MURAL Operations and Troubleshooting Guide

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

Text Part Number: OL-29616-03
THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB’s public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED “AS IS” WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF INFRINGEMENT, RESULTING FROM THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Cisco and the Cisco Logo are trademarks of Cisco Systems, Inc. and/or its affiliates in the U.S. and other countries. A listing of Cisco’strademarks can be found at www.cisco.com/go/trademarks. Third party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1005R)

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

MURAL Operations and Troubleshooting Guide

Copyright © 2014, Cisco Systems, Inc. All rights reserved.
# Table of Contents

**Overview of MURAL** ................................................................. 8

- Understanding the Reflex Platform ........................................... 8
- Introducing Reflex Platform Technology .................................. 8
- Introducing the Reflex Platform ................................................. 9
- System Components and Flow of Data .................................... 10
- Information Bases ..................................................................... 19
- Scheduling and Running Jobs ..................................................... 19
- Understanding Data Bins ............................................................ 21
- Frequently Asked Questions About Data Flow and Display .......... 25

**Understanding MURAL System Basics** ..................................... 27

- System Basics .......................................................................... 27
- System Logs ............................................................................. 34
- Collection of Logs using Centralized Log Management ............... 40
- Starting and Stopping Jobs ......................................................... 42
- Basic CLI Commands ............................................................... 45

**Monitoring** ............................................................................. 50

- Monitoring Processes on All Nodes ........................................... 50
  - Viewing Process Status on Collector and Insta Nodes ............... 50
- Monitoring Collector Nodes ...................................................... 51
- Monitoring Oozie Jobs ............................................................. 53
- Monitoring Computed Data ....................................................... 55
- Monitoring Processes on the Insta Nodes .................................. 59
- System Alerts ........................................................................... 62
Viewing Reports for Uncategorized Entities .................................................. 66

**Adding New Components** ................................................................. 80

Adding a New ASR for EDR and Bulkstats ................................................. 81
Adding a UI Node to the Existing Cluster .................................................. 87
Adding a Compute Node to the Existing Cluster ......................................... 94
Adding Postdated Data to the Collector Using Replay Mode ......................... 99
Manually Manufacturing the Master GMS Blade ......................................... 104
Setting Up the Master GMS Node ............................................................ 110

**Modifying and Configuring Components** ............................................. 113

Modifying Gateway Attributes .................................................................. 113
Defining Top N Parameters ...................................................................... 119
Overriding Categorizations From Information Bases .................................... 122
Configuring ASR Settings for MURAL ...................................................... 130
Job Timeout Configurations and Auto Bin Sliding in Collector ...................... 133
Upgrading the Bulkstats Schema Version .................................................. 135

**Modifying the Database** ................................................................. 139

Switching from Replay to Live Mode to Receive Live Data ......................... 139
Cleaning and Replaying the Insta Database .............................................. 144
Update Databases for URL Categories and IMEI Numbers .......................... 159

**Backup Strategy** ................................................................. 162

Backing Up Data and IBs ........................................................................... 162
Backup GMS Configuration Files ............................................................. 163
Backing Up and Restoring Computed Data ............................................... 164
Backing Up and Restoring a Large (TB) Insta Database ............................ 170
Back Up and Restoring Information Bases ........................................... 181
Back Up Raw Data ............................................................................. 187
Back Up and Restoring PGSQL DB ..................................................... 189
Troubleshooting .............................................................................. 193
Troubleshooting User Interface Access Problems ............................... 193
Troubleshooting Abrupt Failure of Jobs ............................................ 195
Troubleshooting Servers After Chassis Failure ................................. 212
Troubleshooting Interruption of Data Flow to Collector Node ............. 213
Recovering Components .................................................................. 218
Recovering a Collector Node ............................................................. 218
Recovering a Compute Node ............................................................. 228
Recovering an Insta Node .................................................................. 234
Recovering a Failed UI Node: Scenario 1 .......................................... 241
Recovering a Failed UI Node: Scenario 2 .......................................... 246
Recovering from a Single Node Failure ............................................ 248
Restarting UI Processes After a UI Node Reboots .............................. 249
Recovering from Failure of Multiple Nodes of the Same Type .......... 253
Replacing a LUN on the Insta Node .................................................. 261
Replacing a LUN on Rubix Node ....................................................... 271
Restarting Bulkstats After Updating Start Time ............................... 274
Overview of MURAL

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.

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 \texttt{drbd} synchronizes the NameNode's metadata on the master and standby Collector node. The metadata file is \texttt{/dir/fsimage}. A corresponding edit log, \texttt{/dir/editlog}, records the last few insertions, deletions and movements within HDFS (which are not yet captured in the metadata).

\textbf{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.

\textbf{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.

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 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.

The Insta process is the primary interface with the Insta database. You can use all standard MySQL commands to access the data stored in it and view the table structure.
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.

**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 bases 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-cond-
ditions 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 interrupted. 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@collectornode1
enable
```

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:
- **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.

- **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.

- **Shell**—The unix regular terminal (or bash shell) used for commands that are not yet implemented in the analytics CLI and PMX software. 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. See "Component Specific CLI Commands" on page 1 for system configuration commands.

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:

<table>
<thead>
<tr>
<th>Cluster ID:</th>
<th>COL-CLUSTER1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cluster name:</td>
<td>COL1-CLUST</td>
</tr>
<tr>
<td>Management IP:</td>
<td>192.168.100.73/24</td>
</tr>
<tr>
<td>Cluster master IF:</td>
<td>eth0</td>
</tr>
<tr>
<td>Cluster node count:</td>
<td>2</td>
</tr>
<tr>
<td>Local name:</td>
<td>COL1-100-2</td>
</tr>
<tr>
<td>Local role:</td>
<td>master</td>
</tr>
<tr>
<td>Local state:</td>
<td>online</td>
</tr>
<tr>
<td>Master address:</td>
<td>192.168.100.2 (ext) 192.168.100.2 (int)</td>
</tr>
<tr>
<td>Master state:</td>
<td>online</td>
</tr>
</tbody>
</table>

   **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—Limited to and provides Hadoop level configuration
- subshell oozie—Limited to and provides Oozie level configuration
- subshell aggregation_center
- subshell bulkstat
- subshell healthcheck
- subshell iscsi
- subshell multipath
- subshell patch
- subshell platform
- subshell tps

Go to any one of these subshells using:

```plaintext
> 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.

The admin and monitor accounts are enabled initially and do not require a password for login. 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:
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cli</td>
<td>Configure CLI shell options</td>
</tr>
<tr>
<td>collector</td>
<td>Collector Configuration</td>
</tr>
<tr>
<td>enable</td>
<td>Enter enable mode</td>
</tr>
<tr>
<td>exit</td>
<td>Log out of the CLI</td>
</tr>
<tr>
<td>help</td>
<td>View description of the interactive help system</td>
</tr>
<tr>
<td>ping</td>
<td>Send ICMP echo requests to a specified host</td>
</tr>
<tr>
<td>show</td>
<td>Display system configuration or status</td>
</tr>
<tr>
<td>slogin</td>
<td>Log into another system securely using ssh</td>
</tr>
<tr>
<td>telnet</td>
<td>Log into another system using telnet</td>
</tr>
<tr>
<td>terminal</td>
<td>Set terminal parameters</td>
</tr>
<tr>
<td>tps</td>
<td>Third Party Process Starter</td>
</tr>
<tr>
<td>traceroute</td>
<td>Trace the route packets take to a destination</td>
</tr>
</tbody>
</table>

### Configuring User Account Commands

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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>username userid</td>
<td>Create a new user account.</td>
</tr>
<tr>
<td>username userid password</td>
<td>Set password for a user.</td>
</tr>
<tr>
<td>username userid disable</td>
<td>Disable a user account.</td>
</tr>
<tr>
<td>no username userid</td>
<td>Delete a user account.</td>
</tr>
<tr>
<td>show users</td>
<td>View a list of users currently logged in.</td>
</tr>
<tr>
<td>show whoami</td>
<td>View information about the current user.</td>
</tr>
<tr>
<td>show usernames</td>
<td>View information about all configured users.</td>
</tr>
<tr>
<td>set username privilege admin</td>
<td>Monitor</td>
</tr>
</tbody>
</table>
Creating a Read-Only Account

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

```
no username loginnname
disable username loginnname
capability monitor username loginnname
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.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ntp enable</td>
<td>enable NTP on a node</td>
</tr>
<tr>
<td>ntp disable</td>
<td>disable NTP on a node</td>
</tr>
<tr>
<td>show ntp</td>
<td>verify NTP status</td>
</tr>
<tr>
<td>show clock</td>
<td>show the current clock time</td>
</tr>
</tbody>
</table>
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.

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 maximum number of logging files
- Logging level—info

Do not change the defaults except as directed by Technical Support.

Logging messages are categorized at the following levels:

<table>
<thead>
<tr>
<th>Level name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>alert</td>
<td>Action must be taken immediately</td>
</tr>
<tr>
<td>crit</td>
<td>Critical conditions</td>
</tr>
<tr>
<td>debug</td>
<td>Debug-level messages</td>
</tr>
<tr>
<td>emerg</td>
<td>Emergency: system is usable</td>
</tr>
<tr>
<td>err</td>
<td>Error conditions</td>
</tr>
<tr>
<td>info</td>
<td>Informational messages</td>
</tr>
<tr>
<td>none</td>
<td>Disable logging</td>
</tr>
<tr>
<td>notice</td>
<td>Normal but significant condition</td>
</tr>
<tr>
<td>warning</td>
<td>Warning conditions</td>
</tr>
</tbody>
</table>

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:
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-admin-jobtracker* /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:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# pmx subshell subshell-name
```

2. ==

Login and go to the pmx oozie subshell:
```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# pmx subshell
```

**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-admin-jobtracker* /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/hadoop-root-tasktracker* /data/Log_dump
```
Collecting Logs from a Caching Compute Node

- To collect Calpont 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/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/reportAtlas/1/bin/rubix*.log /data/Log_dump/edr_cacheless/
  ```

- To collect FLEX logs on a master UI node:
  
  ```
  # cp /data/instances/atlas/1/bin/flex*.log /data/Log_dump/
  ```

- To collect Catalina logs on a master UI node:
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-CISCO-COL-151-79-20130405-104001.txt
   SYSDUMP=/var/opt/tms/sysdumps//sysdump-CISCO-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 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 activate another xml, then the log collection will not automatically be enabled. 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:** Rest 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-UCS-2.

