "show pm process tps"
   | grep "Current status"
         Current status:  running

   [admin@host dir]# cli -t "en" "config t" "show pm process
   collector" | grep "Current status"
         Current status:  running
   ```

3. Verify if the timezone conversion script is running:

   ```
   [admin@host ~]# ps -ef | grep -i collector_file_generator |
   grep -v grep
   ```

   The resulting output may resemble:

   ```
   admin     9405     1  0 Jun14 ?        00:00:22
   /usr/bin/python /opt/etc/scripts/collector_file_generator.py
   /data/asr_file_processor/config.json 10
   ```

   If the timezone conversion script is not running:

   - Check `/var/log/messages` for any errors related to this script
   - Collect all logs as per "System Logs" on page 33
   - Contact Cisco Technical Support

4. Repeat steps 1 and 2 on the standby Collector node.

5. Log into the master Collector node and go to `_shell`:

```
host [cluster : master]> en
host [cluster : master]# _shell
```

6. Verify the relevant hdfs processes (as displayed in the sample output) are up or not:

```
[admin@host dir]# ps -ef | grep java | grep Dproc | grep -v
Dproc_dfs | awk '{print $9}'
```

The resulting output may resemble:

```
-Dproc_datanode
-Dproc_namenode
-Dproc_jobtracker
-Dproc_secondarynamenode
-Dproc_namenode
```

7. Run the following command on the standby Collector node:

```
# ps -ef | grep java | grep Dproc | grep -v Dproc_dfs | awk '
{print $9}'
```

The resulting output may resemble:

```
-Dproc_namenode
-Dproc_namenode
-Dproc_datanode
-Dproc_secondarynamenode
-Dproc_journalnode
```

8. If all of the processes listed are correct, then proceed to the next step of troubleshooting.

Checking done.txt Jobs

1. Log into the master Collector node and go to `_shell`:

```
host [cluster : master]> en
host [cluster : master]# _shell
```

2. Check the various jobs listed in `done.txt` by running:

```
[admin@host dir]# hdfs dfs -cat /data/job-name/done.txt
```

Where `job-name` is replaced with:

- EDR
- CubeExporter
- Bulkstat
- BSAgg15min
- BulkStatExporter_15min

The resulting output may resemble:

```
2013-06-14T09:00Z
```

Note: Repeat this command until all job-names have been checked.

If the jobs listed have the same time-stamp (+1 hour) as that of current time, then there are no problems with the communication between the jobs and the Insta node, and you should continue with the next section on troubleshooting.

If the jobs listed do not show the current system time (-1 or 2 hour), then either the jobs have an issue or there is no input data coming to the system.

Checking Most Recently Processed Data

Find the last bin processed by collector and use that information to determine what date and time the data was processed by the Collector.

1. Log into the master Collector node:

```
[admin@host]# cli -m config
```

2. Run the following command:

```
host [cluster : master|standby](config)# collector stats
instance-id 1 adaptor-stats process-name last-freezed-bin
```

Where `process-name` is netflowis:

- edrhttp
- edrflow
- bukstats

Note: Repeat this command until all process-names have been checked.

3. The output shows the epoch time. Convert the time by going to bash shell mode and running:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# _shell
date -d @<epoch_time>
```

4. If this is not the latest 5 min and it is around the same time as the jobs' `done.txt` timestamps (can be up-to +1 hour), then it means jobs are okay and the Collector itself is either not receiving or not processing any data.

5. If the most recently processed data is dated within the last 5 minutes or so, progress to the next step in troubleshooting.

Verify Receipt of Files on MURAL

This step will check whether or not files are coming into the MURAL system from the ASR end, and being stored in the input Collector directory as configured in `config.json`.

1. Log into the master Collector node and go to `_shell`:

```
host [cluster : master]> en
host [cluster : master]# _shell
```

2. Run these commands:

```
[admin@host dir]# cd /data/configs/
[admin@host dir]# cat gateway.conf
```

The resulting output may resemble:

```
[
 {
"Location": "INDIA",
"Name": "Delhi",
"IP": "10.10.10.255",
"file_pattern": "*_MURAL-edr_*_%MM%DD%YYYY%hh%mm%ss.*",
"output_dir": {
               "flow": "/data/collector/edrflow",
               "http": "/data/collector/edrhttp"
              },
"Schema Version": "15",
"timezone": "Asia/Kolkata",
"input_dir": "/data/collector/DC1/",
"Type": "HA",
"Associated Region": "EAST"
 },
"Location": "INDIA",
"Name": "Delhi",
"IP": "10.10.10.255",
"file_pattern": "*_%YYYY%MM%DD%hh%mm%ss",
"output_dir": {
              "Delhi": "/data/collector/bulkstats_files/Delhi"
              },
"Schema Version": "15",
"timezone": "Asia/Kolkata",
"input_dir": "/data/collector/GW1/",
"Type": "HA",
"Associated Region": "EAST"
},
```

3. In a live and running system ASRs are configured to send data on the various `input_dir` configured in `gateway.conf` through Wrapper CLI. Therefore, you should monitor the configured directories.

   a. Log into the master Collector unix terminal, run the following code to set-up a watchlist where *`name`* is `DC1`:

   ```
   host [cluster : master]> en
   host [cluster : master]# _shell
   [admin@host dir]# watch 'ls -lrt /data/collector/name/'
   ```

   b. Log into another master Collector unix terminal and repeat the previous step with `GW1` as the *name*.

4. If files are not coming after monitoring directories for 10 -15 minutes, there may be a connectivity issue from the ASR. In this case, your organization's IT Support Team needs to resolve the issue. Contact your IT Support Team for assistance with this issue.

5. If files are coming in, progress to the next step in troubleshooting.

Verifying the Amount of Local Disk Space on the Collector

You need to ensure there is enough local disk space on the `/data` and `/var` mount points because if not, the collector and its jobs may not be able to process any data, causing jobs to crash.

1. To verify log into the master Collector node and go to `_shell`:

```
host [cluster : master]> en
host [cluster : master]# _shell
```

2. Verify that there is enough disk space (Use% less than 90 percent) on `/data` and `/data/collector` by running:

```
[admin@host dir]# df -kh
```

   The resulting output may resemble:

```
Filesystem            Size  Used Avail Use% Mounted on
rootfs                4.0G  2.0G  1.9G  52% /
/dev/root             4.0G  2.0G  1.9G  52% /
...
/dev/sda11            498G   84G  389G  18% /data
none                   48G     0   48G   0% /dev/shm
...
/dev/mapper/mpath5    5.0T  152G  4.6T   4% /data/collector
/dev/drbd0            5.0G  172M  4.6G   4% /data/drbd
```

3. Log into the NameNode and go to `_shell`:

```
host [cluster : master]> en
host [cluster : master]# _shell
```

4. Ensure that there is enough disk space (DFS Used% less than 90 percent) on HDFS. Also verify that HDFS processes are up and running in the report:

```
[admin@host dir]# hdfs dfsadmin -report
```

The resulting output may resemble:

```
Configured Capacity: 3246762123264 (2.95 TB)
 Present Capacity: 3081828670132 (2.8 TB)
 Configured Capacity: 3246762123264 (2.95 TB)
 Present Capacity: 3081828670132 (2.8 TB)
 DFS Remaining: 186530689024 (173.72 GB)
 DFS Used: 2895297981108 (2.63 TB)
 DFS Used%: 93.95%
 Under replicated blocks: 72
 Blocks with corrupt replicas: 0
 Missing blocks: 0
 -------------------------------------------------
 Datanodes available: 3 (3 total, 0 dead)
 Name: 10.10.2.18:50010
 Decommission Status : Normal
 Configured Capacity: 1082254041088 (1007.93 GB)
 DFS Used: 1000751046988 (932.02 GB)
 Non DFS Used: 55154527924 (51.37 GB)
 DFS Remaining: 26348466176(24.54 GB)
 DFS Used%: 92.47%
 DFS Remaining%: 2.43%
 Last contact: Wed Nov 07 10:04:41 GMT 2012

 Name: 10.10.2.14:50010
 Decommission Status : Normal
 Configured Capacity: 1082254041088 (1007.93 GB)
 DFS Used: 906842024426 (844.56 GB)
 Non DFS Used: 54873441814 (51.1 GB)
 DFS Remaining: 120538574848(112.26 GB)
 DFS Used%: 83.79%
 DFS Remaining%: 11.14%
 Last contact: Wed Nov 07 10:04:42 GMT 2012

 Name: 10.10.2.13:50010
 Decommission Status : Normal
 Configured Capacity: 1082254041088 (1007.93 GB)
 DFS Used: 987704909694 (919.87 GB)
 Non DFS Used: 54905483394 (51.13 GB)
 DFS Remaining: 39643648000(36.92 GB)
 DFS Used%: 91.26%
 DFS Remaining%: 3.66%
 Last contact: Wed Nov 07 10:04:40 GMT 2012

...
DFS Used%: 83.79%
DFS Remaining%: 11.14%
...
Name: 10.10.2.13:50010
...
DFS Used%: 91.26%
DFS Remaining%: 3.66%
Last contact: Wed Nov 07 10:04:40 GMT 2012
```

If the disk space is less in HDFS, verify that the CleanUp Jobs are running or have been successful in cleaning the old data from the HDFS. For more information about how to run a job, see "Starting and Stopping Jobs " on page 42

5. If the disk usage on the HDFS shows a very high percentage, then also check HDFS disk space on `hadoop /data/` to see all directories which are

taking more space and report them to Cisco Technical Support.

```
[admin@host dir]# hdfs dfs -du /data
```

The resulting output may resemble:

```
Found 39 items
18         hdfs://Namenode-VIP:9000/data/AggregationJobDaily
18         hdfs://Namenode-VIP:9000/data/AggregationJobWeekly
18         hdfs://Namenode-VIP:9000/data/BSAgg15min
18         hdfs://Namenode-VIP:9000/data/BSAgg1Day
18         hdfs://Namenode-VIP:9000/data/BSAgg1hour
18         hdfs://Namenode-VIP:9000/data/BulkStat
...
```

6. If the collector data is as per latest system time and only jobs are lagging behind with no disk space issues, proceed to the next troubleshooting step.

Checking the Status of Failed Jobs

If you notice jobs that are lagging behind when running the steps from the previous section, follow these steps to look for errors in the `stdout` and `stderr` files.

1. Log into the master Collector node and go to the pmx oozie subshell:

```
> en
# conf t
(config)# pmx subshell subshell-name
```

2. Run this command to see the headers for the columns of output to be generated in the next step:

```
pm extension (oozie)> show workflow
```

The resulting output may resemble:

```
FAILED     KILLED     PREP       RUNNING    SUCCEEDED  all
```

3. Run this command to determine the job ID for the job you are checking on:

```
pm extension (oozie)> show workflow RUNNING jobs
```

For this example, the Job ID of the `CubeExporter` job is the first value after the dashed line separating the output.

```
Job ID  App Name    App Path    Console URL User    Group
Run Created Started Status  Last Modified   Ended
--------------------------------------------------------------
0003006-121209074445646-oozie-admi-W    CubeExporter    -
http://CISCO-COL1-147-11:8080/oozie?job=0003006-
121209074445646-oozie-admi-W    admin   users   2012-12-12
06:15    2012-12-12 06:15    RUNNING 2012-12-12 06:15    -
--------------------------------------------------------------
...
```

4. For this example, the job ID for `CubeExporter` is shown in the previous step to be `0003006-121209074445646-oozie-admi-W`.

   Look up any errors associated with that particular process by logging into the master Collector node, go to `_shell`, and run:

   ```
   [admin@host dir]# cd /data/oozie-admi/0003006-121209074445646-
   oozie-admi-W/ checkdone--ssh/    done--ssh/         exporter--
   ssh/     exporterfail--ssh/
   ```

   If after some time the `CubeExporter` job completes with a status of `SUCCEEDED`, the cube exporter has been able to push data to Insta suc-cesfully.

   If not, and `CubeExporter` moved to a `KILLED`/`FAILED` state or remains in `RUNNING` state for more than an hour, check each of the four the folders above for errors in the `stdout` and `stderr` files. You can use one of two approaches:

   a. Run the following command for each folder:

      ```
      [admin@host dir]# cd folder-name
      [admin@host dir]# ls *std*
      ```

Where *folder-name* is replaced with:

- checkdone--ssh
- done--ssh
- exporter--ssh
- exporterfail--ssh

Note: Repeat this command until all folder-names have been checked.

b. Instead of running a command for each folder, you can use this command which to perform the same check on all four folders for one process at the same time:

```
[admin@host dir]# ls -l */*std*
```

Note: If the map reduce job action fails, check the `stderr` file for the HDFS job ID. Make a note of that ID as it will be helpful in debugging and troubleshooting on hdfs for issues.

5. If exporters and other jobs are showing the same `done.txt` timestamp, it indicates that the Collectors are running and the system does not have disk space issues. If this is the case, identify which are the latest jobs which have had issues:

   a. Log into the master Collector and go to the pmx oozie subshell:

   ```
   > en
   # conf t
   (config)# pmx subshell subshell-name
   ```

   b. Find the `FAILED` jobs:

   ```
   pm extension (oozie)> show workflow FAILED jobs
   ```

   c. Find the `KILLED` jobs:

   ```
   pm extension (oozie)> show workflow KILLED jobs
   ```

Debugging Compute Nodes

Perform these troubleshooting steps if any Compute nodes have gone to a dead state as seen in the section on "Verifying the Amount of Local Disk Space on the Collector" on page 210.