2. Run an `ls` command to list nodes 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. The total space of all logs from a particular host must not exceed 100 GB and no file older than 30 days needs to be kept. Files beyond these bounds are deleted.

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 individually start or stop 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, go to `shell` and invoke a new CLI shell before going to the oozie subshell:

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

2. Run the following command:

```bash
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. In this section, the EDR job is used as an example.

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

```bash
host [cluster : master]> en
host [cluster : master]# _shell
[admin@host ~]# cli -m config
host [cluster : master](config)# pmx subshell oozie
```
2. Stop the EDR job:

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

The resulting output may resemble:

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

3. In order to stop and re-run the EDR 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 EDR job:

   a. A similar update can be done for other jobs also if required. Set the new time from which you want to run the job, in this example, the EDR and cube exporter jobs:

```
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 EDR attribute jobStart 2012-11-05T03:00Z
setting attribute (jobStart)
```

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

   b. Run the following command for both the EDR and CubeExporter jobs:
pm extension (oozie)> rollback 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://MUR-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 EDR 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
Basic CLI Commands

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**

ntpd date <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://admin:password@hostname/config/db/<filename>
Displaying Software Versions and Images

Display current software version

    show version

Display all available images

    show images
Monitoring

This section provides an overview of monitoring and administration commands available on a typical MURAL installation. If not yet familiar with the system architecture and deployment topology, see "Overview of MURAL" on page 8.

Monitoring Processes on All Nodes

To show processes managed by process-monitor (PM):

```
host [cluster : master|standby](config)# show pm process
```

The resulting output may resemble:

<table>
<thead>
<tr>
<th>Process acctd (AAA Accounting Daemon)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration:</td>
</tr>
<tr>
<td>Launchable: yes</td>
</tr>
<tr>
<td>Auto-launch: no</td>
</tr>
<tr>
<td>Auto-relaunch: yes</td>
</tr>
<tr>
<td>Launch path: /opt/tms/bin/acctd</td>
</tr>
<tr>
<td>Re-exec path: (none)</td>
</tr>
</tbody>
</table>

Viewing Process Status on Collector and Insta Nodes

On a Collector or Insta node, to check that the process is running:

```
host [cluster : master|standby](config)# show pm process node-type
```

Where `node-type` is either collector or insta.

The resulting output may resemble:

<table>
<thead>
<tr>
<th>Process collector</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration:</td>
</tr>
<tr>
<td>Launchable: yes</td>
</tr>
<tr>
<td>Auto-launch: yes</td>
</tr>
<tr>
<td>Auto-relaunch: yes</td>
</tr>
<tr>
<td>Launch path: /opt/tms/bin/collector</td>
</tr>
</tbody>
</table>
Re-exec path:  (none)
ARGV:       /opt/tms/bin/collector
Max snapshots:  10
Launch order:  0
Launch timeout:  0
Shutdown order:  0
Kill timeout:  10 sec
CPU Affinity:  (not set)
Test liveness:  yes
Hung count:  4

State:
Current status:  running
PID:  28755
Num. failures:  0
Last launched:  2012/09/18 16:35:48.575 (21 days 1 hr 44 min 25.109 sec ago)
Last terminated:  2012/09/18 16:35:48.574 (21 days 1 hr 44 min 25.110 sec ago)
Next launch:

And if test liveness = yes, then the following output would be at the end:

Num. liveness timeouts:  0

Monitoring Collector Nodes

Restarting the Collector Process

To restart the Collector process:

```
host [cluster : master|standby](config)# pm process collector restart
```
**Viewing Status of Collector Counters**

To check that the Collector process is collecting data:

```
host [cluster : master|standby](config)# collector stats instance-id 1 adaptor-stats [ edrflow | edrhttp ] ?
```

The resulting output may resemble:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>average-bin-size</td>
<td>Display average bin size of the adaptor since start</td>
</tr>
<tr>
<td>average-flow-rate</td>
<td>Display average flow rate of the adaptor since start</td>
</tr>
<tr>
<td>dropped-flow</td>
<td>Display number of dropped flows of the adaptor since start</td>
</tr>
<tr>
<td>last-freezed-bin</td>
<td>Display last freezed bin time of the adaptor</td>
</tr>
<tr>
<td>last-freezed-bin-size</td>
<td>Display last freezed bin size of the adaptor</td>
</tr>
<tr>
<td>max-flow</td>
<td>Display max flow rate of the adaptor since start</td>
</tr>
<tr>
<td>prorated-flow</td>
<td>Display total flow of the adaptor since start</td>
</tr>
<tr>
<td>total-flow</td>
<td>Display total flow of the adaptor since start</td>
</tr>
</tbody>
</table>

**Verifying Collector Output**

Use the following command to look at the HDFS file system and verify that the Collector is binning and storing the data appropriately. By default, bins are stored in the /data/collector folder in the HDFS file system.

```
hadoop dfs -ls path-to-output-folder
```

For example:

```
hadoop dfs -ls /data/collector/1/output/edrflow/2012/10/09
```

The resulting output may resemble:

```
Found 19 items
drwrxr-x-x  - admin supergroup          0 2012-10-09 00:55
/data/collector/1/output/edrflow/2012/10/09/00
```

```
Drwrxr-x-x  - admin supergroup          0 2012-10-09 01:55
/data/collector/1/output/edrflow/2012/10/09/01
```
Emptying the Contents of a Collector Bin

You can use the validator commands to dump the contents of a bin after it has been processed by the collector. The exact command depends on the type of data, for example, `edr_validator`. You can perform the same command on both `bulkStats` and `edrhttp`.

```
[admin@host ~]# edr_validator -d
/data/collector/1/output/edrflow/2012/10/09/00/00 -p | head
```

These commands read from HDFS and write the output to the console. The resulting output may resemble:

```
filename: TESTLAB.EDRFLOW.1349740800.0
fileTime = 1349740800  StartTime = 0  EndTime = 0
```

Monitoring Oozie Jobs

Verifying the Status of Oozie Jobs from CLI

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

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

<table>
<thead>
<tr>
<th>Job ID</th>
<th>App Name</th>
<th>App Path</th>
<th>Console URL</th>
<th>User</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>06:15 2012-12-12 06:15</td>
<td>RUNNING 2012-12-12 06:15</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

... 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:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](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]# hadoop dfs -ls /data/output
```

The resulting output may resemble:

```
Found 13 items
drw-xr-x   - admin supergroup  0 2012-09-18 17:11
/data/output/AtlasBaseCubes
drw-xr-x   - admin supergroup  0 2012-09-18 17:14
```
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]# hadoop 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 dfsadmin 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 ~]# hadoop 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

```bash
[admin@host ~]# hadoop jar /opt/hadoop/hadoop-test-0.20.203.0.jar TestDFSIO -write -nrFiles 10 -filesize 1000
```

The resulting output may resemble:

```
TestDFSIO.0.0.4
12/10/09 20:18:37 INFO fs.TestDFSIO: nrFiles = 10
12/10/09 20:18:37 INFO fs.TestDFSIO: fileSize (MB) = 1
12/10/09 20:18:37 INFO fs.TestDFSIO: bufferSize = 1000000
12/10/09 20:18:37 INFO fs.TestDFSIO: creating control file: 1 mega bytes, 10 files
12/10/09 20:18:38 INFO fs.TestDFSIO: created control files for: 10 files
...
12/10/09 20:18:38 INFO mapred.FileInputFormat: Total input paths to process : 10
12/10/09 20:18:38 INFO mapred.JobClient: Running job: job_201209172118_2556
12/10/09 20:18:39 INFO mapred.JobClient:  map 0% reduce 0%
12/10/09 20:18:53 INFO mapred.JobClient:  map 100% reduce 0%
12/10/09 20:19:08 INFO mapred.JobClient:  map 100% reduce 100%
12/10/09 20:19:13 INFO mapred.JobClient: Job complete: job_201209172118_2556
...
```

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

```bash
[admin@host ~]# hadoop jar /opt/hadoop/hadoop-test-0.20.203.0.jar TestDFSIO -clean
```

The resulting output may resemble:

```
TestDFSIO.0.0.4
12/10/09 20:19:26 INFO fs.TestDFSIO: nrFiles = 1
12/10/09 20:19:26 INFO fs.TestDFSIO: fileSize (MB) = 1
```
Monitoring Processes on the Insta Nodes

The Insta database stores all the processed data, which is accessed by the UI nodes based on the queries received.

- Use the `show pm process` command to monitor the status of the database daemons. See "Viewing Process Status on Collector and Insta Nodes" on page 50.

- Use the `get-status-info` command to check all the processes that Insta uses:

```
admin@host (config)# insta infinidb get-status-info
```

The resulting output may resemble:

<table>
<thead>
<tr>
<th>Component</th>
<th>Status</th>
<th>Last Status Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>System</td>
<td>ACTIVE</td>
<td>Tue Oct 9 10:53:12</td>
</tr>
<tr>
<td>2012 Module pml</td>
<td>ACTIVE</td>
<td>Tue Oct 9 10:53:06</td>
</tr>
</tbody>
</table>

Calpont Process statuses

<table>
<thead>
<tr>
<th>Process Change</th>
<th>Module</th>
<th>Status</th>
<th>Last Status Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>ProcessMonitor</td>
<td>pml</td>
<td>ACTIVE</td>
<td>Tue Oct 9</td>
</tr>
<tr>
<td>10:52:43 2012</td>
<td>6158</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ProcessManager</td>
<td>pml</td>
<td>ACTIVE</td>
<td>Tue Oct 9</td>
</tr>
<tr>
<td>10:52:49 2012</td>
<td>6522</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBRMCControllerNode</td>
<td>pml</td>
<td>ACTIVE</td>
<td>Tue Oct 9</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Querying the Database Using the SQL Interface

The persistent storage provides an SQL interface, which you can access from the shell. Processed data is stored in the ucsdb database as shown below:

```
# idbmysql -u root -D ucsdb
mysql> show tables;
```

The resulting output may resemble:

```
<table>
<thead>
<tr>
<th>Tables_in_ucsdb</th>
</tr>
</thead>
<tbody>
<tr>
<td>bin_metatable</td>
</tr>
<tr>
<td>d_60min_all_dev</td>
</tr>
<tr>
<td>d_60min_apn_app</td>
</tr>
<tr>
<td>d_60min_apn_dev_appcat</td>
</tr>
<tr>
<td>d_60min_apn_dev_nhcat</td>
</tr>
<tr>
<td>d_60min_apn_dev_webcat</td>
</tr>
<tr>
<td>d_60min_apn_nh_app</td>
</tr>
<tr>
<td>d_60min_apn_rnc</td>
</tr>
<tr>
<td>d_60min_apn_seg_app</td>
</tr>
<tr>
<td>d_60min_apn_seg_nh_app</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>
```

410 rows in set (0.00 sec)
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 facing 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 hdfsNamnodeStatusTrap 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, or TACs 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.

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 122.

If there are certain user agents, URLs 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, 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.
- Uncategorized aggregator (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:

```bash
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# pmx subshell
subshell-name
```
2. Run:

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

Or,

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

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

2. Run:

```bash
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 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, and URLs; a total of 18 different reports.

**Note:** The daily uncategorized data reports are available only for the previous seven days, as per the current system time.

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

```bash
host [cluster : master|standby]> en
host [cluster : master|standby]# _shell
admin@host# hadoop 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:

```bash
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
```
2. Run:

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

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

Where:

- `report-name` is one of the following values:
  - `report_ct_bytes.map`
  - `report_ct_hits.map`
  - `report_tac_bytes.map`
  - `report_tac_hits.map`
  - `report ua_bytes.map`
  - `report ua_hits.map`
  - `report_url_bytes.map`
  - `report_url_hits.map`

- `time` is a date in the format `YYYY-MM-DDT00:00Z`. For example, `2013-07-07T00:00Z`.

**Examples of Reports**

This section show examples for each of the eighteen reports 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:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](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 2013-10-11T00:00Z
```

The resulting output may resemble:
Report Fetched Successfully
report_url_bytes.map For DataSet uncategorized_weekly_report and for Date 2013/10/11
---------------------------------------------------------------
<table>
<thead>
<tr>
<th>URL</th>
<th>HITS</th>
<th>TONNAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.wumpasworld.com">www.wumpasworld.com</a></td>
<td>1560</td>
<td>551711160</td>
</tr>
<tr>
<td>c2.cduniverse.ws</td>
<td>1578</td>
<td>730761174</td>
</tr>
<tr>
<td>widget.youappi.com</td>
<td>3154</td>
<td>1059928516</td>
</tr>
<tr>
<td><a href="http://www.thisis40movie.com">www.thisis40movie.com</a></td>
<td>1565</td>
<td>647650991</td>
</tr>
<tr>
<td><a href="http://www.morritastube.com">www.morritastube.com</a></td>
<td>1571</td>
<td>591687369</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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
report_url_hits.map For DataSet uncategorized_weekdaily_report and for Date 2013/10/10
---------------------------------------------------------------
<table>
<thead>
<tr>
<th>URL</th>
<th>HITS</th>
<th>TONNAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>c2.cduniverse.ws</td>
<td>2842</td>
<td>1316020400</td>
</tr>
<tr>
<td>widget.youappi.com</td>
<td>5678</td>
<td>1908140283</td>
</tr>
<tr>
<td><a href="http://www.morritastube.com">www.morritastube.com</a></td>
<td>2829</td>
<td>1065407349</td>
</tr>
<tr>
<td>meet24.com</td>
<td>9959</td>
<td>2412706289</td>
</tr>
<tr>
<td>slots.sharkparty.com</td>
<td>11353</td>
<td>5231791141</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

UANUMBER Dortage
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

UANUMBER Dortage
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
...

Copyright © 2014, Cisco Systems, Inc.
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

<table>
<thead>
<tr>
<th>TAC</th>
<th>HITS</th>
<th>TONNAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>350176</td>
<td>10706085</td>
<td>3746556562178.00</td>
</tr>
<tr>
<td>351308</td>
<td>10753822</td>
<td>3808925836088.00</td>
</tr>
<tr>
<td>351002</td>
<td>10768013</td>
<td>3789770208197.00</td>
</tr>
<tr>
<td>1216200</td>
<td>10715887</td>
<td>3755896252546.00</td>
</tr>
<tr>
<td>520250</td>
<td>10688524</td>
<td>3738700196803.00</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
```

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

<table>
<thead>
<tr>
<th>TAC</th>
<th>HITS</th>
<th>TONNAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>350176</td>
<td>5951243</td>
<td>2082657141356.00</td>
</tr>
<tr>
<td>351308</td>
<td>5978177</td>
<td>2117437827993.00</td>
</tr>
<tr>
<td>351002</td>
<td>5985072</td>
<td>2106367056306.00</td>
</tr>
<tr>
<td>1216200</td>
<td>5956928</td>
<td>2087884304997.00</td>
</tr>
<tr>
<td>520250</td>
<td>5941658</td>
<td>2078276199339.00</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
```

Copyright © 2014, Cisco Systems, Inc.
Uncategorized Device Heavy Hitters by Tonnage

```
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
Operas 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

```
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
Operas 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:

<table>
<thead>
<tr>
<th>SP</th>
<th>IS_WHITELIST_MEMBER</th>
<th>HITS</th>
<th>TONNAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>report_sp_bytes.map</td>
<td>1</td>
<td>0</td>
<td>18923864722</td>
</tr>
<tr>
<td>total</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>SP</th>
<th>IS_WHITELIST_MEMBER</th>
<th>HITS</th>
<th>TONNAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>report_sp_hits.map</td>
<td>1</td>
<td>0</td>
<td>18923864722</td>
</tr>
<tr>
<td>total</td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Copyright © 2014, Cisco Systems, Inc.
### Uncategorised 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:

<table>
<thead>
<tr>
<th>APP</th>
<th>IS_WHITELIST_MEMBER(1)</th>
<th>HITS</th>
<th>TONNAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>report_mobile_app_bytes.map</td>
<td>For DataSet uncategorized_weekly_report and for Date 2013/10/11</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
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
...```

### Uncategorised 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:

<table>
<thead>
<tr>
<th>APP</th>
<th>IS_WHITELIST_MEMBER(1)</th>
<th>HITS</th>
<th>TONNAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>report_mobile_app_hits.map</td>
<td>For DataSet uncategorized_weekly_report and for Date 2013/10/11</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
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

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

The resulting output may resemble:

<table>
<thead>
<tr>
<th>MIME</th>
<th>HITS</th>
<th>TONNAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>report_uncat_ct_bytes.map For DataSet uncategorized_weekly_report and for Date 2013/10/11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>audio/amr-wb+</td>
<td>643753</td>
<td>234771172339.00</td>
</tr>
<tr>
<td>audio/EVRCB</td>
<td>650552</td>
<td>223021173784.00</td>
</tr>
<tr>
<td>video/vnd.dece.sd</td>
<td>638886</td>
<td>226238961590.00</td>
</tr>
<tr>
<td>application/vnd.smaf</td>
<td>614370</td>
<td>217668901502.00</td>
</tr>
<tr>
<td>application/vnd.rainstor.data</td>
<td>604876</td>
<td>214049781874.00</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Uncategorized Content Type Heavy Hitters by Hits

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

The resulting output may resemble:

<table>
<thead>
<tr>
<th>MIME</th>
<th>HITS</th>
<th>TONNAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>report_uncat_ct_hits.map For DataSet uncategorized_weekly_report and for Date 2013/10/10</td>
<td></td>
<td></td>
</tr>
<tr>
<td>audio/amr-wb+</td>
<td>1159276</td>
<td>422773142078.00</td>
</tr>
<tr>
<td>audio/EVRCB</td>
<td>1171468</td>
<td>401603258676.00</td>
</tr>
<tr>
<td>video/vnd.dece.sd</td>
<td>1150489</td>
<td>407407640052.00</td>
</tr>
<tr>
<td>application/vnd.smaf</td>
<td>1106342</td>
<td>391972307777.00</td>
</tr>
<tr>
<td>application/vnd.rainstor.data</td>
<td>1089236</td>
<td>385449436723.00</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Uncategorized Ports Heavy Hitters by Tonnage

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

The resulting output may resemble:

```
Report Fetched Successfully

<table>
<thead>
<tr>
<th>PORT</th>
<th>HITS</th>
<th>TONNAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1304</td>
<td>3390432</td>
<td>1204814403192.00</td>
</tr>
<tr>
<td>1301</td>
<td>3378333</td>
<td>1202639295828.00</td>
</tr>
<tr>
<td>666</td>
<td>3401119</td>
<td>1175700715925.00</td>
</tr>
<tr>
<td>132</td>
<td>3319717</td>
<td>1173322825895.00</td>
</tr>
<tr>
<td>23</td>
<td>3429917</td>
<td>1199045015196.00</td>
</tr>
</tbody>
</table>

...```

### Uncategorized Ports Heavy Hitters by Hits

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

The resulting output may resemble:

```
Report Fetched Successfully

<table>
<thead>
<tr>
<th>PORT</th>
<th>HITS</th>
<th>TONNAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1304</td>
<td>3390432</td>
<td>1204814403192.00</td>
</tr>
<tr>
<td>1301</td>
<td>3378333</td>
<td>1202639295828.00</td>
</tr>
<tr>
<td>666</td>
<td>3401119</td>
<td>1175700715925.00</td>
</tr>
<tr>
<td>132</td>
<td>3319717</td>
<td>1173322825895.00</td>
</tr>
<tr>
<td>23</td>
<td>3429917</td>
<td>1199045015196.00</td>
</tr>
</tbody>
</table>

...```
Exporting Uncategorized Data Reports

The eighteen 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 -lrth 1381363200*

[admin@Rubix-GMS-1 uncat_reports]# ls -lrth 1381363200*
[admin@Rubix-GMS-1 uncat_reports]# ls -lrth 1381363200*
```

**Note:** 1381363200 is the epoch time of 10 October 2013 on which these reports were generated.

The resulting output may resemble:

```
-rw-r--- 1 admin root 0 Oct 10 06:26 1381363200.report_ct_bytes.map
-rw-r--- 1 admin root 0 Oct 10 06:26 1381363200.report_ct_hits.map
-rw-r--- 1 admin root 14K Oct 10 06:26 1381363200.report_device_bytes.map
-rw-r--- 1 admin root 14K Oct 10 06:26 1381363200.report_device_hits.map
-rw-r--- 1 admin root 8.3K Oct 10 06:26 1381363200.report_mobile_app_bytes.map
...
```
Stopping Automatic Export of Reports

To stop the OfflineEngineUncatReports:

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

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

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

If any of these steps fail, do not proceed. Instead, contact Cisco Technical Support to resolve the issue.