1. Log into the nodes whose disk is 100% full and check the processes:

   ```
   # ps -ef| egrep -i "datanode|_nodemanager" |grep -v grep |wc –
   l
   2
   ```

   The output must display the following two processes as running:

   - /usr/java/default/bin/java -Dproc_datanode
   - /usr/java/default/bin/java -Dproc_nodemanager

   The resulting output may resemble:

   ```
   /dev/mapper/mpathep1 on /data/hadoop-admin type ext3
   (rw,relatime,errors=continue,barrier=1,data=ordered)
   ```

2. Check the mount points to determine if they have gone to read only mode:

   ```
   host [cluster : master|standby]# mount | grep hadoop-admin
   ```

   The resulting output may resemble:

   ```
   /dev/mapper/mpathep1 on /data/hadoop-admin type ext3
   (rw,relatime,errors=continue,barrier=1,data=ordered)
   ```

3. Also, check the local disk space on the Compute node:

   ```
   host [cluster : master|standby]# df -kh
   ```

   The resulting output may resemble:

   ```
   Filesystem            Size  Used Avail Use% Mounted on
   rootfs                 40G  2.5G   35G   7% /
   /dev/root              40G  2.5G   35G   7% /
   devtmpfs               48G  312K   48G   1% /dev
   /dev/sda3             130M   12M  111M  10% /boot
   ```

```
/dev/sda1              130M  4.3M  119M   4% /bootmgr
/dev/sda8              190M  5.9M  175M   4% /config
/dev/sda10             427G  3.0G  402G   1% /var
/dev/sdb1              551G  437G   86G  84% /data
tmpfs                   48G     0   48G   0% /dev/shm
none                   3.0G     0  3.0G   0% /guavus/insta
tmpfs                   64M     0   64M   0% /vtmp
/dev/mapper/mpathdp1 1008G  329G  628G  35% /data/hadoop-admin
```

4. If any of the partitions are full, notify Cisco Technical Support and provide the log files as explained in "System Logs" on page 33

Troubleshooting Servers After Chassis Failure

The system was initially setup such that nodes of each module type - Collector, Compute, Insta, Cube (Rubix) Node, and UI were equally distributed across two chassis. This section explains the procedure to recover blade servers that were part of a chassis which failed.

1. Set up the chassis and all its blades.
2. Once hardware setup is done, follow the remaining steps to restore each module.

Collector Node Failure

Replace the failed node with a new Collector node, per the section of this guide on "Recovering a Collector Node" on page 227.

Compute Node Failure

Failures of multiple Compute nodes is not currently supported, and in the event of such a large failure, there may be some data loss. This may require extra steps to rebuild and recover, depending on the system's condition.

For more information, see "Recovering a Compute Node" on page 243.

Insta Node

Follow the steps on "Recovering an Insta Node " on page 252 to install the new Insta node in place of the failed one.

Rubix Node

Follow the steps within "Recovering a Failed UI Node: Scenario 1" on page 258 to install the new UI node in place of the failed one.

Troubleshooting Interruption of Data Flow to Collector Node

This section describes how to troubleshoot and recover from a situation when Collector does not write data to the HDFS for two or three hours. The reason of this interruption can be either collector is dropping the data due to some unexpected format, field, and timestamp or it is not receiving any data in its input path.

If data is not being received on the Collector nodes, it might be due to one of the following reasons:

- Connectivity between ASR and Collector nodes is disrupted. (Link down alarms.)
- Disk space is full on the SAN which is mounted on collector blades which has the input directory on which feeds come.
- Local disk space on the master Collector node is full.

Disk Full Alerts

In the event of a disk full alert on the SAN:

Log into the console via SSH and Identify which partition is full (>80%).

- If the `/var` partition is full, clear older log files on the device.
- If the `/data` partition is full, check for SAN connectivity issues.

If partitions are still full, contact Technical Support.

Restore Data Flow

1. Determine if data was not received on the Collector node by looking at either one of the following traps:

   a. `noDataTrap` for each adaptor on the master Collector node. If this alert appears, one or more of these data flows has stopped receiving data and needs to be reset: `edrhttp`, `or edrflow` or `bulkstats`.

   b. `diskSpaceLow` trap indicates a threshold has been reached on either the Compute node or master Collector node for any of the partitions: `/var`, `/data`, or `/data/hadoop-admin`.

2. Check for a connectivity break between the ASR and the collector blades. If

one is found, contact your local IT support group to resolve the problem with the server. If one is not found, determine if there is a failure at the intermediary network devices connecting the ASR and the Collector.

Once connectivity is restored, verify that Collector data has resumed on SNMP traps for each of the configured adaptors `edrhttp`, `or edrflow` or `bulkstats`).

3. If you see an SNMP trap for high disk usage for `/data` or `/data/collector` (directory where SAN is mounted for collector blades) or `/var`, log into the master NameNode or Collector and run:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# _shell
[admin@host ~]# df -k | awk -F ' ' '{print $6, $5}'
```

   This command displays the percentage of disk space each partition is using. The resulting output may resemble:

```
Mounted Use%
/ 52%
/ 52%
/dev 1%
/boot 9%
/bootmgr 4%
/config 5%
/var 38%
/data 18%
/dev/shm 0%
/guavus/insta 0%
/vtmp 0%
/data/collector 3%
/data/drbd 4%
```

4. Determine the amount of disk space used for a particular mounted drive under each of its subfolders by using the command `du -sh *`.

For example:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# _shell
[admin@host ~]# du -sh /data/*
```

The resulting output may resemble:

```
28M    /data/084.0K    /data/BaseCubes.json.30rc1
 4.0K    /data/ReservedNames.json
 7.4M    /data/apache-tomcat
 4.0K    /data/atlasData
 14G    /data/backup-hadoop-admin
 7.6G    /data/backup_hadoop_logs
 4.0K    /data/changeInstaIPInExporterJobs
 264K    /data/cli_config.txt
 6.7G    /data/collector
 484K    /data/configs
 8.1M    /data/core.4111
 57M    /data/drbd
 74G    /data/edr
 128M    /data/hadoop-admin
 50G    /data/hadoop_logs
 454M    /data/ib
 4.0K    /data/insta
 57M    /data/jboss
 16K    /data/lost+found
 240M    /data/oozie-admi
 169M    /data/oozie-server
 1.5G    /data/oozie_logs
 557M    /data/patch
 279M    /data/prashant
 12K    /data/qedata
 1.6G    /data/rahul
 692K    /data/rahul2
```

```
884M    /data/utils
20K    /data/virt
36K    /data/xx
4.0K    /data/yy
1.6M    /data/zz
```

5. Once the problem for connectivity or disk usage is resolved, check for the input directory on SAN storage mounted on collector blades for each of `edrhttp`, and `edrflow` and `bulkstats`.

6. Check whether files are coming into the input directory:

   a. To find the input directory, log into the Collector cluster and run:

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# _shell
   [admin@host ~]# cli -t "en" "conf t" "show runn full" |
   grep "input-directory" | awk -F ' ' '{print $9}'
   ```

   The resulting output may resemble:

   ```
   /data/collector/bulkstats_files/Delhi/
   /data/collector/bulkstats_file/Mumbai/
   /data/collector/edrflow
   /data/collector/edrhttp
   ```

   This command displays:

   - Input directory for bulkstats for both of the servers configured in the system
   - Input directory for edrflow as `/data/collector/edrflow`
   - Input directory for edrhttp as `/data/collector/edrhttp`

   b. Check for files in the input directories for each of the adaptors listed in the previous step (http, and flow and bulkstats):

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# _shell
   ```

```
[admin@host ~]# watch 'ls -lrt /data/collector/edrhttp'
[admin@host ~]# watch 'ls -lrt /data/collector/edrflow'
[admin@host ~]# watch 'ls -lrt /data/collector/bulkstats_
files/Delhi'
[admin@host ~]# watch 'ls -lrt /data/collector/bulkstats_
files/Mumbai'
```

c. If the files are not coming in these directories, please check the original directories where the ASR is sending the files. Those paths can be retrieved from `gateway.conf` at location `/data/-configs/gateway` on the collector:

```
{
"Name": "Delhi",
"Associated Region": "NORTH",
"IP": "10.10.2.1",
"file_pattern": "*_MURAL-edr_*_%MM%DD%YYYY%hh%mm%ss_*.*",
"output_dir": {
"http": "/data/collector/edrhttp",
"flow": "/data/collector/edrflow"
},
"Schema Version": "15",
"timezone": "Asia/Kolkata",
"input_dir": "/data/collector/edr1",
"Type": "HA",
"Location": "INDIA"
},
...
```

d. For each DC or gateway, check the input directory configured here. If files are being received but are not yet present in input directories for adaptors.

For example `/data/collector/edrhttp` or `/data/-collector/edrflow`.

Then check if the timezone converting process is running or not on that collector:

```
[admin@host gateway]# ps -ef  | grep collector
```

The resulting output may resemble:

```
admin    19379     1  0 Sep30 ?        00:01:02
/usr/bin/python /opt/etc/scripts/collector_file_
generator.py /data/configs/gateway/gateway.conf 10
```

e. Restart this process on master or standby Collector:

```
Kill -9 <pid>
```

f. Also, a new trap `DataResumeTrap` should be seen on the SNMP manager.

If data is received on the Collector nodes but is not getting written to the HDFS, then it may be possible that collector is dropping the records or files. Ensure that data is received from the ASR in the input directory as mentioned in the `gateway.conf` file for each DC and after successful time conversion it gets written in HDFS by the collector. You can check the dropped files or flow count and identify the reason for this drop in data by performing the steps in the following section:

1. Run the following commands to check total dropped flows count or dropped flow count in 5 minutes, 1 hour, and 1 day intervals since the last retstart of collector:

```
> en
# conf t
(config) # collector stats instance-id 1 adaptor-stats
<edrflow|edrhttp> dropped-flow
(config) # collector stats instance-id 1 adaptor-stats
<edrflow|edrhttp>  dropped-flow interval-type 5-min interval-
count 12
(config) # collector stats instance-id 1 adaptor-stats
<edrflow|edrhttp>  dropped-flow interval-type 1-hour interval-
count 12
```

2. Run the following commands to check if any files are being dropped due to incompatible filename format or due to old timestamp:

```
(config) # collector stats instance-id 1 adaptor-stats
<edrflow|edrhttp> num-files-dropped
(config) # collector stats instance-id 1 adaptor-stats
<edrflow|edrhttp> num-files-with-error
```

3. Perform the following steps to check the reason for dropped records:

   a. To check if the records are getting dropped because of the timestamp being too old:

   ```
   (config) # collector stats instance-id 1 adaptor-stats
   <edrflow|edrhttp> dropped-expired-binning-flow
   ```

   b. To check if the records are getting dropped because of the timestamp being in future:

   ```
   (config) # collector stats instance-id 1 adaptor-stats
   <edrflow|edrhttp> dropped-future-binning-flow
   ```

   c. To check if the records are getting dropped because of a mismatch in field count in header row and record in a file:

   ```
   (config) # collector stats instance-id 1 adaptor-stats
   <edrflow|edrhttp> dropped-flow-invalid-field-count
   ```

   d. To check if the records are getting dropped because of missing header or incorrect header line in files:

   ```
   (config) # collector stats instance-id 1 adaptor-stats
   edrflow dropped-flow-missing-header
   (config) # collector stats instance-id 1 adaptor-stats
   edrflow dropped-header
   ```

   e. To check if the records are getting dropped because of invalid field values or invalid field format. For example, values are missing for mandatory headers, and the type of value specified is not valid:

```
(config) # collector stats instance-id 1 adaptor-stats
<edrflow|edrhttp> dropped-flow-invalid-field-format
(config) # collector stats instance-id 1 adaptor-stats
<edrflow|edrhttp> dropped-flow-invalid-field-value
(config) # collector stats instance-id 1 adaptor-stats
<edrflow|edrhttp> dropped-flow-length-mismatch
```

Recovering Components

Recovering a Collector Node

This section describes how to recover when a Collector node fails in a high availability environment, that includes two nodes (Collector-GMS-1 and Collector-GMS-2) in the Collector node cluster.

When a node fails in this environment, regardless of whether it is a master or standby, the blade server of the failed node must be completely re-built.

- If the standby Collector node fails, re-build the standby Collector blade.
- If the master Collector node fails, the standby Collector node automatically becomes the master in a high availability environment. Build a new node to replace the failed one. The newly built Collector node joins the cluster as a standby Collector node.

If any of these steps fail, do not proceed. Instead, contact Technical Support to resolve the issue.

Building the Collector Node

1. Load the GMS UI with the xml used during initial configurations. Go to the “Server Details” tab and select the machine which is down.
2. Update the Mac_Address with regard to the new machine and save the xml.
3. SSH to the GMS server using the management IP address and activate the updated XML:

   ```
   > en
   # conf t
   (config)# gms config <file-name>.xml activate
   ```

   Note: The XML file is the same file used during GMS Configuration screens.
4. Log into the GMS node and manufacture the new blade using PXE boot via GMS:

```
> en
# conf t
(config)# image fetch scp://admin@<IP_LOCAL_
MACHINE>/<directory_path_on_local_machine>/<iso_image_name>
(config)# image mount <iso_image_name>
```

For example :

```
(config)# image fetch scp://admin@192.168.147.18/datamfgcd-
atlas3.7.rc1.iso
(config)# image mount mfgcd-atlas3.7.rc1.iso
```

Where `IP_Address` is the IP Address of the machine where iso image for the build has been downloaded.

The resulting output may resemble:

```
Copying linux...
Copying rootflop.img
Copying image.img...
```

5. Reboot the new blade from KVM manager and press F12 so as to boot them from Network Boot.
6. Once blades start booting from network, GMS will push the image on the blade using PXE boot and manufacture process will eventually start on blade.

   Wait for 30 minutes to get all the blades manufactured with image 3.5.
7. Verify that the same Logical Unit (LUN) (which has the same WWID) which was earlier associated with the faulty node blade is associated with the new blade.

Finding the WWID of LUNs

1. Log into the EMC
2. Click the **Storage** tab
3. Click **LUNs**

4. Highlight the destination LUN
5. Click **Properties**

The EMC Web UI shows the unique ID.

For the WWID, remove the separator ':' from the unique ID and prefix the complete unique ID with 3, for example:

```
360060160c7102f004887f4015d49e212
```

To compare to the LUN WWID with the LUNs that were set up during the original installation, refer to the XML configuration file on the GMS server. We are assuming here that LUN isn’t faulty and previous LUN has been assigned to the new node as well.