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.

Applying Collector Node Configurations

Create Directories to Receive Input

If not already created, this section will provide steps to create directories on Collector nodes which receive input data files from the new ASR gateway.

1. Log into the master Collector node and run:

   **Note:** ASR creates a subdirectory such as gateway-edr-file-path on the master Collector as an absolute path for eg /data/collector/edr1/edr. Another ASR path to be set up is /data/collector/edr1.

   ```
   host [cluster : master]> en
   host [cluster : master]# _shell
   admin@host# mkdir -p /data/collector/gateway-edr-file-path
   admin@host# mkdir -p /data/collector/gateway-bulkstats-file-path
   ```

For example,
Add Configurations to the Collector

This section describes how to add a new ASR gateway to the Collector.

1. Determine the gateway count, \textit{gwCount}, in the Collector configuration.

   Log into the master Collector node and run:

   \begin{verbatim}
   host [cluster : master] > en
   host [cluster : master]# _shell
   admin@host# mkdir -p /data/collector/California/edr111/
   admin@host# mkdir -p /data/collector/California/bs111/
   \end{verbatim}

   2. Repeat step 1 on the standby Collector node.

   The output of the above command shows a value that specifies the number of ASR gateways configured in the system. In the above example, the value is two.

   In this procedure, we will add a third gateway, making the \textit{gwCount} = 3.

   \textbf{Note:} If the output of the above command returns nothing, no bulkstats gateway is configured on the Collector. In this case \textit{gwCount} = 1.

2. Log into the master Collector node and run the following commands to add a new ASR configuration to the Collector:

   \begin{verbatim}
   host [cluster : master] > en
   host [cluster : master]# conf t
   host [cluster : master](config)# collector modify-instance 1
   modify-adaptor bulkStats add-file-if bulkStatsFile<gwCount>
   host [cluster : master](config)# collector modify-instance 1
   modify-adaptor bulkStats modify-file-if bulkStatsFile<gwCount>
   \end{verbatim}
filename-format bs-filename-pattern
host [cluster : master](config)# collector modify-instance 1
modify-adaptor bulkStats modify-file-if bulkStatsFile<gwCount>
sync-wait-period 60
host [cluster : master](config)# collector modify-instance 1
modify-adaptor bulkStats modify-file-if bulkStatsFile<gwCount>
compression none
host [cluster : master](config)# collector modify-instance 1
modify-adaptor bulkStats modify-file-if bulkStatsFile<gwCount>
backup-directory /data/collector/bulkstats_files_backup/
gateway-name/
host [cluster : master](config)# collector modify-instance 1
modify-adaptor bulkStats modify-file-if bulkStatsFile<gwCount>
input-directory /data/collector/bulkstats_files/gateway-name/
host [cluster : master](config)# collector modify-instance 1
modify-adaptor bulkstats modify-file-if bulkStatsFile<gwCount>
backup-file-expiry-period 168

Where:

- **gwCount** is the new gateway count determined in previous step.
- **bs-filename-format** is the format the system shall follow when naming the file.
- **gateway-name** is the name of the gateway which would be visible on UI for the bulkstats data that would be sent from this gateway.

For example,

- **bulkStatsFile<gwCount> = bulkStatsFile3**
- **bs-filename-format = *%YYYY%MM%DD%hh%mm%ss**
- **gateway-name = GMPLAB1**

3. Repeat step 2 on the standby Collector node.

4. Verify that the filename format for the new gateway EDR is the same as the existing EDR file name format configured on the master Collector. Existing filename formats configured on the collector can be printed using these
commands:

```
cli -t "en" "conf t" "show runn full" | grep edrflow | grep filename-format
cli -t "en" "conf t" "show runn full" | grep edrhttp | grep filename-format
```

Alternatively, you can view formats on an existing ASR.

**Update Information Bases**

Once the above mentioned variable are present and collector changes are applied successfully, then update the IOs for the new gateway on master Collector node.

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

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](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
   type gateway-type
   edr-file-path gateway-edr-file-path
   bulkstat-file-path incoming-BS-files-path-on-collector
```

**Note:** See "Modifying Gateway Attributes" on page 113 for more information on gateway attributes and a sample of the output generated by this command.
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
bulkstat-file-path /California/bs111
```

- 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
```

Log into the standby Collector node and fetch IBs from the pmx subshell inbox for both EDR and bulkstats:

- Log into the standby Collector node and go the pmx aggregation_center subshell to pull EDR IBs:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# pmx subshell
subshell-name
```

- Fetch all IBs from inbox:

```
pm extension (subshell-name)> fetch all ibs from inbox
```
Switch to the pmx bulkstats subshell and repeat the last step to fetch bulk-stats IBs.

**Restarting Processes**

Restart the Collector process on both the master and on the standby Collector nodes:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# write memory
host [cluster : master|standby](config)# pm process collector restart
```

**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 a UI node to the existing UI/Cube (Rubix) cluster. In the MURAL system environment, the UI engine is hosted on the same server as the Cube (Rubix) engine; therefore the node is sometimes interchangeably referred to as either UI or Cube node.

Before You Begin

The new node which is being added should be on 3.3 release with all the relevant patches.

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: atlas3.5.rc1
```

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
(+) atlas3.5.rc1.p3 2014-04-18 12:30:03 +0000
(+) atlas3.5.rc1.p2 2014-04-18 12:29:17 +0000
(+) atlas3.5.rc1.p1 2014-04-18 12:29:12 +0000
```

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 `mural.xml` to the GMS UI and edit it with the
details of the new UI node.

**Note:** The xml file `mural.xml` 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-guavus-x86_64-20140315-133212.iso
host [cluster : master|standby](config)# image mount mfgcd-guavus-x86_64-20140315-133212.iso

The resulting output may resemble:

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

**Note:** The `mural.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 3.3.

**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:
**MURAL Operations and Troubleshooting Guide**

```bash
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):

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

The resulting output may resemble:

```
Node successfully installed.
```

4. Add any patches to the UI nodes that have been released for this build. See the *MURAL Release Notes* for a list of patches and installation instructions.

**Pushing IBs and Certificates to New Node**

1. Log into the master Collector node and go to the aggregation_center sub-shell:

```bash
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# pmx subshell
subshell-name
```

2. Push IBs to the new UI node:

```bash
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:

```bash
pm extension (aggregation_center)> quit
pm extension (bulkstats)> quit
```
4. If single certificate files were installed on the UI nodes during the original system setup, restore these files on the new UI node.

5. Log into the master UI node and go to `_shell`:

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

6. Copy the keystore to the new UI node:

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

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

**Edit the Tomcat Files on Master UI Node**

1. Log into the master UI node.

2. Edit the EDR tomcat file: `/data/apache-tomcat/apache-tomcat-7.0.27/-conf/server.xml`

   **Search for Interceptor className and replace the existing interceptor tag line with the following:**

```
<Interceptor className="com.guavus.tomcat.interceptor.DisableMulticastInterceptor"/>
<Interceptor className="org.apache.catalina.tribes.group.interceptors.TcpFailureDetector"/>
<Interceptor className="org.apache.catalina.tribes.group.interceptors.StaticMembershipInterceptor">
<Member className="org.apache.catalina.tribes.membership.StaticMember" port="4001" host="Control_IP_UI_1"
uniqueId="{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0}"/>
<Member className="org.apache.catalina.tribes.membership.StaticMember" port="4002" host="Control_IP_UI_1"
uniqueId="{1,0,0,0,0,0,0,0,0,0,0,0,0,0,0}"/>
```
<Member className="org.apache.catalina.tribes.membership.StaticMember" port="4000" host="Control_IP_UI_2" uniqueId="(1,0,0,0,0,0,0,0,0,0,0,0)"/>
<Member className="org.apache.catalina.tribes.membership.StaticMember" port="4001" host="Control_IP_UI_2" uniqueId="(1,0,0,0,0,0,0,0,0,0,0,0)"/>
<Member className="org.apache.catalina.tribes.membership.StaticMember" port="4000" host="Control_IP_UI_3" uniqueId="(1,0,0,0,0,0,0,0,0,0,0,0)"/>
<Member className="org.apache.catalina.tribes.membership.StaticMember" port="4001" host="Control_IP_UI_3" uniqueId="(1,0,0,0,0,0,0,0,0,0,0,0)"/>
</Interceptor>

Where:

- Control_IP_UI_1 is the control IP address of master UI node
- Control_IP_UI_2 is the control IP address of standby UI node
- Control_IP_UI_3 is the control IP address of normal (newly added) UI node

3. If editing files on the master UI node, repeat the previous step by replacing the Interceptor className line on the:

   - bulkstats tomcat file: /data/apache-tomcat-bulkstats/apache-tomcat-7.0.27/conf/server.xml
   - RGE tomcat file: /data/rge/apache-tomcat/apache-tomcat-7.0.27/conf/server.xml
   - second tomcat file: /data/apache-tomcat2/apache-tomcat-7.0.27/conf/server.xml

4. Log into the standby normal (newly added) Rubix node.

5. If editing files on the standby UI node, repeat step 2 by replacing the Interceptor className line on the:
• **first tomcat file**: /data/apache-tomcat/apache-tomcat-7.0.27/conf/server.xml

• **second tomcat file**: /data/apache-tomcat2/apache-tomcat-7.0.27/conf/server.xml

### 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 all 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
   - launcher

   **Note:** Wait for two minutes between starting each process.

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

4. Log into the newly added third UI node and repeat the above steps to restart the tomcat instances.
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:** mural.xml 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:
   - Under Slot, click **Add** and provide the details for the compute node (host name, RAC IP address, RAC user, and RAC password).
   - 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
```

   - 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.

e. Under **Clusters**, add the new compute 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 `mural.xml` 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 3.4.