8. Apply any patches to the Collector nodes that have been released for this build. See the Release Notes for a list of patches and installation instructions.
9. To add the template for Service ID Report and DWI (Data Warehouse Integration), follow the following steps:
   - To add the following template on GSM machine for Service ID Report, execute the following commands:

     ```
     cd /config/gms/Profiles/Custom/
     ln -s /opt/deployment/GMS_
     Templates/oozie/App/serviceId/workflow_serviceId_apn_
     with_timeout.xml
     ```

   - To add the following template on GSM machine for DWI, execute the following commands:

     ```
     cd /config/gms/Profiles/Custom/
     ln -s /opt/deployment/GMS_
     Templates/oozie/App/DWH/workflow_dwh_with_timeout_
     jobs.xml
     ```

10. Establish an SSH connection to the GMS server using management IP address and start the installation on Collector node:

```
> en
# conf t
(config)# install appliance cluster cluster-name COLL-CLUS-GMS
node Collector-GMS-1
```

Where:

- The cluster-name `COL-CLUS-GMS` is the name of the GMS cluster. It will come up automatically when you press tab after typing cluster-name.
- `Collector-GMS-1` is the node that is being newly prepared.

11. Monitor the status of the installation status on the Collector blades by executing this command to show the percentage completed:

```
(config)# install appliance show installation-status cluster
COL-CLUS-GMS node Collector-GMS-1
```

12. Wait for the node successfully installed message to appear before proceeding.

Configuring the Collector Nodes

This section describes configuring the Collector node and displaying the configurations as they are applied:

If you receive an error message, enter `YES` to the question of whether to revert the configuration, and contact Cisco Technical Support.

1. Log into the master Collector node (NameNode) and go to the oozie subshell:

```
> en
# conf t
(config)# pmx subshell subshell-name
```

2. Identify the start time for the dataset, as the system uses this in the `job-start` script.

```
 pm extension (oozie)> show config dataset atlas_edrflow_1
```

The resulting output may resemble:

```
    Attributes:
    -----------
        doneFile     : _DONE
        endOffset    : 1
        frequency    : 5
        outputOffset : 0
        path         :
/data/collector/1/output/edrflow/%Y/%M/%D/%H/%mi
        pathType     : hdfs
        startOffset  : 12
        startTime    : 2012-11-28T07:50Z
```

Make a note of the value in the **startTime** field. In the preceding output, it is **2012-11-28T07:50Z**.

3. If there is an SSH key for the newly manufactured Collector node in the known hosts file of the GMS, remove it.

   Log into the GMS and go to the `_shell` prompt:

```
> en
# _shell
# vi /var/home/root/.ssh/known_hosts
```

4. Log into the GMS and change the directory:

```
 admin@host> en
admin@host# _shell
admin@host# cd /opt/deployment/MURAL_setStartTime
```

5. Run the script:

```
 admin@host# ./setOozieTime --dataStartTime data-start-time --
node management-ip --password admin@123 --verbose
```

   Where:

- *data-start-time* is obtained through step 1.
- *management-ip* is the IP address of the new restored collector node.
- The password *admin@123* is for an admin user.

Site-Specific Configuration

Perform the following steps to set up the applications as per the site requirements:

1. If Anomaly Detection feature is enabled, run the following command on restored Collector (NameNode):

```
# pmx subshell oozie set dataset anomalyMonthly attribute path
/data/output/AnomalyAggDay/%Y/%M/%D/%H
```

2. If Tethering feature is enabled, run the following command on the restored Collector (NameNode):

```
# pmx subshell oozie set dataset edr_
TetheringSubscribersReport attribute frequency 60
# cli -m
config# write memory
```

Pushing IBs to the New Collector

Insta node processes must be running to execute the following commands.

If there is an SSH key for the newly manufactured Collector node in the known hosts file of the NameNode, remove it:

Log into the GMS and go to the `_shell` prompt:

```
> en
# _shell
# vi /var/home/root/.ssh/known_hosts
```

1. Run add gateway command for the restored collector IP so that configurations get updated on this node.

2. To check the values that need to be provided in add gateway command,

check the values from this file for each gateway on the master Collector (NameNode):

```
# cat /data/configs/gateway/gateway.conf
```

For example,

```
[
 {
  "Location": "USA",
  "IP": "10.10.2.1",
  "Name": "GMPLAB1",
  "input_dir": "/data/collector/GMPLAB1/edr/",
  "Dc": "GMPLAB1",
  "file_pattern": "*_*-edr_*_*_%MM%DD%YYYY%hh%mm%ss_*.gz",
  "output_dir": {
   "flow": "/data/collector/edrflow",
   "http": "/data/collector/edrhttp",
   "asn": "/data/collector/edrAsn"
},
"Schema Version": "0",
"timezone": "US/Pacific",
"Type": "HA",
"Associated Region": "EAST"
}
]
```

```
# pmx subshell aggregation_center
pm extension (aggregation center)> add gateway name GMPLAB1 dc
GMPLAB1 region EAST location USA ip 10.10.2.1 timezone
US/Pacific edr-filename-pattern *_*-edr_*_
*_%MM%DD%YYYY%hh%mm%ss_*.gz edr-collector-filename-pattern *_
*_*_*_%MM%DD%YYYY%hh%mm%ss_*.gz collector-ip 10.20.20.240 edr-
file-path /GMPLAB1/edr1/   type HA
```

The sample output may resemble as follows:

```
Gateway/DC already exists with following configuration

*************************Gatewa
y*******************************
Name: GMPLAB1
Dc: GMPLAB1
Associated Region: EAST1
Location: USA
IP: 10.10.2.1
Timezone: US/PacificFlow-EDR/Http-EDR/Asn-EDR
Filename Pattern:
Type: HAFlow-EDR/Http-EDR
Filename Path:

*************************************************************
**
Do you want to add/update the gateway/DC?(Yes/No): Yes
pm extension (aggregation center)> show collector IPs
```

3. Set the BulkStats timezone to UTC in the `gateway.conf` file for each gateway. This is because ASR internally changes the time zone to GMT for the BulkStats file. The gateway.conf file for each BulkStats source is available at `/data/configs/gateway/gateway.conf "timezone": "UTC"`.

```
pm extension (aggregation center)> push gateway configuration
```

4. Add collector IPs on the recovered Collector node:

```
# pmx subshell aggregation_center
pm extension (aggregation center)> show  collector IPs
pm extension (aggregation center)> set collector IPs
0.20.20.240,10.20.20.236,10.20.20.237,10.20.20.239
```

Provide a comma separated list of the data network IPs of all collectors in the setup.

5. To update the lastet IBs, execute the following commands:

```
>en
#_shell
#pmx subshell aggregation_center
(aggregation_center) # update all ibs from image
(aggregation_center) # generate urlcat ib from image
(aggregation_center) # show urlcat version
(aggregation_center) # generate all ibs
(aggregation_center) # push ib serializedUrlCatObj
(aggregation_center) # push all ibs
(aggregation_center) # quit
```

For Bulkstats

1. Go to the pmx bulkstats subshell:

```
> en
# conf t
(config)# pmx subshell subshell-name
```

2. Generate and push all IBs:

```
 pm extension (bulkstats)> generate all ibs
pm extension (bulkstats)> push all ibs
```

3. Log into the restored Collector node and go to the pmx bulkstats subshell as in step 2.

   IBs are fetched automatically on the standby node.

For EDR

1. Go to the pmx `aggregation_center` subshell.

```
> en
# conf t
(config)# pmx subshell subshell-name
```

2. Generate and push all IBs.

```
pm extension (subshell-name)> generate all ibs
pm extension (subshell-name)> push all ibs
```

IBs are fetched automatically on the standby node.

Using the Anomaly Subshell

This section describes the use of Anomaly subshell for generating and pushing the IBs. Perform these tasks only if Anomaly application is enabled.

The following commands can be used to generate and push the IBs:

```
>en
#conf t
(config)# pmx subshell anomaly
pm extension (anomaly)> update all ibs
pm extension (anomaly)> generate all ibs
pm extension (anomaly)> push all ibs
pm extension (anomaly)> quit
```

Applying Performance Related Changes

This section describes some manual changes and configurations that are not part of the installation script but that are specific to the product installation.

1. Log into the newly configured Collector node and execute:

```
 host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# internal set modify -
/tps/process/hadoop/attribute/
mapred.min.split.size/value value string 268435456
host [cluster : master|standby](config)# internal set create -
/tps/process/hadoop/attribute/
mapred.tasktracker.map.tasks.maximum/value value string 10
host [cluster : master|standby](config)# write memory
```

2. Use SCP to transfer these files from the GMS to the new Collector node:

- `/data/work/serverFile_tethering`
- `/data/work/serverFile_uncatReports`
- `/data/work/serverfile_Rulebase` (required only if Rulesbase reporting feature is enabled for the site)

3. Create a work directory on the new Collector node:

```
 host [cluster : master|standby](config)# _shellhost [cluster
: master|standby]# cd /data
host [cluster : master|standby]# mkdir workhost [cluster :
master|standby]# cd work
```

4. Login into the master Collector node and go to `_shell`:

```
 (config)# _shell
```

5. Change your location to the newly created work directory:

```
 # cd /data/work
```

6. Copy files from master Collector to the new Collector server:

```
# scp serverFile_uncatReports admin@<Restored Mgmt
IP>:/data/work/serverFile_uncatReports
# scp serverFile_tethering admin@<Restored Mgmt
IP>:/data/work/serverFile_tethering
# scp serverFile_tethering_subscribers_report admin@<Restored
Mgmt IP>:/data/work/serverFile_tethering_subscribers_report
# scp serverfile_ServiceId admin@<Restored_Mgmt
IP>:/data/work/serverfile_ServiceId ( For service-id report
feature)
# scp serverFile_Tethering_ApnDc_report admin@<Restored_Mgmt
IP>:/data/work/serverFile_Tethering_ApnDc_report
```

Note: The last two tethering related exists only if tethering application is enabled at the site and these files were created at the time of installation.

7. To enable the service ID in the flow file, execute the following commands:

```
>en
#_shell
#python /opt/etc/oozie/SolutionConfigs/EnableServiceId.py yes
(to enable service id in flow feed file)cli -m config(config)
# internal set modify -
/tps/process/oozie/jobs/TransferJob/actions/TransferOozieActio
n/attribute/destNamenode/value value string (Dest cluster
name)
```

Note: WebHDFS must be enabled on both the clusters.Each DataNode/NameNode must have the mapping of the source and destination DataNode/NameNode in the `hosts` file of `/etc` folder.

For Example:

```
192.168.194.121      machine-194-121 (destination NameNode)
192.168.194.122      machine-194-122 (destination DataNode)
192.168.192.148      machine-192-148 (source NameNode)
192.168.192.149      machine-192-149 (destination DataNode)
192.168.192.151      machine-192-151 (destination DataNode)
192.168.192.152      machine-192-152 (destination DataNode)
192.168.192.41       MUR-COL-CLUS(destination cluster name)
```

8. To modify the IP filter rules, perform the following steps:
   a. To check the input IP rules, run the following commands:

```
echo "show ip filter configured" | cli -m config | grep
DROP | grep 50070
71  DROP    tcp     all   all         dpt 50070
echo "show ip filter configured" | cli -m config | grep
DROP | grep 50075
75  DROP    tcp     all     all       dpt 50075
```

   b. To remove the input IP rules, execute the following commands:

```
cli -m config -t "no ip filter chain INPUT rule 75"
cli -m config -t "no ip filter chain INPUT rule 71”
```

9. Configure the start time of job/dataset, ensure that the start time is same as configured on other collector node using commands mentioned in section Stopping and Restarting Specific Jobs.

Start Processes on the Recovered Collector Node

1. Stop the gms_server process on the master GMS node:

```
> en
# conf t
# pm process gms_server terminate
```

2. Re-add this Collector as journal node from master Name Node (Collector Node):

```
> en
# conf t
(config) # pmx subshell hadoop_yarn
pm extension (hadoop_yarn)> remove journalnode TMO-COLL-1
Stopping JournalNode on Host: TMO-COLL-1
Executing: ssh -q -o ConnectTimeout=10 root@TMO-COLL-1
"/opt/hadoop/sbin/hadoop-daemon.sh stop journalnode"
```

Here, "TMO-COLL-1” is the hostname of the recovered Collector.

Wait for 5 minutes before running the following command:

```
pm extension (hadoop_yarn)> add journalnode TMO-COLL-1
Setting up node TMO-COLL-2
Transferring hadoop config to node: TMO-COLL-2
Transferring dfs.journalnode.edits.dir to the new server.
Executing Command scp -rqp root@TMO-COLL-
2:/data/yarn/journalnode/ root@TMO-COLL-
```

```
1:/data/yarn/journalnode >/dev/null
Access restricted to authorised users only, welcome to MURAL
3.5
Connection to TMO-COLL-2 closed.
Setting up node TMO-COLL-1
Transferring hadoop config to node: TMO-COLL-1
Starting JournalNode on Host: TMO-COLL-1
Executing: ssh -q -o ConnectTimeout=10 root@TMO-COLL-1
"/opt/hadoop/sbin/hadoop-daemon.sh start journalnode"
Executing cmd: ssh -q -o ConnectTimeout=10 root@TMO-COLL-2
'USER=admin /opt/hadoop/sbin/hadoop-daemon.sh stop
namenode'stopping namenode
Executing cmd: ssh -q -o ConnectTimeout=10 root@TMO-COLL-2
'USER=admin /opt/hadoop/sbin/hadoop-daemon.sh start
namenode'starting namenode, logging to
/opt/hadoop/logs/hadoop-admin-namenode-TMO-COLL-2.out
Executing cmd: ssh -q -o ConnectTimeout=10 root@TMO-COLL-1
'USER=admin /opt/hadoop/sbin/hadoop-daemon.sh stop
namenode'stopping namenode
Executing cmd: ssh -q -o ConnectTimeout=10 root@TMO-COLL-1
'USER=admin /opt/hadoop/sbin/hadoop-daemon.sh start
namenode'starting namenode, logging to
/opt/hadoop/logs/hadoop-admin-namenode-TMO-COLL-1.out
Safemode is ON
NameNode process is running. SAFE mode status: 1
In WaitForNameNode, Node is master or standalone
Executed cmd : /opt/hadoop/bin/hdfs haadmin -getServiceState
TMO-COLL-1, Output : standby
Executed cmd : /opt/hadoop/bin/hdfs haadmin -getServiceState
TMO-COLL-1, Output : standby
Setting namenode to Active
Executing: /opt/hadoop/bin/hdfs haadmin -getServiceState TMO-
COLL-2standby
```

```
Executing: /opt/hadoop/bin/hdfs haadmin -failover --
forceactive TMO-COLL-2 TMO-COLL-1Failover from TMO-COLL-2 to
TMO-COLL-1 successful
Namenode is in safemode, Sleeping for SAFEMODEWAIT(20s) time
Starting Datanodes
Datanode is already running on: 10.20.20.232
Datanode is already running on: 10.20.20.233
configured DNs:2 actual DNs:1
Safemode is OFF
NameNode process is running. SAFE mode status: 0
In WaitForNameNode, Node is master or standalone
Executed cmd : /opt/hadoop/bin/hdfs haadmin -getServiceState
TMO-COLL-1, Output : active
Datanode is running on: 10.20.20.232
Atleat one of the datanode is up, exiting WaitForNameNode.
configured DNs:2
actual DNs:1
Executed cmd : /opt/hadoop/bin/hdfs haadmin -getServiceState
TMO-COLL-1, Output : active
pm extension (hadoop_yarn)> quit
```

3. Start the gms_server process on the master GMS node:

```
> en
# conf t
# pm process gms_server restart
```

4. Restart the processes to add the recovered Collector to the HDFS cluster:

```
# pm process tps restart
```

Note: Wait for 10 min to let the processes initialize the system.

```
(config) # _shell
# ps -ef | grep java | grep Dproc | grep -v Dproc_dfs | awk '
{print $9}'
```

```
-Dproc_namenode
-Dproc_namenode
-Dproc_datanode
-Dproc_secondarynamenode
-Dproc_journalnode
```

5. Start the Collector process on the new or recovered Collector node:

```
# cli -m config
(config) # pm process collector restart
(config) # pm process pgsql restart
(config) # _shell
# cli -t "en" "conf t" "show pm process collector" | grep
status
Current status:  running
```

Setting Up a New User for the ASR in the Collectors

To set up a new user for the ASR in the Collectors:

1. Log in to the master Collector node:

```
 host [cluster : master]# en
host [cluster : master]> conf t
```

2. Create the user:

```
 host [cluster : master](config)> username user-id password
password
host [cluster : master](config)> write memory
```

Where the username and password should be the same ones configured for EDR and bulkstats files transfer on the ASR.