9. 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-CLU
```
10. Monitor the installation status on the Compute node by executing this command to show the percentage completed:

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

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

When complete, the output shows the message:

```
Node successfully installed
```

11. Add any patches to the Compute nodes that have been released for this build. See the MURAL Release Notes for a list of patches and installation instructions.

### Checking the New Compute Node

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

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

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

The resulting output may resemble:

```
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

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

**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 | grep -i "datanode|_tasktracker" | grep -v grep | awk '{print $NF}'
org.apache.hadoop.hdfs.server.datanode.DataNode
org.apache.hadoop.mapred.TaskTracker
# mount | grep hadoop-admin
/dev/mapper/mpathep1 on /data/hadoop-admin type ext3
(rw,relatime,errors=continue,barrier=1,data=ordered)
```
Adding Postdated Data to the Collector Using Replay Mode

This topic describes how to replay data in a fresh MURAL system on Collector nodes for testing purposes.

**Note:** Incorporating older data in replay mode means that all calculations must be rerun. This is generally too resource-intensive to be worthwhile in anything other than a test environment.

Replay mode is used to add timestamped data to the Collector from the past. When ready to process live data in a production environment, the collector must be switched from a replay to live mode. In live mode, postdated data that comes in to the Collector is discarded and incoming current data goes into the current 5-minute bin. For more information, see "Understanding Data Bins" on page 21.

**Before You Begin**

Install and configure the system following these instructions taken from the *MURAL Installation Guide* to disable job timeout and collector bin sliding:

1. Download the `revert_timeOut_binSliding.txt` file to local machine.
2. SCP the `revert_timeOut_binSliding.txt` file to the master and standby collector nodes under `/data` directory.

   **Note:** This file can be obtained from Cisco Technical Support.

3. Run the file to disable timeout for all jobs and bin sliding in collector:

   a. Log into the master Collector node and configure the timeout for the jobs on the master node:

   ```
   host [cluster : master]> en
   host [cluster : master]# _shell
   admin@host# cli /data/revert_timeOut_binSliding.txt
   ```

   b. Repeat commands above on the standby Collector node.
Configuring Collector in Replay Mode

1. From CLI, run the following commands to configure the Collector nodes to run in replay mode:

   ```
   host [cluster : master]> en
   host [cluster : master]# conf t
   host [cluster : master](config)# pm process collector launch
   params 1 -r
   host [cluster : master](config)# write memory
   host [cluster : master](config)# pm process collector restart
   ```

2. Send input files to the Collector for both EDR and bulkstats for the ASR which was configured as part of the MURAL Installation Guide.

3. Validate data according to the MURAL Installation Guide prior to going onto the next step. Do not go on until you have verified that data is available in Hadoop.

4. Set jobs time using the `setOozieTime` script.

   **Using setOozieTime script**

   Execute the script to set the data start times to the start of the time for which which EDR and bulkstats data is being fed into the system and is available in Hadoop.

   1. Log into the GMS.
   2. Go to directory `/opt/deployment/Mural_setStartTime`.
   3. Run the following command for both master and standby NameNode:

   ```
   ./setOozieTime --dataStartTime <data_start_time> --node <Collector_IP> --password <Admin-password> --manual --verbose
   ```

   For example:
5. Log into the master node and go to the pmx oozie subshell:

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   host [cluster : master|standby](config)# pmx subshell
   subshell-name
   ```

1. Run all mapreduce jobs:

   ```
   pm extension (oozie)> run job CoreJob
   pm extension (oozie)> run job EDR
   pm extension (oozie)> run job BulkStat
   ```

2. Wait 3 hours and start the bulkstat aggregation job:

   ```
   pm extension (oozie)> run job BSAgg15min
   ```

3. Wait 1 hour and stop each mapreduce job:

   ```
   pm extension (oozie)> stop jobname CoreJob
   pm extension (oozie)> stop jobname EDR
   pm extension (oozie)> stop jobname BulkStat
   pm extension (oozie)> stop jobname BSAgg15min
   ```

4. Run all mapreduce jobs:

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

   **Warning:** Because we used `run all jobs`, cleanup jobs have started. We need to stop the cleanup jobs as quickly as possible because we are trying to process older data and cleanup jobs may delete that data before it is processed.

5. To get a complete list of cleanup jobs currently running, use the
following command while still in _shell mode on the master collector:

```
# cli -t "en" "show run full" | grep "Cleanup" | awk -F'/ ' '{print $6}' | sort -u | grep -v "value"
```

6. Go to the pmx oozie subshell and stop each job listed by the above command which may include but not be limited to:

- # stop jobname Cleanup15minOutput
- # stop jobname Cleanup1DayOutput
- # stop jobname Cleanup1WeekOutput
- # stop jobname Cleanup1hrOutput
- # stop jobname Cleanup4WeekOutput
- # stop jobname Cleanup5minOutput
- # stop jobname CleanupAtlas
- # stop jobname CleanupCollector
- # stop jobname CleanupLogs
- # stop jobname CleanupRawRecords
- # stop jobname CleanupSegment
- # stop jobname CleanupSubcrBytes
- # stop jobname CleanupTBA4Weekly
- # stop jobname CleanupTBAWeekly
- # stop jobname CleanupTethering
- # stop jobname CleanupTopSubcr

7. Wait 2 hours to get Insta data pushed.

6. Check disk space frequently as data is being processed:

We cannot continuously feed data to the system because we cannot run cleanup jobs in replay mode. As a result, the hdfs disk is likely to get full if we feed unlimited data to the system.

a. Run `hadoop dfsadmin -report` on the master Collector node.

b. Ensure that the value for **DFS Used%** is less than 70%.
c. If DFS Used% is higher than 70%, some data cleanup is required and you need to contact Technical Support.

Note: Follow procedure "Switching from Replay to Live Mode to Receive Live Data" on page 139 once the customer has confirmed post data has been processed successfully.
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_64-20140315-133212.iso`

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

   ```
   # md5sum /data/mfgcd-guavus-x86_64-20140315-133212.iso
   48fc39f77bc7ab8847ca2f7a684d2ace /data/mfgcd-guavus-x86_64-20140315-133212.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. At the **#** prompt, run the **manufacture** command to manufacture the master GMS blade.

    The following command is appropriate for a single disk configured as RAID 1, as indicated by the **-L 1D** argument. The master GMS 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/init.d/rcS.d/S34automf
- 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 /dev/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 0.0ZMiB 30000MiB 30000MiB ext3 primary

Writing partition table to DISICZ
Disk devsdb: S7Z3Z6MiB
Sector size (logicalphysical): 512B,512B
Partition Table: gpt
Number Start End Size File system Mame Flags
1 0.0ZMiB 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 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.
#

11. 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.

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

    # reboot

13. Use SSH to log in to the master GMS blade as user **admin**.
Setting Up the Master GMS Node

After the master GMS blade is manufactured as described in "Manually Manufacturing the Master GMS Blade" on page 104, you need to set the administrator password for the node, assign IP addresses, and verify correct zoning on the fabric interconnect.

Setting the Password and IP Addresses

To set the administrator password and IP addresses on the master GMS node, perform the following steps:

1. Log in to the master GMS node via the console as **admin**, if you have not already.

2. Enter config mode and set the password for the **admin** user. We recommend setting the password to **admin@123**, which is the standard value that Cisco Technical Support expects to use when they log in to a blade to help you with a problem.

   ```
   > en
   # conf t
   (config)# username admin password admin@123
   (config)# write mem
   (config)# exit
   ```

3. Enter config mode again and define the IP addresses for the management interface and default gateway.

   ```
   > en
   # conf t
   (config)# interface mgmt-interface ip-address mgmt-IP-of-GMS-server subnetmask-of-mgmt-net
   (config)# ip-default-gateway mgmt-network-default-gateway-IP
   (config)# license install LK2-RESTRICTED_CMDS-88A4-FNLG-XCAU-U
   (config)# write memory
   (config)# _shell
   ```

4. Apply all patches on GMS node using the below procedure:
a. Download patches released with the build in the /data/tmp_patch directory. (directory needs to be created manually)

b. Download the scripts (expect_shell.sh and patch_link_install.sh) to /data directory.

c. Apply the patches on GMS server by running:

```
[admin@GMS data] # chmod 755 expect_shell.sh patch_link_install.sh
[admin@GMS tmp_patch]# ./patch_link_install.sh self
```

d. Check the status of patches applied by the script in step above:

```
> en
# conf t
(config)# pmx subshell patch
pm extension (patch)> show patch_history
(+) atlas3.5.rc1.p3 2014-04-18 12:30:03 +0000
(+) atlas3.5.rc1.p2 2014-04-18 12:29:17 +0000
(+) atlas3.5.rc1.p1 2014-04-18 12:29:12 +0000
```

5. Start the GMS Server:

```
> en
# conf t
(config)# pm process gms_server launch auto
(config)# pmx register pgsql
(config)# write memory
(config)# pm process tps restart
(config)# pm process gms_server restart
```

6. Check the status of the GMS Server by running the following command:

```
# _shell
# cli -t "en" "config t" "show pm process gms_server" | grep "Current status"
Current status: running
```
Verifying Zoning and FLOGI on the Fabric Interconnect

To verify zoning and the fabric login (FLOGI) on the fabric interconnect (FI), perform the following steps:

1. Use SSH to log in to the FI.
2. Run the `connect nxos A` command to connect to the FI.
3. Run the `show zoneset active` command and verify that its output reports the fiber channel ID (FCID) for all world wide port names (WWPNs) and hosts, as shown for FI A and FI B in the following example.

**Note:** In the following output, the identifiers are examples only and are different in your deployment. Also, the term `pwwn` in the output refers to WWPNs.