Recovering a Compute Node

This section explains how to add a new Compute node should an existing one fail. You must add the new Compute node to the existing cluster.

If any of these steps fail, do not proceed. Instead, contact Technical Support to resolve the issue.

To verify that a Compute node is down or inaccessible:

1. Log into the master Name (Collector) node using an SSH connection and check the status of the nodes:

```
host [cluster : master]> en
host [cluster : master]# _shell
host [cluster : master]# hdfs dfsadmin -report
```

   The resulting output may resemble:

```
  WARNING: org.apache.hadoop.metrics.jvm.EventCounter is
deprecated.
 Please use org.apache.hadoop.log.metrics.EventCounter in all
the
 log4j.properties files.
  Configured Capacity: 1082268307456 (1007.94 GB)
  Present Capacity: 1027082702770 (956.55 GB)
  DFS Remaining: 645792276480 (601.44 GB)
  DFS Used: 381290426290 (355.1 GB)
  DFS Used%: 37.12%
  Under replicated blocks: 31032
  Blocks with corrupt replicas: 0
  Missing blocks: 0

  -------------------------------------------------

  Datanodes available: 3 (3 total, 1 dead)
```

```
Name: 10.10.2.13:50010
Decommission Status : Normal
Configured Capacity: 1082254041088 (1007.93 GB)
DFS Used: 335519627680 (312.48 GB)
Non DFS Used: 55049049696 (51.27 GB)
DFS Remaining: 691685363712(644.18 GB)
DFS Used%: 31%
DFS Remaining%: 63.91%
Last contact: Mon Dec 17 09:38:00 GMT 2012

Name: 10.10.2.14:50010
Decommission Status : Normal
Configured Capacity: 1082254041088 (1007.93 GB)
DFS Used: 489931977134 (456.28 GB)
Non DFS Used: 54972370514 (51.2 GB)
DFS Remaining: 537349693440(500.45 GB)
DFS Used%: 45.27%
DFS Remaining%: 49.65%
Last contact: Mon Dec 17 09:38:00 GMT 2012



Name: 10.10.2.18:50010
 Decommission Status : Normal
 Configured Capacity: 0 (0 KB)
 DFS Used: 0 (0 KB) Non DFS Used: 0 (0 KB)
 DFS Remaining: 0(0 KB)
 DFS Used%: 100%
 DFS Remaining%: 0%   Last contact: Mon Dec 17 05:38:00 GMT
2012
```

A Compute node that is down or is not accessible shows values of zero for:

- DFS Used
- Non DFS Used
- DFS Remaining

Building a New Compute Node

1. Load the GMS UI with the xml used during initial configurations. Go to the **Server Details** tab and select the machine which is down.

2. Update the `Mac_Address` with regard to the new machine and save the xml.

3. SSH to the GMS using the management IP and start the installation on the nodes that failed:

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   host [cluster : master|standby](config)# gms config mural.xml
   activate
   ```

   Note: The `mural.xml` file is the same file used during GMS Configuration screens.

4. Manufacture all the blades using PXE boot via GMS. Log into the GMS and run:

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   host [cluster : master|standby](config)# image fetch
   scp://admin@ip-local-machine/
   directory-path-on-local-machine/iso-image-name
   host [cluster : master|standby](config)# image mount iso-
   image-name
   ```

   Where `ip-local-machine` is the IP address of the machine where the iso image for the build has been downloaded.

   For example:

   ```
   host [cluster : master|standby](config)# image fetch
   scp://admin@192.168.147.18/data/mfgcd-atlas3.7.rc1.iso
   host [cluster : master|standby](config)# image mount mfgcd-
   atlas3.7.rc1.iso
   ```

   The resulting output may resemble:

```
Copying linux...
Copying rootflop.img...
Copying image.img...
```

5. Reboot the blade from KVM Manager and press F12 to boot them from network boot. Once blades start booting from the network, GMS will push the image on the blade using PXE boot and manufacture process will eventually start on blade.

   Wait 20 minutes to get the blade manufactured with image 3.5.

6. Verify that the same Logical Unit (LUN) (which has the same WWID) which was earlier associated with the faulty node blade is associated with the new blade.

   #### Finding the WWID of LUNs

   1. Log into the EMC
   2. Click the **Storage** tab
   3. Click **LUNs**
   4. Highlight the destination LUN
   5. Click **Properties**

   The EMC Web UI shows the unique ID.

   For the WWID, remove the separator ':' from the unique ID and prefix the complete unique ID with 3, for example:

   ```
   360060160c7102f004887f4015d49e212
   ```

   To compare to the LUN WWID with the LUNs that were set up during the original installation, refer to the mural.xml configuration file on the GMS. We are assuming here that LUN isn’t faulty and previous LUN has been assigned to the new node as well.

Configuring the New Compute Node

1. Apply patches released with the new build to the Compute node. For more information, refer to the *Release Notes* for a list of patches and installation

instructions.

2. To configure the new node with the new image, log in to the GMS server using an SSH connection and execute the following command:

```
admin@host > en admin@host # conf t admin@host (config) #
install appliance cluster cluster-name ComputeNode-CLUS node
Compute-GMS-1
```

Where,

- `ComputeNode-CLUS` is the name of the cluster given to Compute node as per the XML. This name is read from the xml specified (mur-al.xml) and should come automatically when you press tab.
- `Compute-GMS-1` is the name of the Compute node specified in the the XML file. This name is read from the XML file and should come up automatically when you press tab.

3. Monitor the installation status on the server by executing this command to show the percentage completed:

```
admin@host (config) # install appliance show installation-
status cluster ComputeNode-CLUS node Compute-GMS-1
```

When the installation is completed, the "Node successfully installed." message is displayed.

Adding New Node to Cluster

1. Log in to the master Collector node and run:

```
host [cluster : master]> en
host [cluster : master]# _shell
host [cluster : master]# echo "" > /var/home/root/.ssh/known_
hosts
```

2. Go to the CLI and then to the pmx hadoop subshell:

```
host [cluster : master]# cli -m config
host [cluster : master](config)# pmx
pm extension> subshell hadoop
```

3. Log in to the standby Collector node and execute:

```
host [cluster : master]> en
host [cluster : master]# _shell
host [cluster : master]# echo "" > /var/home/root/.ssh/known_
hosts
```

4. Check if the DN was also behaving as a journal node. If it was, perform this step. Otherwise, skip this step and proceed to the next step.

   a. Check the journal nodes:

   ```
   host [cluster: master] > en
   host [cluster: master] #> conf t
   (config) # pmx subshell hadoop_yarn
   (config) # show config
   ```

   Check the Nodes displayed against "journalnodes:". If the hostname of the Data node that is being recovered is listed, then perform the steps b and c. Otherwise, proceed to step 5.

   b. Stop the gms_server process on the Master GMS node:

   ```
   > en
   # conf t
   # pm process gms_server terminate
   ```

   c. Add the DN as journal node again:

   ```
   pm extension (hadoop_yarn)> remove journalnode TMO-DN-1

   Stopping JournalNode on Host: TMO-DN-1
   ```

```
Executing: ssh -q -o ConnectTimeout=10 root@TMO-DN-1
"/opt/hadoop/sbin/hadoop-daemon.sh stop journalnode"
```

Note: Wait for 5 minutes before running the following command.

```
pm extension (hadoop_yarn)> add journalnode TMO-DN-1
Setting up node TMO-COLL-2
Transferring hadoop config to node: TMO-COLL-2
Transferring dfs.journalnode.edits.dir to the new server.
Executing Command scp -rqp root@TMO-COLL-
2:/data/yarn/journalnode/ root@TMO-DN-
1:/data/yarn/journalnode >/dev/null

Access restricted to authorised users only, welcome to
MURAL 3.5
Connection to TMO-COLL-2 closed.
Setting up node TMO-DN-1
Transferring hadoop config to node: TMO-DN-1
DN profile configured for slave: TMO-DN-1 is compute_DN-
CLUS
Transferring datanode config to node: TMO-DN-1
Starting JournalNode on Host: TMO-DN-1
Executing: ssh -q -o ConnectTimeout=10 root@TMO-DN-1
"/opt/hadoop/sbin/hadoop-daemon.sh start journalnode"
Executing cmd: ssh -q -o ConnectTimeout=10 root@TMO-COLL-
2 'USER=admin /opt/hadoop/sbin/hadoop-daemon.sh stop
namenode'
stopping namenode
Executing cmd: ssh -q -o ConnectTimeout=10 root@TMO-COLL-
2 'USER=admin /opt/hadoop/sbin/hadoop-daemon.sh start
namenode'
starting namenode, logging to /opt/hadoop/logs/hadoop-
admin-namenode-TMO-COLL-2.out
Executing cmd: ssh -q -o ConnectTimeout=10 root@TMO-COLL-
```

```
1 'USER=admin /opt/hadoop/sbin/hadoop-daemon.sh stop
namenode'
no namenode to stop
Executing cmd: ssh -q -o ConnectTimeout=10 root@TMO-COLL-
1 'USER=admin /opt/hadoop/sbin/hadoop-daemon.sh start
namenode'
starting namenode, logging to /opt/hadoop/logs/hadoop-
admin-namenode-TMO-COLL-1.out
Safemode is ON
NameNode process is running. SAFE mode status: 1
In WaitForNameNode, Node is master or standalone
Executed cmd : /opt/hadoop/bin/hdfs haadmin -
getServiceState TMO-COLL-1, Output : active
Namenode is in safemode, Sleeping for SAFEMODEWAIT(20s)
time
Starting Datanodes
Datanode is already running on: 10.20.20.232
Datanode is already running on: 10.20.20.233
Sleeping for 60 seconds before forcing to exit safemode
Safemode is OFF
Executed cmd : /opt/hadoop/bin/hdfs haadmin -
getServiceState TMO-COLL-1, Output : active
pm extension (hadoop_yarn)> quit
```

d. Start the gms_server process on the master GMS node:

```
> en
# conf t
# pm process gms_server restart
```

5. Restart the processes to add the recovered DN to the HDFS cluster. Log in to the master Name node or Collector node and run the following command:

```
host [cluster: master] > pm process tps restart
```

Wait for 10 minutes for all the processes to come up.

6. Verify that the Compute node was added to the cluster:

```
host [cluster : master]# hdfs dfsadmin -report
```

Where the output from the above command resembles the output from step 1, but with a key difference: the Compute node, earlier shown as dead, returns with a status of `Normal`.

Recovering an Insta Node

This section describes how to add a new Insta (also called Caching Compute) node if any of the existing Insta nodes fail.

If any of these steps fail, do not proceed. Instead, contact Technical Support to resolve the issue.

Checking Status of Insta Nodes

If a node fails, regardless of whether the node is functioning as a master or standby, the blade server of the failed node must be completely re-built.

- If the standby Insta node fails, re-build the standby Insta blade.
- If the master Insta node fails, the standby Insta node automatically becomes the master in a high availability environment. Build a new node to replace the failed one. The newly built Insta node joins the cluster as a standby Insta node.

Building a New Insta Node

1. Load the GMS UI with the xml used during initial configurations. Go to the **Server Details** tab and select the machine which is down.
2. Update the `Mac_Address` with regard to the new machine and save the xml.
3. SSH to the GMS using the management IP and start the installation on the nodes that failed:

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   host [cluster : master|standby](config)# gms config mural.xml
   activate
   ```

   Note: The `mural.xml` file is the same file used during GMS Configuration screens.
4. Manufacture all the blades using PXE boot via GMS. Log into the GMS and run:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# image fetch
scp://admin@ip-local-machine/
directory-path-on-local-machine/iso-image-name
host [cluster : master|standby](config)# image mount iso-
image-name
```

Where `ip-local-machine` is the IP address of the machine where the iso image for the build has been downloaded.

For example:

```
host [cluster : master|standby](config)# image fetch
scp://admin@192.168.147.18/data/mfgcd-atlas3.7.rc1.iso
host [cluster : master|standby](config)# image mount mfgcd-
atlas3.7.rc1.iso
```

The resulting output may resemble:

```
Copying linux...
Copying rootflop.img...
Copying image.img...
```

5. Reboot the blade from KVM Manager and press F12 to boot them from network boot. Once blades start booting from the network, GMS will push the image on the blade using PXE boot and manufacture process will eventually start on blade.

   Wait 20 minutes to get the blade manufactured with image 3.5.

6. Verify that the same Logical Unit (LUN) (which has the same WWID) which was earlier associated with the faulty node blade is associated with the new blade.

Finding the WWID of LUNs

1. Log into the EMC
2. Click the **Storage** tab

3. Click **LUNs**
4. Highlight the destination LUN
5. Click **Properties**

The EMC Web UI shows the unique ID.

For the WWID, remove the separator ':' from the unique ID and prefix the complete unique ID with 3, for example:

```
360060160c7102f004887f4015d49e212
```

To compare to the LUN WWID with the LUNs that were set up during the original installation, refer to the mural.xml configuration file on the GMS. We are assuming here that LUN isn't faulty and previous LUN has been assigned to the new node as well.

7. Stop all other modules interacting with the Insta node:

   a. Log into master Collector blade of this setup, go to the oozie subshell, and stop all jobs:

   ```
   host [cluster: master]> en
   host [cluster: master]# conf t
   host [cluster: master](config)# pmx subshell subshell-
   name
   pm extension (subshell-name)> stop jobname all
   pm extension (subshell-name)> show coordinator RUNNING
   jobs
   ```

   Note: This command will take approximately 5-10 minutes to stop all jobs.

   b. Log into the master UI node and stop the Rubix process:

   ```
   # conf t
   (config) # pm process rubix terminate
   ```

   c. Repeat this on the standby UI node.

   d. Stop insta processes on the master Insta node:

```
> en
# conf t
# pm process insta terminate
# _shell
# cc shutdownsystem y
```

Configuring the New Insta Node

To configure either the original master or the standby node:

1. Apply all patches that have been released for this build to the recovered iNSTA node. See the *Release Notes* for a list of patches and installation instructions.
2. Establish an SSH connection to the GMS using management IP and start the installation on the standby UI node that failed:

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   ```

3. Start the installation on the Insta node:

   ```
                     host [cluster : master|standby](config)#
   install appliance cluster cluster-name  INSTA-CLUS
   ```

   Where:

   - **INSTA-CLUS** is the name of the cluster used while configuring GMS. It comes up automatically when you press tab after typing `cluster-name`.

4. Monitor the installation status:

   ```
   host [cluster : master|standby](config)# install appliance
   show installation-status cluster INSTA-CLUS
   ```

   Infinidb takes about 45-60 minutes to install. Wait for the message `Node successfully installed` before proceeding.

Log into each of the Insta nodes and monitor the status of the database using the below command:

```
host [cluster : master|standby]# insta infinidb get-status-
info
```

Note: Wait until it starts showing the install status as `INSTALLED`, the adaptor as `RUNNING`, and both the instances also as `RUNNING`. This may take up to 45 minutes. The resulting output may resemble:

```
Install status : INSTALLED
Total instances configured: 2
Insta instance 0 service status : RUNNING
Insta instance 1 service status : RUNNING
Infinidb Adaptor status : Adaptor Running
```

Note: The output may display up to three instances if BulkStats and HET applications are enabled.

5. Log in to each of the Insta nodes and run the following commands to ensure that the Insta process and pgsql process are up and running:

   a. Run the following command to verify the status of Insta process:

   ```
   # cli -t "en" "conf t" "show pm process insta" | grep
   "Current status"
   ```

   This command must return status as `running`.

   b. Run the following command to verify the status of PGSQL process:

   ```
   # cli -t "en" "conf t" "show pm process pgsqld" | grep
   "Current status"
   ```

   This command must return status as `running`.

If any of these steps fail, do not proceed. Instead, contact Technical Support to resolve the issue.

Re-Starting Jobs / tomcats on Collector / Rubix Nodes

1. To restart oozie jobs:

   a. Log into the master Collector node and go to the pmx oozie subshell:

   ```
   > en
   # conf t
   (config)# pmx subshell subshell-name
   ```

   b. Start all jobs:

   ```
   pm extension (oozie)> run job all
   ```

2. Log into the master Rubix node and restart the Rubix process:

   ```
   host [cluster : master]> en
   host [cluster : master]# conf t
   host [cluster : master](config)# pm process rubix restart
   ```

3. Repeat the above steps on the standby Rubix node.

Recovering a Failed UI Node: Scenario 1

If the Rubix node in a cluster goes down due to a hardware failure, you must build and configure a new node as described in the subsequent sections.

When a node fails in this environment, regardless of whether it is a master or standby, the blade server of the failed node must be completely re-built.

- If the standby Rubix node fails, re-build the standby Rubix blade.
- If the master Rubix node fails, the standby Rubix node automatically becomes the master in a high availability environment. Build a new node to replace the failed one. The newly built Rubix node joins the cluster as a standby Rubix node.

If any of these steps fail, do not proceed. Instead, contact Technical Support to resolve the issue.

Building a New UI Node

1. Load the GMS UI with the xml used during initial configurations. Go to the **Server Details** tab and select the machine which is down.
2. Update the `Mac_Address` with regard to the new machine and save the xml.
3. SSH to the GMS using the management IP and start the installation on the nodes that failed:

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   host [cluster : master|standby](config)# gms config mural.xml
   activate
   ```

   Note: The `mural.xml` file is the same file used during GMS Configuration screens.
4. Manufacture all the blades using PXE boot via GMS. Log into the GMS and run:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# image fetch
scp://admin@ip-local-machine/
directory-path-on-local-machine/iso-image-name
host [cluster : master|standby](config)# image mount iso-
image-name
```

Where `ip-local-machine` is the IP address of the machine where the iso image for the build has been downloaded.

For example:

```
host [cluster : master|standby](config)# image fetch
scp://admin@192.168.147.18/data/mfgcd-atlas3.7.rc1.iso
host [cluster : master|standby](config)# image mount mfgcd-
atlas3.7.rc1.iso
```

The resulting output may resemble:

```
Copying linux...
Copying rootflop.img...
Copying image.img...
```

5. Reboot the blade from KVM Manager and press F12 to boot them from network boot. Once blades start booting from the network, GMS will push the image on the blade using PXE boot and manufacture process will eventually start on blade.

   Wait 20 minutes to get the blade manufactured with image 3.5.

6. Verify that the same Logical Unit (LUN) (which has the same WWID) which was earlier associated with the faulty node blade is associated with the new blade.

Finding the WWID of LUNs

1. Log into the EMC
2. Click the **Storage** tab

3. Click **LUNs**
4. Highlight the destination LUN
5. Click **Properties**

The EMC Web UI shows the unique ID.

For the WWID, remove the separator ':' from the unique ID and prefix the complete unique ID with 3, for example:

```
360060160c7102f004887f4015d49e212
```

To compare to the LUN WWID with the LUNs that were set up during the original installation, refer to the mural.xml configuration file on the GMS. We are assuming here that LUN isn’t faulty and previous LUN has been assigned to the new node as well.

Configuring the New UI Node

To configure either the original master or the standby node:

1. Apply all patches that have been released for this build to the recovered iNSTA node. See the *Release Notes* for a list of patches and installation instructions.
2. Establish an SSH connection to the GMS using management IP and start the installation on the standby UI node that failed:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
```

3. Start the installation on the UI node:

```
                    host [cluster : master|standby](config)#
install appliance cluster cluster-name  Rubix-Cluster-Name
restored-UI-node
```

Where:

  - **Rubix-Cluster-Name** is the name of the cluster used while configuring GMS. It comes up automatically when you press tab after typing `cluster-name`.
  - **restored-UI-node** is the hostname of the UI node that failed.

4. Monitor the installation status:

   ```
   host [cluster : master|standby](config)# install appliance
   show installation-status cluster Rubix-Cluster-Name
   restored-UI-node
   ```

   Wait for the message `Node successfully installed` before proceeding.

   Note: The output may display up to three instances if BulkStats and HET applications are enabled.

Pushing IBs to the New Node

1. Before pushing IBs to the restored node, remove any SSH keys for the newly manufactured rubix node in the known hosts file of the master and standby Collector nodes:

   Log into the GMS and go to the `_shell` prompt:

   ```
   > en
   # _shell
   # vi /var/home/root/.ssh/known_hosts
   ```

2. Log into the master Collector node and go to the pmx bulkstats subshell:

   ```
   > en
   # conf t
   (config)# pmx subshell subshell-name
   ```

3. Push all IBs:

   ```
   pm extension (bulkstats)> push all ibs mgmt-ip-of-restored-
   rubix-node
   ```

4. Change subshell to `anomaly`:

```
pm extension (bulkstats)> quit
pm extension> subshell anomaly
```

5. Push all IBs:

```
pm extension (subshell)> push all ibs
pm extension (subshell)> quit
```

6. Change subshell to `aggregation_center`:

```
pm extension> subshell aggregation_center
```

7. Push all IBs:

```
pm extension (aggregation_center)> push all ibs mgmt-ip-of-
restored-rubix-node
pm extension (aggregation_center)> quit
pm extension> quit
```

Starting Processes on Standby UI Node

1. Log into the standby UI node and restart the Rubix process:

```
host [cluster : standby]> en
host [cluster : standby]# conf t
host [cluster : standby](config)# pm process rubix restart
```

2. Start tomcats:

```
host [cluster : master|standby](config)# rubix modify-app
process-name enable
host [cluster : master|standby](config)# rubix modify-app
process-name modify-instance 1 enable
```

Where `process-name` is replaced with:

- atlas
- reportAtlas
- bulkstats
- rge
- httperror

- ruleEngine
- launcher

Note: Wait for two minutes between starting each process.

Recovering from Failure of Multiple Nodes of the Same Type

The following sections describe how to recover if multiple nodes of the same type fail, including Collector, Insta, and Rubix nodes.

When two or more nodes fail, the blade servers of the failed nodes must be completely re-built.

Manufacturing the Nodes

1. If you are restoring failed Collector nodes, log into the ASR and stop the EDR and bulkstats feeds.
2. Build the new nodes, per the section of the same name:
   - "Recovering a Collector Node" on page 227
   - "Recovering a Compute Node" on page 243
   - "Recovering an Insta Node " on page 252
   - "Recovering a Failed UI Node: Scenario 1" on page 258
3. If you are restoring failed Collector nodes, data on the Compute nodes was also wiped out. This is due to the fact that the cluster was running on the Collector nodes, keeping the metadata for the information stored on Compute nodes. Therefore, you must also reset the Compute nodes.
4. Build the Insta node, per the section of the same name under "Recovering an Insta Node " on page 252, replacing all appearances of the cluster-name placeholder with the name of the cluster.
5. Build the Rubix node, per the section of the same name under "Recovering a Failed UI Node: Scenario 1" on page 258, replacing all appearances of the cluster-name placeholder with the name of the cluster.

Configuring the Collector and Compute Nodes

1. If you are configuring a Collector or Compute node, log into GMS node:

```
admin@host> en
admin@host# conf t
```

2. Validate the modified xml file through the GMS applet before activating it.

3. Activate the XML which was used for installing the setup:

```
admin@host(config)# gms config mural.xml activate
admin@host(config)# install appliance cluster DN-CLUS-GMS
admin@host (config) # install appliance cluster cluster-name
COLL-CLUS-GMS
```

4. Check for installation status of the Compute nodes cluster and wait for successfully installed message to appear for all nodes before proceeding to next step.

5. Check for installation status of Collector cluster and wait for a successfully installed message to appear for both Collectors before proceeding.

Starting Node Processes

1. Add the ASR user again on both the master and standby collectors:

```
> en
# conf t
(config) # username <username> password <password>
(config) # wri mem
```

2. Start sending data to the MURAL platform from the ASR 5000.

3. Send the EDR and bulk stats data feeds to the MURAL Platform. If the ASR is used as an input, note the start time from the filename that is created in the Collector folder. The start time is used in the next step.

4. Set the `dataStartTime`.

5. Log into the GMS to set the `dataStartTime` in the configuration of both the master and standby Collector nodes, based upon the start time you noted in the previous step. This script sets the `dataStartTime` to the time from which EDR and bulkstats data starts entering the system.

   Run:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# _shell
admin@host# mount -o remount,rw /
admin@host# cd /opt/deployment/Mural_setStartTime/
admin@host# ./setOozieTime --dataStartTime data-start-time --
node collector-mgmt-ip --password admin-user-pwd
```

6. Execute the script to set the data start times to the time from which EDR and bulkstats data starts coming into the system.

   For example, if EDR and bulkstats data starts coming into the system from 1st April, 2014, 06:00 onwards, run the following scripts with the start_time value as "2014-04-01T06:00Z":

   Note: Enter minutes as a multiple of 5. For example, "2014-04-01T06:00Z".

   ```
   admin@host# ./setOozieTime --dataStartTime  2014-04-01T06:00Z
   --node 192.168.147.11 --password admin@123
   ```

7. Execute the `Set Job Time Script` for both the master and standby Collector node.

8. Restore the IBs (whitelist entries). See "Backing Up and Restoring Information Bases" on page 181.

Start the Data Processing

1. Start oozie jobs:

   a. Log into the master Collector node and go to the pmx oozie subshell:

   ```
   > en
   # conf t
   (config)# pmx subshell subshell-name
   ```

   b. Start all jobs:

   ```
   pm extension (oozie)> run job all
   ```

The output shows all the jobs that were initiated and if the jobs started successfully or not.

Recovering from Multiple Insta Node Failures

Perform the following steps:

1. Log into the master Collector node and go to the pmx oozie subshell:

```
> en
# conf t
(config)# pmx subshell subshell-name
```

2. Stop the jobs:

```
pm extension (subshell-name)> stop jobname job-name
```

   Where `job-name` is:

   - CubeExporter
   - BulkStatExporter_15min
   - ErrorExport (if HET is enabled)

3. Log into each UI Node:

```
> en
# conf t
```

4. Stop the processes:

```
(config)# pm process rubix terminate
```

5. Log into GMS node and activate the XML if not already activated when manufacturing the nodes using pxe boot. This XML was updated to include the details of the new insta nodes.

```
admin@host> en
admin@host# conf t
admin@host(config)# gms config muralconfigure.xml activate
```

```
admin@host(config)# install appliance cluster clustername
INSTA-CLUS-GMS
```

6. Check for installation status of Insta cluster and wait for the message Node successfully installed to appear for both Insta nodes before proceeding to next step. The process might take 1 hour to complete:

   ```
   admin@host(config)# install appliance show installation-status
   cluster INSTA-CLUS-GMS
   ```

   Note: Before performing the next step, ensure to generate and push IBs from Master Name node through the `aggregation_center` and `bulkstats` subshell.