```
hostname-A(nxos)# show zoneset active
zone name ucs_hostname_A_10_GMS1_vHBA-A vsan 3010
* fcid 0x100005 [pwwn 20:00:00:05:ad:1e:11:1f]
* fcid 0x1000ef [pwwn 50:06:01:60:3e:a0:28:d2]
* fcid 0x1001ef [pwwn 50:06:01:69:3e:a0:28:d2]

hostname-B(nxos)# show zoneset active
zone name ucs_hostname_B_22_GMS1_vHBA-B vsan 3020
* fcid 0x420001 [pwwn 20:00:00:05:ad:1e:1:4e]
* fcid 0x4200ef [pwwn 50:06:01:61:3e:a0:28:d2]
* fcid 0x4201ef [pwwn 50:06:01:68:3e:a0:28:d2]
```
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:

   ```bash
   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   host [cluster : master|standby](config)# pmx subshell
   subshell-name
   ```

2. Add a gateway by running:

   ```bash
   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
   ```
**MURAL Operations and Troubleshooting Guide**

```
<table>
<thead>
<tr>
<th>edr-filename-pattern incoming-EDR-fileName-pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>bulkstat-filename-pattern incoming-BS-fileName-pattern</td>
</tr>
<tr>
<td>type gateway-type</td>
</tr>
<tr>
<td>edr-file-path gateway-edr-file-path</td>
</tr>
<tr>
<td>bulkstat-file-path incoming-BS-files-path-on-collector</td>
</tr>
</tbody>
</table>
```

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:

```bash
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_ASM5K.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:
Defining Top N Parameters

The user interface displays various lists of entities, such as subscribers, devices, or content categories, displaying the remainder beyond the top N aggregated together under a category for Other. You can change the setting for N for the following types of entities:

<table>
<thead>
<tr>
<th>Entity</th>
<th>Name in script</th>
<th>Default Setting</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subscribers</td>
<td>TopSubscribers</td>
<td>1000</td>
<td>1-1000</td>
</tr>
<tr>
<td>Uncategorized URLs</td>
<td>topUnCatUrls</td>
<td>1000</td>
<td>1-1000</td>
</tr>
<tr>
<td>Uncategorized UAs</td>
<td>topUnCatUAs</td>
<td>1000</td>
<td>1-1000</td>
</tr>
<tr>
<td>Uncategorized TACs</td>
<td>topUnCatTACs</td>
<td>1000</td>
<td>1-1000</td>
</tr>
</tbody>
</table>

**Note:** Contact Cisco Technical Support to determine what values to set for your environment.

To set the Top N parameters for these entities, you need to run a script called TopN.py on the GMS which updates the N values on both the master and standby Collector nodes. TopN.py can be obtained from Cisco Technical Support.

If any of these steps fail, do not proceed. Instead, contact Cisco Technical Support to resolve the issue.

**Note:** Changing the top N parameters causes the hourly data processing job (EDR) and the top subscribers calculation jobs to be restarted.

To define the value for N in the top N lists:

1. Download the TopN.py script from the new GMS.

2. Ensure that the NameNode is reachable from GMS with the username and password provided in the script.

3. Use SSH to connect to the GMS using the management IP address.

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

4. To transfer the TopN.py script file to the GMS, change your drive to this
location:

```
admin@host# cd /opt/tps/scripts
```

5. From the GMS, and run the TopN.py script for the master Collector node:

```
admin@host# python2.7 TopN.py Collector-IP username password
```

Where you provide the management ip address of the master Collector node and your login credentials for that node or host.

For example:

```
admin@host# python2.7 TopN.py 192.168.10.9 admin admin@123
```

6. The resulting output generates prompts with questions for you to answer. Provide the values you want to set.

For instance, if you enter a value greater than the maximum value, the script uses the maximum value.

The resulting output may resemble:

```
The script will cause jobs to rerun on Master Collector..Do you want to continue (Yes / No):- Yes
Specify the type of node it is (Master / Standby) :- Master
========== TOPN Values to be set in 192.168.151.171 ==========
STOPPING EDR JOB BEFORE SETTING TOPN VALUES
EDR JOB STOPPED SUCCESSFULLY
STOPPING TopSubcr JOB
TopSubcr JOB STOPPED SUCCESSFULLY
STOPPING ConsistentSubcrs JOB
ConsistentSubcrs JOB STOPPED SUCCESSFULLY
Enter the topUnCatUAs value to be set[1-1000]:- 100
Value to be set is 100
Enter the topUnCatUrls value to be set[1-1000]:- 500
Value to be set is 500
```
Enter the topUnCatTACs value to be set [1-1000]: – 40
Value to be set is 40
Enter the TopSubscribers value to be set [1-1000]: – 500
Value to be set is 500
STARTING EDR JOB
EDR JOB STARTED
STARTING TopSubcr JOB
TopSubcr JOB STARTED
Starting ConsistentTopSubcr JOB
ConsistentTopSubcr JOB STARTED
#

7. From the GMS, re-run the TopN.py script for standby Collector node.
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), the application used on the device to pull the data (based on user agent), or the service provider used from the device. Use the same procedure to add categories for entities that the system is currently classifying as uncategorized.

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 information made available in the next release or patch release. You might want to override the information base and change classifications in order to:

- Add a new category—for example, a recently released device is not yet included in the information base. Or, reduce the number of records that are sent to the uncategorized bucket. See "Backing Up and Restoring a Large (TB) Insta Database" on page 170.

- Make the categories broader or narrower—for example, to collapse all iPhone categories into one category that includes all types of iPhones.

If user agents and TACs 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 the user agents and TACs 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

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

UA Device Model Overriding :
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
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

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 66.

Overriding Categorizations from Information Bases

To override or add category mappings in information bases:

1. Log into the master Collector node (NameNode).
2. 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.

```plaintext
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
```

3. Go to the `pmx` aggregation_center subshell:

```plaintext
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# pmx subshell subshell-name
```

4. Edit the information base:

```plaintext
pm extension (aggregation_center)> edit ib information-base-name.map add relevant-field-names
```

Where `information-base-name.map` and `relevant-field-names` follow the examples in the following sections.

5. Generate all information bases:

```plaintext
pm extension (aggregation_center)> generate all ibs
```

6. Push all information bases:

```plaintext
pm extension (aggregation_center)> push all ibs
```

7. Synchronize the information bases on the standby Collector node with the master:

   a. Log into the standby Collector node and go to the `aggregation_center` subshell:

```plaintext
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
```
b. Fetch the information bases from inbox:

```
pm extension (aggregation_center)> fetch all ibs from inbox
```

You do not need to restart jobs because the system will pick up the updated information bases the next time the job runs (for the next hour's data).

**Note:** The following examples are examples of steps to edit information bases and must be followed by the `generate` and `push all ibs` steps.

**Example: Changing Categorization of Mobile Applications by User Agent**

To override default category of a mobile application based on a user agent, edit the `ua_mobile_app.map` information base as shown below:

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

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

2. Run the `edit ib` command as shown:

```
pm extension (aggregation_center)> edit ib ua_mobile_app.map
add
User Agent: Microsoft BITS/7.5
Mobile Application Name: BITS
Mobile Application Type: Non Browser
Mobile Application Category: Productivity
```

In this example, the default mobile application that is mapped to the Microsoft BITS/7.5 user agent was **Business**, and we are overriding this default category with the category **Productivity**.
The resulting output may resemble:

```
updating mobileappname.list
updating mobileappcategory.list
```

3. To verify the update was made, run the `show ib` command and look for the entry for Microsoft BITS/7.5:

```
pm extension (aggregation_center)> show ib ua_mobile_app.map
1 [Microsoft BITS/7.5][BITS][Non Browser][Productivity]
```

**Example: Changing Categorization of Mobile Devices by User Agents**

To override default category of a mobile device based on the user agent, edit the `ua_manuf_model.map` information base:

1. Log in and go to the `aggregation_center` subshell:

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

2. Run the `edit ib` command as shown:

```
pm extension (aggregation_center)> edit ib ua_manuf_model.map
addUser Agent: 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
Manufacturer: LG
Model: Gelato
```

The default device model that is mapped to this user agent is LG VS700 Gelato Q. We are overriding this device name with the custom one, Gelato.

The resulting output may resemble:
3. To verify if the update was made, run the `show ib` command and look for the entry for the user agent:

```plaintext
pm extension (aggregation_center)> show ib ua_manuf_model.map
1 [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][LG][Gelato]
```

**Example: Changing Categorization of Mobile Devices by TAC Identifier**

To override default category of a mobile device based on its TAC identifier, edit the `tac_manuf_model.map` information base:

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

   ```plaintext
   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   host [cluster : master|standby](config)# pmx subshell
   subshell-name
   ```

2. Run the `edit ib` command as shown:

   ```plaintext
   pm extension (aggregation_center)> edit ib tac_manuf_model.map
   add
   TAC: 98005800
   Manufacturer: BlackBerry
   Model: BB Bold
   ```

   In this example, the default device manufacturer for TAC 98005800 is RIM and the model is RIM BlackBerry 9650 Bold Tour 2. We are overriding the device manufacturer with the value BlackBerry, and overriding the device model name with BB Bold.