7. Start the CubeExporter jobs from master Collector node so that processed mapreduce data gets pushed to Insta:

   a. Log into the master Collector node and go to the pmx oozie subshell:

      ```
      > en
      # conf t
      (config)# pmx subshell subshell-name
      ```

   b. Start CubeExporter Jobs:

      ```
      pm extension (subshell-name)> run jobname job-name
      ```

      Where job-name is:

      - CubeExporter
      - BulkStatExporter_15min
      - ErrorExport (if HET is enabled)

8. Now start UI processes on all UI nodes by following the steps in the *MURAL Installation Guide*.

Start UI Processes and Verify Data

If recovering from multiple UI (Rubix) node failures, complete steps are provided here:

1. Run the following commands on all the UI nodes:

```
> en
# conf t
(config) # pm process rubix restart
```

2. Log into GMS node:

```
admin@host> en
admin@host# conf t
```

3. Activate the XML which was used for installing the setup:

```
admin@host(config)# gms config mural.xml activate
admin@host(config)# install appliance cluster cluster-name
RUBIX-CLUS-GMS
```

4. Check for installation status of Rubix cluster and wait for successfully installed message to appear for both UI nodes before proceeding to next step.

   Note: This process may take 30 minutes to complete.

```
admin@host(config)# install appliance show installation-status
cluster RUBIX-CLUS-GMS
```

5. Once the above operations return success, push IBs from Collector nodes:

   a. Log into the master Collector and go to the the pmx `aggregation_center` subshell:

   ```
   > en
   # conf t
   (config)# pmx subshell subshell-name
   ```

   b. Push the IBs from the master Collector node to the two UI nodes.

      Run the following commands on the pmx `aggregation_center` sub-shell, then backout and run it on the bulkstats subshell:

```
pm extension (sub-shell)> push all ibs Control-IP-UI-
NODE-1
pm extension (sub-shell)> push all ibs Control-IP-UI-
NODE-2
```

Replacing `Control-IP-UI-NODE-1` and `Control-IP-UI-NODE-1`.

Note: If Anomaly is enabled, execute these commands in the `anomaly` subshell as well.

c. Start UI processes and verify data following the steps in the section of the same name (Rubix) under "Recovering a Failed UI Node: Scenario 1" on page 258.

Replacing a LUN on a PGSQL Node

This chapter describes how to replace a failed logical unit number (LUN) on a node that is also running Postgres (PGSQL) database and then restore the previously backed up PGSQL data on it. In MURAL, pgsql process runs on GMS nodes and Insta nodes. In case of Standard Pack setup, pgsql process runs on Collector (Name Node) nodes as well. Follow the procedure in the chapter as applicable for your site.

If the LUN failed on the standby node, there is no functional impact and you can proceed with recovery steps as described in "Case 1" on the next page.

If the LUN failed on the current master node, then you are required to cleanup the current data and restore from the previous backup as described in "Case 2" on page 274.

Caution: If any of these steps fail, do not proceed. Instead, contact Technical Support to resolve the issue.

Prerequisites

Ensure that the following requirements are met:

- The hardware is set up and configurations are done for the new LUN. It is associated with the corresponding node. The WWID of the new LUN to be used with the node is known.

- Switchover must not occur during the restore or recovery procedure.

Case 1

Perform the following procedure if the LUN failed on the standby node:

1. Assign a new LUN with a new WWID from the storage EMC to this node.

   Remove the failed LUN from EMC so that it gets dissociated from the node.

2. SSH to the standby node (on which the LUN failed) and stop the pgsql process:

```
> en
# conf t
(config)# pm process pgsqld terminate
```

3. Reboot this node so that the new LUN starts showing up on this node:

```
> en
# conf t
(config)# wr mem
(config)# reload
```

   Note: Wait for the node (on which the LUN failed) to come up as the standby node again.

4. Unmount and remove the old filesystem that was managing the old LUN. On the standby node (on which the LUN failed), check the filesystem name:

```
> en
# conf t
(config) # show tps fs
```

   The output may resemble:

```
Filesystem fs1
Enabled: yes
Mounted: yesMount point: /data/pgsql
```

```
UUID: f5e896d2-d84c-4d0a-abc7-df11762766ac
WWID: 3600601609cc03000c18fa002ebede111
```

Note: The output may display a list additional file systems for other processes.

5. Remove the filesystem that has /data/pgsql as its mount point:

   ```
   (config t)# no tps fs <fs_name> enable
   (config t)# no tps fs <fs_name>
   ```

   For example:

   ```
   (config t)# no tps fs fs1
   (config t)# show tps fs
   ```

   The output should no longer list the filesystem that was deleted.

6. Check that the new LUN is displayed in the output of multipath on the Standby node where the old LUN failed. You can check that the WWID listed in brackets () in this output is same as that of the new LUN that you assigned for this node.

   ```
   (config) # tps multipath renew
   (config) # tps multipath show
   mpathc(3600601609cc03000c08fa002ebede111) dm-1 SGI,RAID 0
   size=5.0T features='0' hwhandler='0' wp=rw
   `-+- policy='round-robin 0' prio=1 status=active
   `- 2:0:0:0 sdg 8:96 active ready running
   ```

7. Log in to the standby node (on which pgsql LUN failed) and format the new LUN with the WWID as assigned by storage:

   ```
   (config)# tps fs format wwid <wwid_of_new_lun> fs-type ext3
   no-strict
   ```

   For example:

```
(config)# tps fs format wwid 3600601609cc03000c08fa002ebede111
fs-type ext3 no-strict
```

The output may resemble:

```
[admin@Collector-1 ~]# tps fs format wwid
36d4ae520006e6f4f00000da053e0d8d5 fs-type ext2 no-strict
0.1% [#####]
```

8. Configure the file system to manage the new LUN. On the standby node (on which the LUN failed and new LUN is assigned), add the filesystem that was removed earlier.

```
(config)# tps fs fs1
(config)# tps fs fs1 wwid <WWID_Of_New_LUN>
(config)# tps fs fs1 mount-point /data/pgsql
(config)# tps fs fs1 enable
(config)# write memory
```

   Ensure that there is no trailing backslash (/) at the end while specifying the mount-point. Specifying mount-point as /data/pgsql/ is incorrect configuration. Replace <WWID_Of_New_LUN> in the preceding command with WWID ID of the new LUN assigned for this pgsql on this node.

   For example:

```
(config)# tps fs fs1 wwid 3600601609cc03000c08fa002ebede111
```

   Check if the storage is mounted successfully. The output must list the LUN mounted on "/data/pgsql" in read/write mode.

```
(config) # _shell
# mount | grep "pgsql"
/dev/mapper/mpathcp1 on /data/pgsql type ext3
(rw,relatime,errors=continue,barrier=1,data=ordered)
```

9. Start the process on the standby node on which the LUN maintenance activity is completed:

```
# cli -m config
(config) # pm process pgsql restart
```

Note: Wait for 10 minutes to let the processes initialize the system.

Case 2

Caution: Data such as filters and users that were created after the last backup of the pgsql database, will be lost.

Perform the following procedure if the LUN failed on the current master node:

1. Log in to the master and standby pgsql nodes where the LUN is to be replaced, and stop the process:

   ```
   > en
   # conf t
   (config) # pm process pgsqld terminate
   ```

2. Stop processes on other nodes:

   a. If the pgsql LUN failed on the master Insta Node, stop UI processes. On the master UI node, log in and stop all java processes running on the system:

      ```
      (config) # pm process rubix terminate
      ```

      On the standby UI node, repeat this.

   b. If the pgsql LUN failed on the master Insta Node, and it is a Starteror Medium Pack setup, stop Hive jobs. Log in to the master Name node (Collector node) and stop all running Hive jobs:

      ```
      # pmx
      pm extension> subshell oozie
      pm extension (oozie)> stop jobname AnomalyHiveHourly
      pm extension (oozie)> stop jobname AnomalyHiveDaily
      pm extension (oozie)> stop jobname AnomalyHiveAggDaily
      pm extension (oozie)> stop jobname AnomalyHiveMonthly
      ```

```
pm extension (oozie)> stop jobname
AnomalyCleanupHiveHourly
pm extension (oozie)> stop jobname
AnomalyCleanupHiveDaily
pm extension (oozie)> stop jobname
AnomalyCleanupHiveAggDaily
pm extension (oozie)> stop jobname
AnomalyCleanupHiveMonthly
pm extension (oozie)> stop jobname bsHiveJob
pm extension (oozie)> stop jobname bsCleanupHive
pm extension (oozie)> stop jobname edrFlowHiveJob
pm extension (oozie)> stop jobname edrHttpHiveJob
pm extension (oozie)> stop jobname edrflowCleanupHive
pm extension (oozie)> stop jobname edrhttpCleanupHive
```

Note: Stop the jobs that are configured for a given site.

c. If the pgsql LUN failed on the master Name node (in case of Standard pack setup only), stop Hive jobs as described in the preceding step.

d. If the pgsql LUN failed on the master GMS node, stop the running GMS server running on master and standby GMS nodes:

```
(config) # pm process gms_server terminate
(config) # show pm process gms_server
State:
Current status: stopped
```

3. Assign a new LUN with a new WWID from the storage EMC to this node. Remove the old bad LUN from EMC so that it gets dissociated from the node.

4. Reboot this node so that the new LUN starts showing up on this node:

```
> en
# conf t
```

```
(config)# wr mem
(config)# reload
```

Note: Wait for the node (on which the LUN failed) to come up as a standby node again.

Stop the pgsql process on the rebooted node:

```
(config) # pm process pgsqld terminate
```

5. Unmount and remove the old filesystem that was managing the old LUN. On the Standby Node (on which LUN has failed), to check the filesystem name:

```
> en
# conf t
(config) # show tps fs
```

The output may resemble as follows:

```
Filesystem fs1
Enabled: yes
Mounted: yes
Mount point: /data/pgsql
UUID: f5e896d2-d84c-4d0a-abc7-df11762766ac
WWID: 3600601609cc03000c18fa002ebede111
```

Note: The output may also display a list of additional file systems for other processes.

6. Remove the filesystem that has "/data/pgsql" as its mount point:

```
(config)# no tps fs <fs_name> enable
(config)# no tps fs <fs_name>
```

For example,

```
(config)# no tps fs fs1 enable
(config)# no tps fs fs1
(config)# show tps fs
```

The output should no longer list the filesystem that was deleted.

7. Check if the new LUN is displayed in the output of multipath on the Standby node where old LUN had failed. You can check that the WWID listed in brackets () in this output is same as that of the new LUN that you assigned for this node.

```
(config) # tps multipath renew
(config) # tps multipath show
mpathc(3600601609cc03000c08fa002ebede111) dm-1 SGI,RAID 0
size=5.0T features='0' hwhandler='0' wp=rw
`-+- policy='round-robin 0' prio=1 status=active
`- 2:0:0:0 sdg 8:96 active ready running
```

8. Log in to the standby node (on which pgsql LUN failed) and format the new LUN with the WWID as assigned by storage:

```
(config)# tps fs format wwid <wwid_of_new_lun> fs-type ext3
no-strict
```

For example,

```
(config)# tps fs format wwid 3600601609cc03000c08fa002ebede111
fs-type ext3 no-strict
```

The output may resemble as follows:

```
[admin@Collector-1 ~]# tps fs format wwid
36d4ae520006e6f4f00000da053e0d8d5 fs-type ext2 no-strict
0.1% [#####]
```

9. Configure the file system to manage the new LUN. Log in to the standby node (on which the LUN had failed and new LUN is assigned), and add the filesystem that was removed earlier.

```
(config)# tps fs fs1
(config)# tps fs fs1 wwid <WWID_Of_New_LUN>
(config)# tps fs fs1 mount-point /data/pgsql
(config)# tps fs fs1 enable
(config)# write memory
```

Ensure that there is no trailing backslash (/) at the end while specifying the mount-point. Specifying mount-point as `/data/pgsql/` is incorrect configuration. Replace `<WWID_Of_New_LUN>` in the preceding command with the WWID of the new LUN assigned for this pgsql on this node.

For example,

```
(config)# tps fs fs1 wwid 3600601609cc03000c08fa002ebede111
```

Check if the storage got mounted successfully. The output must list the LUN mounted on `/data/pgsql` in read/write mode.

```
(config) # _shell
# mount | grep "pgsql"
/dev/mapper/mpathcp1 on /data/pgsql type ext3
(rw,relatime,errors=continue,barrier=1,data=ordered)
```

10. Check that the pgsql data directory is clean and then PGSQL process is running on the master (GMS, Insta, or Name) node:

```
(config) # _shell
# cd /data/pgsql
# rm -rf *
# cli -m config
(config) # pm process pgsqld restart
(config) # show pm process pgsqld
```

The current status should be shown as running:

```
Current status: running
```

Repeat this step on the standby GMS, NameNode (Collector), or Insta node.

Note: Wait for at least 10 minutes to let the processes initialize the system.

11. Restore the corresponding backed up PGSQL DB on the master (GMS, Insta, or Name) node:

    Note: The pgsql used to restore must be the same that was backed up from this module’s master node (GMS to GMS, Insta to Insta, and NameNode to Namenode).

    ```
    (config) # _shell
    # cd /data# tar -xvzf /data/pgsql_bkup.tgz
    # cli -m config
    (config) # pgsql file-path <path to backup file that has to be
    restored> dbname <name of db/all> restore
    ```

    For example,

    ```
    (config) # pgsql file-path /data/pgsql_bkup dbname all restore
    ```

12. If the pgsql LUN failed on the previously master Insta Node or the Master NameNode, restart Hive jobs that were stopped earlier. On the master Namenode (Collector node), log in and restart all Hive Jobs as configured at the site:

    ```
    # pmx
    pm extension> subshell oozie
    pm extension (oozie)> run job AnomalyHiveHourly
    pm extension (oozie)> run job AnomalyHiveDaily
    pm extension (oozie)> run job AnomalyHiveAggDaily
    pm extension (oozie)> run job AnomalyHiveMonthly
    pm extension (oozie)> run job AnomalyCleanupHiveHourly
    pm extension (oozie)> run job AnomalyCleanupHiveDaily
    pm extension (oozie)> run job AnomalyCleanupHiveAggDaily
    pm extension (oozie)> run job AnomalyCleanupHiveMonthly
    pm extension (oozie)> run job bsHiveJob
    pm extension (oozie)> run job bsCleanupHive
    ```

```
pm extension (oozie)> run job edrFlowHiveJob
pm extension (oozie)> run job edrHttpHiveJob
pm extension (oozie)> run job edrflowCleanupHive
pm extension (oozie)> run job edrhttpCleanupHive
```

13. If the pgsql LUN failed on the previously master Insta Node, restart processes on the master and standby UI nodes that were stopped earlier.

    a. Log in to the master UI node and run the following command:

    ```
    (config) # pm process rubix restart
    ```

    b. Log in to the standby UI node and repeat the preceding step.

    c. Verify that you can access the UI. Type the URL, https://<-domainName>:21443/ in your browser.

       For example,

       https://demo.sanmateo.com:21443/

       Username: admin

       Password: admin123

    d. Navigate to the **Configure** tab in UI and check that all old users, whose accounts were created on the setup from which backup is taken, are still available.

    e. Verify that filters are available.

14. If the pgsql LUN failed on the older master GMS node, restart the process on Master GMS node that were stopped earlier:

    ```
    (config) # pm process gms_server restart
    (config) # show pm process gms_server
    State:
    Current status: running
    ```

For both the cases, Case 1 and Case 2 update the new LUN ID in the GMS xml file for future use.

1. Log in to the GMS UI.
2. Load the active xml by using the **Load file from server** option.
3. Navigate to the **Server_details** tab, select the chassis where this node is installed.
4. Select the **Node** subtab, navigate to the **Storage** section, and change the WWID of the LUN associated with `/data/pgsql` on this node.

Replacing a LUN on the Insta Node

This topic describes how to replace a LUN on an Insta node in the event of a failure.

If any of these steps fail, do not proceed. Instead, contact Technical Support to resolve the issue.

Before You Begin

Ensure that you have the most recent database backup available for restoring the data. When an Insta node's LUN fails, you have to reconfigure the servers and restore the data which was last backed-up. Therefore, there might be some loss of data, depending on how much time has passed between the last backup and the failure.

Stopping Processes from Accessing the Insta Cluster

1. Log into the master Collector node and go to the pmx oozie subshell:

   ```
   > en
   # conf t
   (config)# pmx subshell subshell-name
   ```

2. Stop all jobs:

   ```
   pm extension (oozie)> stop jobname all
   ```

   Note: This command takes approximately 20-30 minutes to stop all the jobs.

3. After it exits to the prompt, run `show coordinator RUNNING jobs` to check if any jobs are still running. If so, wait for them to stop or manually stop them individually by running this command:

   ```
   pm extension (oozie)# stop job name job-name
   ```

4. Log in to the master UI node and stop all tomcats:

```
host [cluster: master]> en
host [cluster: master]# configure terminal
(config) # pm process rubix terminate
```

5. Repeat step 3 on the standby UI node.

Configuring the LUN

1. Get a new LUN assigned from the EMC storage to the Insta nodes (The LUN requirements for an Insta node must be checked from the sizing sheet). This LUN will be shared between both the Insta nodes.

2. Reboot both the Insta nodes (master and standby) individually:

```
host [cluster : master|standby]> enhost [cluster :
master|standby]# config thost [cluster : master|standby]
(config)# write memory
host [cluster : master|standby](config)# reload
```

   Wait for both the nodes to come up before proceeding to the next step.

3. Log into each Insta node (master and standby) and verify whether or not the new LUN is listed:

```
host [cluster : master|standby]> enhost [cluster :
master|standby]# conf thost [cluster : master|standby](config)
# tps multipath show
```

   Where the output shows:

   - The new LUN (can be checked by comparing its WWID displayed in the bracket).
   - Only one `dbroot` (either `dbroot1` or `dbroot2` depending on which one has not failed) or the failed LUN might be shown with a faulty status.

   For example,

```
dbroot1 (3600601609cc03000f30f7f5debede111) dm-3 SGI,RAID 10
size=1.8T features='0' hwhandler='0' wp=rw
```

```
|-+- policy='round-robin 0' prio=1 status=active
| `- 1:0:0:3 sdf 8:80  active ready running
|-+- policy='round-robin 0' prio=1 status=enabled
| `- 2:0:0:3 sdk 8:160 active ready running
`-+- policy='round-robin 0' prio=1 status=enabled
  `- 2:0:1:3 sdp 8:240 active ready running
mpathf (3600601609cc030004cb68d713ecce211) dm-4 SGI,RAID 10
size=1.9T features='0' hwhandler='0' wp=rw
|-+- policy='round-robin 0' prio=1 status=active
| `- 1:0:0:4 sdg 8:96  active ready running
|-+- policy='round-robin 0' prio=1 status=enabled
| `- 2:0:0:4 sdl 8:176 active ready running
`-+- policy='round-robin 0' prio=1 status=enabled
  `- 2:0:1:4 sdq 65:0  active ready running
```

Updating the GMS and Reconfiguring the Nodes

Modifying the Existing mural.xml

Use the following steps to open the GMS UI and replace the failed LUN's WWID in the `mural.xml` file with the new WWID for the Insta nodes.

1. Update the xml file to reflect the new storage WWID in place of the WWID that failed for both the Insta nodes.

   Load `mural.xml` to the GMS UI and edit it with the details of the new WWID.

   Where the xml file `mural.xml` is the same as what was used to bring up the setup with GMS.

2. Open following GMS link in a browser on a machine from which GMS is reachable: `http://192.168.147.18/applet`

   Where `192.168.147.18` is the IP address assigned to the management interface of the GMS.

3. On the GMS UI, from the second dropdown select the active xml and click **Load Config File from Server**. The configuration file is loaded into the

GMS UI.

The resulting output may resemble:

```
Loading Applet & Configuration...Please wait...
```

This XML file will be updated to configure GMS for the new LUN ID to be used with the Insta nodes.

4. Change the ID for the LUN associated with the Insta nodes on the **Server_ Details** tab:

   a. Go to server details tab and select the first Insta Node in the chassis.

   b. Scroll down and provide the WWID of the new LUN in place of the WWID of the failed LUN.

      The WWID of the LUN is listed in the output of the `tps multipath show` command that is run on the Insta nodes.

   c. Repeat steps 4a and 4b to change the WWID for the second Insta node details in **Server_Details** tab.

   d. Click **Validate** at the bottom of the screen and wait until it shows a success message. If an error is reported instead of success, fix the error and perform this step again.

   e. Click **Save Server File** and click **Yes** when prompted to rewrite the old xml file.

Installing the New XML File

1. Establish an SSH connection to the GMS using management IP and start the installation on the Insta cluster:

```
host [cluster : master|standby]> enhost [cluster :
master|standby]# config t
host [cluster : master|standby](config)# gms config mural.xml
activate
host [cluster : master|standby](config)# install appliance
cluster cluster-name cluster_name force-format
```

Where the `cluster-name` is the name of the Insta cluster and it will automatically come up when you press tab after typing `cluster-name`.

For example,

```
host [cluster : master|standby](config)# install appliance
cluster cluster-name INSTA-CLUS-GMS force-format
```

2. Monitor the status of the installation status on the Collector blades by executing this command to show the percentage completed:

```
host [cluster : master|standby](config) install appliance show
installation-status cluster INSTA-CLUS-GMS
```

Wait for the `Node successfully installed` message to appear before proceeding.

Restoring the Database

Follow the sections in this guide on performing a database restore. Refer to "Backing Up and Restoring a Large (TB) Insta Database" on page 167

Note: You do not need to change the database names while executing the database restore steps as this system has been freshly re-built.

Applying Configurations and Pushing IBs

Bulkstats Configurations

1. Go to the pmx bulkstats subshell:

```
> en
# conf t
(config)# pmx subshell subshell-name
```

2. Generate IBs on the pmx bulkstats subshell:

```
pm extension (subshell-name)> generate all ibs
```

3. Push generated IBs to required nodes:

```
pm extension (bulkstats)> push all ibs
```

IBs are fetched automatically on the standby node.

EDR Configurations

1. Go to the pmx `aggregation_center` subshell.

```
> en
# conf t
(config)# pmx subshell subshell-name
```

2. Generate IBs and push them to required nodes:

```
pm extension (subshell-name)> generate all ibs
pm extension (subshell-name)> push all ibs
```

IBs are fetched automatically on the standby node.

Anomaly Configurations

Perform the following steps only if Anomaly is enabled:

1. Go to the pmx anomaly subshell:

```
> en
# conf t
(config)# pmx subshell subshell-name
```

2. Update IBs on the pmx anomaly subshell:

```
pm extension (subshell-name)> update all ibs
```

IBs are fetched automatically on the standby node.

Restart Jobs and Processes

1. Log into the master Collector node and go to the pmx oozie subshell:

```
> en
# conf t
(config)# pmx subshell subshell-name
```

2. Start all jobs:

```
pm extension (oozie)> run job all
```

3. Log into the master UI node, go to the configure terminal, and start all the tomcat instances:

```
host [cluster : master]> en
host [cluster : master]# conf t
(config)# pm process rubix restart
```

4. Log in to the standby UI node and repeat step 3 to start all the tomcat instances.

Replacing a LUN on the Compute Node

This topic describes how to replace a LUN on a Compute node in the event of a failure.

Before You Begin

Before proceeding, ensure the following Before You Begin are complete:

- The hardware side setup and configurations have been done for the new LUN and it is associated with the Compute node.
- The WWID of the new LUN to be used with Compute node is known.

If any of these steps fail, do not proceed. Instead, contact Technical Support to resolve the issue.

To Replace a LUN of the Compute node:

1. Assign a new LUN with a new WWID from the storage EMC to the Compute node.

   To find the WWID of the LUN, log into the EMC, click the **Storage** tab, click **LUNs**, highlight the destination LUN, and click **Properties**. The EMC Web UI shows the **unique ID**; for the WWID, remove the separator ': ' from the unique ID and prefix the complete unique ID with 3, for example:

   360060160c7102f004887f4015d49e211

2. Remove the failed LUN from EMC so that it is dissociated from the Compute node.

3. Log into the master Collector node and stop all processes on the Compute node:

   ```
   host [cluster : master]> en
   host [cluster : master]# conf t
   host [cluster : master](config)# pmx subshell hadoop_yarn
   pm extension (hadoop_yarn)> stop daemon nodemanager Control-
   IP-Of-ComputeNode
   ```

```
pm extension (hadoop_yarn)> stop daemon datanodeControl-IP-Of-
ComputeNode
```

Note: Provide the control IP of the Compute node whose LUN has failed, in the above commands.

4. Establish an SSH connection to the Compute node on which the LUN failed and check the name of the filesystem that was managing the old LUN:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# show tps fs
```

   The resulting output may resemble:

```
Filesystem fs1
Enabled: yes
Mounted: yes
Mount point: /data/hadoop-admin
UUID: f5e896d2-d84c-4d0a-abc7-df11762766ac
WWID: 3600601609cc03000c18fa002ebede111
```

5. Unmount and remove the filesystem:

```
host [cluster : master|standby](config)# no tps fs fs-name
enable
host [cluster : master|standby](config)# no tps fs fs-name
```

   Where `fs-name` is the same name seen in the output of step 4, above.

   For example,

```
host [cluster : master|standby](config)# no tps fs fs1 enable
host [cluster : master|standby](config)# no tps fs fs1
```

   For example, the output may resemble:

```
No Filesystems Configured!
```

6. Configure the new LUN on the Compute node:

Reboot the node so that node detects the newly assigned LUN:

```
host [cluster : master|standby]> enhost [cluster :
master|standby]# config t
host [cluster : master|standby](config t)# reload
```

Execute the following commands after the node is up:

```
host [cluster : master|standby](config)# tps multipath renew
(config)# tps multipath show
```

You can check that the WWID listed in brackets in the following output is same as that of the new LUN, which we assigned to the new node.

The resulting output may resemble:

```
mpathe (3600601609cc03000c08fa002ebede111) dm-1 SGI,RAID 0
size=5.0T features='0' hwhandler='0' wp=rw
`-+- policy='round-robin 0' prio=1 status=active
  `- 2:0:0:0 sdg 8:96  active ready  running
```

7. Log into the Compute node and format the new LUN with the WWID as determined by the previous step:

```
(config)# tps fs format wwid <wwid_of_new_lun> fs-type ext3
no-strict
```

For example,

```
(config)# tps fs format wwid 3600601609cc03000c08fa002ebede111
fs-type ext3 no-strict
```

The resulting output may resemble:

```
[[admin@Collector-1 ~]# tps fs format wwid
36d4ae520006e6f4f00000da053e0d8d5 fs-type ext2 no-strict
   0.1%  [#####]
```

8. On the Compute node whose LUN failed and was replaced, configure the file

system to manage the new LUN:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# tps fs fs1
host [cluster : master|standby](config)# tps fs fs1 wwid WWID-
Of-New-LUN
host [cluster : master|standby](config)# tps fs fs1 mount-
point /data/hadoop-admin
host [cluster : master|standby](config)# tps fs fs1 enable
host [cluster : master|standby](config)# write memory
```

Note: The **mount-point** value should not end with a /. Do not write **/data/hadoop-admin/** as that would be incorrect.

Where *WWID-Of-New-LUN* is the WWID of the new LUN assigned for this Collector.

For example,

```
host [cluster : master|standby](config)# tps fs fs1 wwid
3600601609cc03000c08fa002ebede111
```

9. Verify that the storage was successfully mounted. The output must list the LUN mounted on the **/data/hadoop-admin** in read and write mode:

```
> en
# _shell
# # mount | grep hadoop-admin
/dev/mapper/mpathep1 on /data/hadoop-admin type ext3
(rw,relatime,errors=continue,barrier=1,data=ordered)
```

10. Log into the master Collector node and start the TPS and Collector processes on the Compute node:

```
> en
host [cluster : master]# conf t
host [cluster : master](config)# pmx subshell hadoop_yarn
```

```
pm extension (hadoop_yarn)> start daemon datanode Control-IP-
Of-ComputeNode
pm extension (hadoop_yarn)> start daemon nodemanager Control-
IP-Of-ComputeNode
pm extension (hadoop_yarn)> quit
pm extension > quit
```

Where *Control-IP-Of-ComputeNode* is the control IP of the Compute node whose LUN failed.