   The resulting output may resemble:
3. To verify the update was made, run the `show ib` command and look for the entry for TAC 98005800:

```
pm extension (aggregation_center)> show ib tac_manuf_model.map
1 [98005800][BlackBerry][BB Bold]
```

**Example: Changing Categorization of a Service Provider**

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

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

1. **Login and go to the aggregation_center subshell:**

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# pmx subshell
    subshell-name
```

2. **Run the `edit ib` command as shown:**

```
pm extension (aggregation_center)> edit ib sp_url_cat.map add
Service Provider: craigslist.org          URL Category: Shopping
```

The resulting output may resemble:

```
updating category.list
```

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

```
pm extension (aggregation_center)> show ib sp_url_cat.map
1 [craigslist.org][Shopping]
```
**Note:** You must generate and push all ibs on the master node. Follow procedures regarding fetch all ib from inbox on the collector as mentioned in "Backing Up and Restoring Information Bases" on page 181.
Configuring ASR Settings for MURAL

This topic describes the fields which must be configured on the ASR as output to MURAL platform and the settings that are supported by the MURAL application. It does not include information about how to configure the output fields on the ASR.

Configuring EDR Flow Field Settings

An updated example is now at the end of MURAL Installation Guide.

Configuring HTTP Field Settings

The following commands specify the ASR settings required for the output to MURAL of HTTP flow fields:

```
configure
active-charging service ecs-svcl
edr-format edr_http_format
attribute sn-start-time format seconds priority 10
attribute sn-end-time format seconds priority 20
attribute radius-calling-station-id priority 30
attribute radius-called-station-id priority 40
rule-variable ip server-ip-address priority 50
rule-variable http host priority 70
rule-variable http url priority 71
rule-variable http content type priority 80
attribute transaction-downlink-bytes priority 90
attribute transaction-uplink-bytes priority 100
attribute transaction-downlink-packets priority 110
attribute transaction-uplink-packets priority 120
rule-variable http user-agent priority 130
rule-variable bearer 3gpp charging-id priority 131
rule-variable bearer 3gpp rat-type priority 140
rule-variable bearer 3gpp imei priority 141
rule-variable bearer tcp os-signature 144
end
```
Updating IBs for Tethering Functionality

When there are updates to the IBs for TACs, UAs, and OSs, new information bases (IBs, or databases) need to be pushed to the ASR servers.

Databases are built at scheduled date and times and are pushed automatically. You can also push the databases to the ASR servers manually, as described below. Prior to sending the data, you can view and modify it.

To push IBs to multiple ASR servers:

```
push report report-name dataset dataset-name time merge-time
serverfile file-with-remote-user-information protocol SFTP
```

Where:

- `report-name` is the name of an existing report to be pulled and run for this instance.
- `dataset-name` is the name of the dataset on which the report should be run.
- `merge-time` is the time that the data is being pushed, expressed in format `YYYY-MM-DDTHH:MMZ`.

For example:

```
push report dataset TetheringMergeIB time 2012-02-02T00:00Z
serverfile /data/server protocol SFTP
```

The following sections are examples of the databases that need to be sent periodically to the ASR servers:

**UA Database**

<table>
<thead>
<tr>
<th>Version 1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN UA-DB</td>
</tr>
<tr>
<td>GoogleAnalytics/1.1 (Linux; U; Android 2.2.1; en-us; SCH-I400; Build/FROYO)</td>
</tr>
<tr>
<td>MOT-ZN4/00.72 UP.Browser/7.2.7.5.253 (GUI) MMP/2.0</td>
</tr>
<tr>
<td>NOKIA-7205 UP.Browser/6.2.3.9.j.1.107 (GUI) MMP/2.0</td>
</tr>
<tr>
<td>BlackBerry9650/5.0.0.345 Profile/MIDP-2.1 Configuration/CLDC-1.1</td>
</tr>
</tbody>
</table>

Copyright © 2014, Cisco Systems, Inc.
OS Database

Version 1.0
BEGIN OS-DB
48|128|1|65535|1260|0|1016
48|128|1|65535|1260|0|1008
48|128|1|65535|1260|0|1009
48|128|1|65535|1260|0|1004
48|128|1|65535|1260|0|1005
48|128|1|65535|1260|0|1013
48|128|1|65535|1260|0|1012
48|128|1|65535|1260|0|1017
END OS-DB

TAC Database

Version 1.0
BEGIN TAC-DB
01193200
35598803
35758404
35341704
01052200
END TAC-DB
Job Timeout Configurations and Auto Bin Sliding in Collector

This section describes the procedure to configure the **TimeOut** parameters for the various jobs in order to avoid the jobs from getting stuck in case data stops coming in. It also enables auto bin sliding in the collector node so that the collector moves ahead even if no data is coming in. Hence, the collector is always at current time.

If any of these steps fail, do not proceed. Instead, contact Cisco Technical Support to resolve the issue.

1. Download the configuration file:
   
   SCP the `timeOut.txt` file to the master and standby collector nodes under the `/data` directory.

2. Log into the master Collector node and go to the oozie subshell:

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   host [cluster : master|standby](config)# pnx subshell
   subshell-name
   ```

3. Stop all jobs:

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

   **Note:** Wait until all jobs have been stopped and the prompt returns.

Applying Configurations

By default, all the jobs wait infinitely until the data is received for that particular instance. Instead, we will set it to a finite time so that once the timeout has been reached, the current instance is executed with the available data and the job proceeds to the next instance.

To set the timeout values:
1. Log into the master Collector node and go to _shell:

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

2. Run as admin:

   admin@host# cli /data/timeOut.txt

3. Repeat steps 1 and 2 on the standby Collector node.

### Restarting Required Processes

1. Log into the master Collector node and restart the Collector:

   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   host [cluster : master|standby](config)# pm process collector restart

2. Repeat step 1 on the standby Collector node to restart the Collector.

3. Log into the master Collector node and go to the pmx oozie subshell:

   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   host [cluster : master|standby](config)# pmx subshell subshell-name

4. Start all jobs:

   pm extension (oozie)> run job all
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, and log in to the standby Collector node to synch the IBs from the master Collector.

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 15.

   **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:

```bash
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
  type gateway-type
  edr-file-path gateway-edr-file-path
  bulkstat-file-path incoming-BS-files-path-on-collector
```

**Note:** See "Modifying Gateway Attributes" on page 113 for more information on gateway attributes and a sample of the output generated by this command.

For example:

```bash
pm extension (aggregation_center) > add gateway name GMPLAB1
  region EAST
  location USA
  schema_version 15
  ip 10.10.10.255
  timezone America/Recife
```
MURAL Operations and Troubleshooting Guide

```
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
bulkstat-file-path /California/bs111
```

4. To view new schema version that has been upgraded:

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

5. Check whether all Collector IPs are added for pushing this information:

```
pm extension (aggregation_center)> show collector IPs
```

6. 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
```

7. Push gateway configuration to all collectors:

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

8. Log into the standby Collector node and go to the pmx aggregation_center subshell:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# pmx subshell subshell-name
```
9. Fetch IBs from the inbox:

```
pm extension (subshell-name)> fetch all ibs from inbox
```

10. Repeat the previous two steps on the pmx bulkstats subshell.
Modifying the Database

Switching from Replay to Live Mode to Receive Live Data

To bring a freshly installed system from testing to production environment, you must change the system from replay mode to live mode to enable reception and processing of current data.

In live mode, incoming data goes into the current 5-minute bin. Once data beyond the time period for that bin starts coming in, the system adds it into the next bin and any data with earlier time stamps that comes to in to the Collector is discarded.

Before You Begin

The procedures described in this section assume that you have setup and tested your system using replay mode, as described in "Adding Postdated Data to the Collector Using Replay Mode" on page 99

Other prerequisites include the following:

- When you switch from replay mode to live mode, we recommend that the data already in the system not be older than 3 or 4 days old. Older data might cause a delay while the system catches up processing to the current time.

- Any data that was fed to the system while in replay mode should be from the same ASR 5000 platforms that will send data to this system in live mode.

- IB entries must be current and accurate. If they are not, update the IBs before switching the system to live mode.

- Before switching from replay mode to live mode, disable any existing ASR 5000 feeds and turn it back on to start sending live feeds for both EDR and bulkstats to the Collectors.
Changing the System from Replay to Live Mode

1. Log into the master Collector node and go to _shell:

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

2. Check the current time on the system:

   ```
   host [cluster : master] # date
   ```

   The resulting output may resemble:

   ```
   Wed Jun 5 07:07:29 UTC 2013
   ```

3. Verify that _DONE files have been created for bins older than the two current 5-minute bins for all three types of data flows—edrhttp, edrflow, and bulkstats. If these files have been created, then data processing on the system has caught up with the current time.

   For example, if current time is **Jun 5 07:07:29 2013**, then the current open bins would be `2013/06/05/07/00` and `2013/06/05/07/05` (timestamps are in the format yyyy/mm/dd/hh/mm) and you would want to find _DONE files for 5-minute bins older than these two bins, such as for `2013/06/05/06/55` or `2013/06/05/06/50` or `2013/06/05/06/45`.

   To verify the presence of the correct _DONE files for edrhttp, edrflow, and bulkstats, run:

   ```
   host [cluster : master] # hadoop dfs -ls
   /data/collector/1/output/*/2013/06/05/06/55
   ```

   The resulting output may resemble:

   ```
   -rw-r--r-- 2 admin supergroup 95 2013-06-05 07:05
   /data/collector/1/output/bulkStats/2013/06/05/06/55/1370415300._DONE
   -rw-r--r-- 2 admin supergroup 95 2013-06-05 06:55
   /data/collector/1/output/edrflow/2013/06/05/06/55/gurgaon.EDRFLOW.1370415300.0
   ```
Notice the strings which end with \_DONE in the above output. If these files are not present, then check again later until they are displayed before proceeding with the next step.

4. Log into the master Collector node and change replay mode to live mode:

```
host [cluster : master]\> en
host [cluster : master]\# conf t
host [cluster : master]\(config)\# pm process collector launch
params 1 \"
host [cluster : master]\(config)\# no pm process collector
launch params 1
host [cluster : master]\(config)\# write memory
```

5. Log into the standby Collector node and repeat the previous step.

6. Log into the master Collector node and restart the Collector process:

```
host [cluster : master]\(config)\# pm process collector restart
```

7. Verify that Collector is continuing to pick up data—ensure that live feeds are coming to the Collector system for both EDR and bulkstats. The Collector should be writing data to the HDFS every 5 minutes.

To view the Collector output, log into the master Collector node and enter:

```
host [cluster : master]\# hadoop dfs -ls
/data/collector/1/output/*/YYYY/MM/DD/hh/*
```
Where you replace the year, month, date, and hour in the above command for the current system time. This command lists the data files being created as per current time for all three data flows—bulkstats, edrhttp & edrflow. (Files for new times are created as the live feeds comes in and system time proceeds.)

For example:

```
host [cluster : master]# date
Wed Jun 5 08:34:12 UTC 2013
host [cluster : master]# hadoop dfs -ls
/data/collector/1/output/*/2013/06/05/08/*
```

The resulting output may resemble:

```
-rw-r--r-- 2 admin supergroup 95 2013-06-05 08:10
/data/collector/1/output/bulkStats/2013/06/05/08/00/1370419200._DONE
-rw-r--r-- 2 admin supergroup 11195 2013-06-05 08:00
/data/collector/1/output/edrflow/2013/06/05/08/00/gurgaon.EDRFLOW.1370419200.0
-rw-r--r-- 2 admin supergroup 21195 2013-06-05 08:00
/data/collector/1/output/edrhttp/2013/06/05/08/00/gurgaon.EDRHTTP.1370419200.0
...
```

8. Enable the job timeout settings in order to ensure that jobs do not hang if input feeds stop coming to collector for a period of time. For information on enabling job timeout settings, see "Job Timeout Configurations and Auto Bin Sliding in Collector" on page 133.