11. Log in to the master Collector node and verify that the Compute node was added to the HDFS cluster:

    Note: The output of this command should list the Compute node with status of **Normal**.

```
# _shell
# hdfs dfsadmin -report

Configured Capacity: 3246762123264 (2.95 TB)
Present Capacity: 3083690599476 (2.8 TB)
DFS Remaining: 1428612595712 (1.3 TB)
DFS Used: 1655078003764 (1.51 TB)
DFS Used%: 53.67%
Under replicated blocks: 14608
Blocks with corrupt replicas: 0
Missing blocks: 0
-------------------------------------------------
Datanodes available: 3 (3 total, 0 dead)
Name: 10.10.2.17:50010
Decommission Status : Normal
Configured Capacity: 1082254041088 (1007.93 GB)
DFS Used: 220870 (215.69 KB)
Non DFS Used: 55249498426 (51.46 GB)
DFS Remaining: 1027004321792(956.47 GB)
DFS Used%: 0%
DFS Remaining%: 94.89%
Last contact: Mon Apr 15 11:32:39 GMT 2013

Name: 10.10.2.13:50010
Decommission Status : Normal
Configured Capacity: 1082254041088 (1007.93 GB)
DFS Used: 827246578190 (770.43 GB)
Non DFS Used: 53424977394 (49.76 GB)
DFS Remaining: 201582485504(187.74 GB)
DFS Used%: 76.44%
DFS Remaining%: 18.63%
Last contact: Mon Apr 15 11:32:40 GMT 2013

Name: 10.10.2.14:50010
Decommission Status : Normal
Configured Capacity: 1082254041088 (1007.93 GB)
DFS Used: 827831204704 (770.98 GB)
Non DFS Used: 54397047968 (50.66 GB)
DFS Remaining: 200025788416(186.29 GB)
DFS Used%: 76.49%
DFS Remaining%: 18.48%
Last contact: Mon Apr 15 11:32:40 GMT 2013
```

12. Log in to the Compute node and verify that the processes are running:

```
# ps -ef| egrep -i "datanode|_nodemanager" |grep -v grep |awk
'{print $NF}
'
/opt/hadoop/conf:/opt/hadoop/etc/hadoop:/opt/hadoop/contrib/……
```

13. Update the new LUN ID in the GMS xml file for future use. Refer to the next section, *Updating GMS Files with New LUN ID*, for instructions.

Updating GMS Files with New LUN ID

Use the following steps to open the GMS GUI and replace the failed LUN's WWID in the **mural.xml** file with the new WWID for the Compute node.

1. Open following GMS link in a browser on a machine from which GMS Management Server is reachable: **http://192.168.147.18/applet**

   Where **192.168.147.18** is the IP Address assigned to the management interface of the GMS.

2. SCP or STFP the xml file to your local machine from the GMS directory path **/config/gms/**.

   Note: Pick the xml file which was originally used to configure the system (**mural.xml**) or the one which was last modified and saved for running another MOP.

   This xml file will be updated to configure GMS for the new LUN ID to be used with the Compute node.

3. On the GMS UI, enter the local path of the xml file and click Load XML so that configuration can be loaded into GMS UI.

   A message **Loading Applet & Configuration....Please wait ...** appears.

4. Change the ID for the LUN associated with the Compute node on the Server_Details tab:

   a. Go to server details tab and select the node in the chassis whose LUN failed.

   b. Scroll down to the **Storage_WWIDs** section and provide the WWID of the new LUN in place of the WWID of the failed LUN.

   c. Click **Validate** at the bottom of the screen and wait until it shows a success message. If an error is reported instead of success, fix the error and perform this step again.

   d. Re-name the updated xml file.

Next to the Save Server File button, enter **Compute_LUN_Fail-ure.xml**and click Save Server File.

e. If a success message displays, the file was saved to the GMS.

Replacing a LUN on a Collector Node

This topic describes how to replace a LUN, which is used for collector input data on the Collector node in the event of a failure.

If the failed LUN was associated with `/data/pgsql` mount point, follow the procedure described in "Replacing a LUN on a PGSQL Node" on page 270.

If the failed LUN was associated with `/data/collector` mount point, follow the procedure described in this chapter.

If the LUN failed on the current master node, it will not accept the files from the server. In such a case, perform a switchover on the collector by rebooting the current master. After this, continue with the recovery steps in the following sections.

If the LUN failed on the standby collector node, there is no functional impact and you can proceed with the recovery steps in the following sections.

If any of these steps fail, do not proceed. Instead, contact Technical Support to resolve the issue.

Before You Begin

Before proceeding with replacing and configuring the LUN, follow steps in this section to ensure the Before You Begin are completed.

1. Assign a new LUN with a new WWID from the storage EMC to this Collector node.

   Finding the WWID of LUNs

   1. Log into the EMC
   2. Click the **Storage** tab
   3. Click **LUNs**
   4. Highlight the destination LUN
   5. Click **Properties**

   The EMC Web UI shows the unique ID.

   For the WWID, remove the separator ':' from the unique ID and prefix the complete unique ID with 3, for example:

```
360060160c7102f004887f4015d49e212
```

2. Remove the failed LUN from the EMC so that it is dissociated from the Collector node.

3. Reboot this Collector node so that new LUN starts showing up on this node:

```
host [cluster : master]> en
host [cluster : master]# conf t
host [cluster : master](config)# wr mem
host [cluster : master](config)# reload
```

Note: Wait for the Collector node (on which LUN has failed) to come up as standby node.

4. Establish an SSH connection to the new standby Collector node (on which LUN has failed) and stop the Collector process:

```
host [cluster : master]> en
host [cluster : master]# conf t
host [cluster : master](config)# pm process collector
terminate
```

5. Unmount and remove the old filesystem that was managing the old LUN. On the standby Collector (on which LUN has failed), check the filesystem name:

```
host [cluster : standby]> en
host [cluster : standby]# conf
host [cluster : standby](config)# show tps fs
```

The resulting output may resemble:

```
Filesystem fs1
Enabled: yes
Mounted: yes
Mount point: /data/collector
```

```
UUID: f5e896d2-d84c-4d0a-abc7-df11762766ac
WWID: 3600601609cc03000c18fa002ebede111
```

Note: This output may list an additional file system for PGSQL based on the site requirements.

6. Go to the section on "Loading and Configuring the New LUN" below.

Loading and Configuring the New LUN

1. Now remove the filesystem that has the `/data/collector` as its mount point:

```
host [cluster : standby](config)# no tps fs fs_name enable
host [cluster : standby](config)# no tps fs fs_name
```

   Where `fs_name` is the filesystem name from the output in the `show tps fs` step.

   In this case, the filesystem name is `fs1`. So for this example, commands should be:

```
host [cluster : standby](config)# no tps fs fs1 enable
host [cluster : standby](config)# no tps fs fs1
```

2. Verify the filesystem has been removed:

```
host [cluster : standby](config)# show tps fs
```

   The resulting output must no longer list the filesystem that was deleted.

3. Check that the new LUN can be seen in the output of multipath on the standby Collector node where old LUN failed.

   The number listed inside the parentheses () of this output is the WWID, the value of the new LUN we have assigned for this collector:

```
host [cluster : standby](config)# tps multipath renew
host [cluster : standby](config)# tps multipath show
```

   The resulting output may resemble:

```
mpathc (3600601609cc03000c08fa002ebede111) dm-1 SGI,RAID 0
size=5.0T features='0' hwhandler='0' wp=rw
`-+- policy='round-robin 0' prio=1 status=active
  `- 2:0:0:0 sdg 8:96  active ready  running
```

4. Log into the standby Collector node (on which the LUN has failed) and format the new LUN with the WWID number determined by the output of the previous step:

```
host [cluster : standby](config)# tps fs format wwid <wwid_of_
new_lun> fs-type ext3 no-strict
```

For example,

```
host [cluster : standby](config)# tps fs format wwid
3600601609cc03000c08fa002ebede111 fs-type ext3 no-strict
```

The resulting output may resemble:

```
admin@Collector-1 ~]#  # tps fs format wwid
36d4ae520006e6f4f00000da053e0d8d5 fs-type ext2 no-strict
   0.1%  [#####
```

5. On the new standby Collector Node, configure the file system to manage the new LUN and add the filesystem that was removed earlier:

```
host [cluster : standby](config)# tps fs fs1host [cluster :
standby](config)# tps fs fs1 wwid WWID-Of-New-LUN
host [cluster : standby](config)# tps fs fs1 mount-point
mount-point-value
host [cluster : standby](config)# tps fs fs1 enable
host [cluster : standby](config)# write memory
```

Where:

- **WWID_Of_New_LUN** is the WWID ID of the new LUN assigned for this collector.
- **mount-point-value** is /data/collector, and it should not end with a /. This means that **/data/collector/** is incorrect.

For example, this is correct:

```
host [cluster : standby](config)# tps fs fs1 wwid
3600601609cc03000c08fa002ebede111
host [cluster : standby](config)# tps fs fs1 mount-point
/data/collector
```

6. Verify that the storage was successfully mounted. The output must list the LUN mounted on `/data/collector` in read and write mode:

```
host [cluster : standby](config)# _shell
# mount
```

The resulting output may resemble:

```
rootfs on / type rootfs (rw)
/dev/root on / type ext3
(ro,noatime,errors=continue,barrier=1,data=ordered)
devtmpfs on /dev type devtmpfs (rw,size=49566512k,nr_
inodes=12391628,mode=755)
none on /proc type proc (rw,nosuid,nodev,noexec,relatime)
...
/dev/mapper/mpathcp1 on /data/collector type ext3
(rw,relatime,errors=continue,barrier=1,data=ordered)
```

7. Start the tps and collector processes on the standby Collector node on which LUN maintenance activity is completed:

```
# cli -m config
(config)# pm process tps restart
```

Note: Wait 10 minutes after starting the tps process to let it initialize the system before starting the collector process.

```
(config)# pm process collector restart
```

8. Log into the master Collector node and re run gateway push steps to re-create the directories under "/data/collector":

```
> en
# conf t
(config) # pmx
pm extension> subshell aggregation_center
pm extension (aggregation center)> push gateway configuration
```

9. Update the new LUN ID in the GMS xml file for future use. For instructions, refer to the section, "Updating GMS XML File with New LUN ID" below.

Updating GMS XML File with New LUN ID

Use the following steps to open the GMS GUI and replace the failed LUN's WWID in the **mural.xml** file with the new WWID for the Collector node.

1. Open following GMS link in a browser on a machine from which GMS Management Server is reachable: `http://192.168.147.18/applet`

   Where **192.168.147.18** is the IP Address assigned to the management interface of the GMS.

2. SCP or STFP the xml file to your local machine from the GMS directory path `/config/gms/`.

   Note: Pick the xml file which was originally used to configure the system (`mural.xml`) or the one which was last modified and saved for running another MOP.

   This xml file will be updated to configure GMS for the new LUN ID to be used with the Collector node.

3. On the GMS UI, enter the local path of the xml file and click Load XML so that configuration can be loaded into GMS UI. The following message appears:

   ```
   Loading Applet & Configuration....Please wait ...
   ```

4. Change the ID for the LUN associated with the Collector node on the **Server_Details** tab:

a. Go to server details tab and select the node in the chassis whose LUN failed.

b. Scroll down to the **Storage_WWIDs** section and provide the WWID of the new LUN in place of the WWID of the failed LUN.

c. Click **Validate** at the bottom of the screen and wait until it shows a success message. If an error is reported instead of success, fix the error and perform this step again.

d. Re-name the updated xml file.

   Next to the Save Server File button, enter **Collector_LUN_Failure.xml** and click **Save Server File**.

e. If a success message displays, the file was saved to the GMS.

Replacing a LUN on Rubix Node

This topic describes how to replace a LUN on the Rubix node in the event of a failure.

Caution: If any of these steps fail, do not proceed. Instead, contact Technical Support to resolve the issue.

Before You Begin

The hardware side setup and configurations have been done for the new LUN and it is associated with the Rubix node. The WWID of the new LUN to be used with Rubix Node is known.

Assign a new LUN with a new wwid from the storage EMC to the Rubix Node

Also remove the old bad LUN from EMC so that it gets dissociated with the Rubix node.

Case 1: LUN on Master Rubix has Failed

Trigger a switchover so as the current master becomes the standby node. Afterwards proceed with the below series of steps of Case 2.

Case 2: LUN on Standby Rubix has failed

1. SSH to the master and standby Rubix node and stop all java processes:

```
host [cluster : master|standby](config)# rubix modify-app
process-name disable
host [cluster : master|standby](config)# rubix modify-app
process-name modify-instance 1 disable
```

   Where `process-name` is replaced with:

   - atlas
   - reportAtlas
   - bulkstats
   - rge
   - httperror

- ruleEngine
- launcher

2. Open GMS UI with the `mural.xml` (assuming this is the active xml) and edit the WWID of the standby Rubix node on the **Server_Details** tab.
3. Save the changes to the current XML by clicking **Save** on the GMS UI.
4. Now login to GMS node and activate the XML:

```
> en
# conf t
(config)# gms config mural.xml activate
```

5. Now run installation on standby Rubix node whose LUN has got changed:

```
(config)# install appliance cluster cluster-name RUBIX-Cluster
node Rubix-GMS-1 force-format
```

   Note: This command assumes that `Rubix-GMS-1` is the standby node on which faulty LUN has been replaced.

6. Now start processes on both Rubix nodes in the order below:
   a. Log in to the master UI node and start the EDR process:

   ```
   > en
   # conf t
   (config)# pm process rubix restart
   (config)# rubix modify-app atlas enable
   (config)# rubix modify-app atlas modify-instance 1 enable
   ```

   b. Log into the standby UI node and repeat the previous step.
   c. Log in to the master UI node and use these commands to start the following processes:

   ```
   host [cluster : master|standby](config)# rubix modify-app
   process-name enable
   ```

```
host [cluster : master|standby](config)# rubix modify-app
process-name modify-instance 1 enable
```

Where `process-name` is replaced with:

- atlas
- reportAtlas
- bulkstats
- rge
- httperror
- ruleEngine
- launcher

d. Log into the standby UI node and repeat the previous step.

7. Access the user interface (UI).