9. Restart the jobs (if you have not already done so as part of enabling job timeout settings).

10. Verify that new data is available in the UI. In a browser, access the UI and:
a. Select **Today**. You should see that data is updated every hour.

b. Go to the bulkstats and KPI pages and verify that recent data is displayed and continues to be updated every hour.
Cleaning and Replaying the Insta Database

Clearing the tables in the Insta database allow space in the database for new data or replaying old data.

If any of these steps fail, do not proceed. Instead, contact Cisco Technical Support to resolve the issue.

Cleaning the Insta Database

1. Before cleaning the database, stop all oozie jobs on the master Collector node to avoid creating an inconsistent state, and ensure the ASR is no longer sending any data:

   a. Log into the master Collector node and go to the oozie subshell:

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   host [cluster : master|standby](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. Go to the master and standby UI nodes and stop running tomcats to avoid interaction with Insta machines:

   ```
   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
- bulkstats
- reportAtlas
- rge
- httperror
- launcher

3. SSH to the GMS using the management IP address and start the installation on Insta nodes with force format:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# install appliance
cluster cluster-name INSTA-CLUSTER-NAME force-format
```

**Note:** Ensure that same version of the mural.xml used during installation is still activated. You can upload the mural.xml as explained in the MURAL Installation Guide.

4. Monitor the installation status on Insta UCS blades:

```
host [cluster : master|standby](config)#
install appliance show installation-status cluster INSTA-CLUSTER-NAME
```

**Note:** The installation process takes approximately one hour to complete, although it may vary by installation site.

Once both nodes are successfully installed below message is shown:

```
Node successfully installed
```

5. Log into the master Insta node to check if Insta is correctly up and running:
MURAL Operations and Troubleshooting Guide

```
host [cluster : master]> en
host [cluster : master]# conf t
(config)# insta infinidb get-status-info
```

The resulting output may resemble:

```
Infinidb 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:** This must show the adaptor and both the instances as **RUNNING** (it may take some time).

**Replaying Data in the Insta Database**

Replay the data in the MURAL system after cleaning the Insta database.

1. Stop all input feeds to the system until after this activity is complete.

2. Stop all UI processes.

   Log into the master and standby UI nodes using shell and run the following to stop all tomcat 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
   - bulkstats
   - reportAtlas
   - rge
   - httperror
   - launcher

3. Stop oozie jobs:
a. Log into the master Collector node and go to the oozie subshell:

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

4. From the CLI, log into the master Collector node and stop the processes to prevent receipt of data:

```
host [cluster : master]> en
host [cluster : master]# conf t
host [cluster : master](config)# pm process collector terminate
```

5. Repeat the above step on the standby Collector node.

6. Reset the hadoop file system (HDFS) and backup HDFS to cleanup all the data present on the HDFS.

   a. Log into the master Collector node and set hadoop and backup HDFS to uninit state:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# pmx
```
b. Repeat the previous step on the standby Collector node.

c. From CLI, restart HDFS on the master Collector node:

```
host [cluster : master|standby](config)# write memory
```

```shell
pm extension> set hadoop namenode UNINIT
pm extension> set backup_hdfs namenode UNINIT
pm extension> quit
```

```shell
host [cluster : master|standby](config)# pm process tps restart
```

d. Repeat the previous step on the standby Collector node.

e. Wait 10-15 minutes for the system to restart HDFS processes. Then log into the master Collector node and verify that all processes are up:

- List NameNode, SecondaryNameNode, and JobTracker processes for primary HDFS:

```
> en
# _shell
# ps -ef | grep java | egrep -v "backup|grep" | awk '{print $NF}'
org.apache.hadoop.hdfs.server.namenode.NameNode
grep Node
org.apache.hadoop.hdfs.server.namenode.SecondaryName
Node
grep.mapred.JobTracker
start
```

- Run the previous command to list the DataNode and NameNode processes on the backup HDFS:

```
# ps -ef | grep -v "grep" | grep -i backup | awk '{print $NF}''
```

The resulting output may resemble:
MURAL Operations and Troubleshooting Guide

- Run the report command to list all Compute nodes which are in normal state, and dead nodes with almost 90% disk availability on all Data nodes:

```
> en
# _shell
# hadoop dfsadmin -report
```

The resulting output may resemble:

```
DFS Used%: 2.41%
...
Datanodes available: 1 (1 total, 0 dead)
... Name: 10.10.2.13:50010
Decommission Status: Normal
...
DFS Used%: 2.12%
DFS Remaining%: 97.53%
```

- Log into the standby Collector node and run the command for the DataNode and NameNode processes on the backup HDFS:

```
> en
# _shell
# ps -ef | grep -v "grep" | grep -i backup | awk '{print $NF}'
```

The resulting output may resemble:

```
org.apache.hadoop.hdfs.server.datanode.DataNode
org.apache.hadoop.hdfs.server.namenode.NameNode
```

7. Disable job timeout and collector bin sliding.
8. Download the configuration file.

   SCP the revert_timeOut_binSliding.txt file to the master and standby Collector nodes under /data directory.

   **Note:** Obtain revert_timeOut_binSliding.txt from Cisco Technical Support.

9. Configure the timeout for all the jobs on both the master and standby Collector nodes:

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# _shell
   admin@host# cli /data/revert_timeOut_binSliding.txt
   ```

10. Start the Collector process on the master, then the standby respectively, by running the following commands:

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   host [cluster : master|standby](config)# pm process collector launch params 1 -r
   host [cluster : master|standby](config)# write memory
   host [cluster : master|standby](config)# pm process collector restart
   ```

11. Check the Insta status on both master and standby Insta nodes:

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   host [cluster : master|standby](config)# insta infinidb get-status-info
   ```

   The resulting output may resemble:

   ```
   Infinidb 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:** Only proceed to next step when the output starts showing the status as **RUNNING** for instance 0, instance 1 and Adaptor.

If the data which you are going to feed to the system in replay mode is for a different or new ASR (one which was not added to the IBs as part of the initial configurations), then go to the section of this guide on "Adding a New ASR for EDR and Bulkstats" on page 81

Otherwise, proceed with the next section to start data and processes.

### Maintaining Information Bases—For Bulkstats

Insta node processes must be running to execute commands in this section.

1. 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.

   For example: `host [cluster: master]`

2. Change the port to 22222.

   **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
   ```

3. Re-generate the 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.

4. Individually add the IP addresses (physical IP address) of each node which needs to receive the IBs. For instance, add the master and standby Collector nodes as well as the master and standby UI node. If they are configured, use control VLAN IP addresses. Otherwise, use the management IP addresses.

Run the following command for each destination:

```
pm extension (bulkstats)> add ib_destination IP-address
```

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 hours to complete.

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-name to IB-destination
```

Note: If the push is interrupted before it completes, you need to restart it.

6. Fetching all IBs received by the inbox in the previous step will sync the standby Collector node with the IBs.

   a. Log into the standby Collector node and go to the bulkstats subshell:

```
host [cluster : master|standby] > en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# pmx subshell subshell-name
```

   b. Fetch IBs from the inbox:

```
pm extension (subshell-name) > fetch all ibs from inbox
```

7. Repeat the previous step on any distro NameNodes.

Maintaining Information Bases—For EDR

Insta node processes must be running to execute commands in this section.

1. Log into the master Collector node 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-
```
2. Once changes are completed, generate and push the IBs:

```
port-number
[admin@host]# write mem
```

```
pm extension (subshell-name) > generate all ibs
pm extension (subshell-name) > push all ibs
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 hours to complete.

3. Fetching all IBs received by the inbox in the previous step will sync the standby Collector node with the IBs.

   a. Log into the standby Collector node and go to the aggregation_center subshell:

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   host [cluster : master|standby](config)# pmx subshell
   subshell-name
   ```

   b. Fetch all IBs from inbox:

   ```
   pm extension (subshell-name) > fetch all ibs from inbox
   ```

**Sending Fresh Data to the System**

1. Start sending EDR records and bulkstats records files to the system and note the time.

2. Log into the GMS machine from which installation scripts were executed:

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# _shell
   ```
3. Set the data start times on both master and standby Collector nodes to the
time from which EDR and bulkstats data starts coming into the system. Run
the set job time script:

```shell
admin@host# ./setOozieTime --dataStartTime data-start-time --node
collector-mgmt-IP-master|standby-collector-IP --password
admin-user-pwd --verbose
```

For example, if EDR and bulkstats data starts coming into the system from
1st April, 2013, 06:00 onwards, run scripts with data-start-time value as
"2013-04-01T06:00Z"

```shell
admin@host# ./setOozieTime --dataStartTime 2013-04-01T06:00Z
--node 192.168.147.11 --password admin@123 --verbose
```

### Restarting Processes

1. From oozie subshell, run data processing commands on the master Col-
lector (Name) node:
   a. Log into the master Collector node and go to the pmx oozie subshell:

   ```shell
   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   host [cluster : master|standby](config)# pmx subshell
   subshell-name
   ```
   b. Start all jobs:

   ```shell
   pm extension (oozie)> run job all
   ```

   Cleanup jobs should not be run as these steps are processing older data and
cleanup jobs may move ahead and delete the data before it gets processed.
They will start when we run the code run job all, so we need to stop them
using the stop jobname job-name command from the pmx oozie subshell.

2. Log into the master Collector and go to _shell:
3. Compile a list of cleanup jobs by running the following command:

```
admin@host# cli -t "en" "show run full" | grep "Cleanup" | awk \\-F'/\' '{print $6}' | sort -u | grep -v "value"
```

4. Go to the pmx oozie subshell:

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

```
subshell-name
```

5. Individually stop each of the cleanup jobs listed in the output of step 3:

```
# stop jobname job-name
```

For example, replacing job-name with:

- Cleanup15minOutput
- Cleanup1DayOutput
- Cleanup1WeekOutput
- Cleanup1hrOutput
- Cleanup4WeekOutput
- Cleanup5minOutput
- CleanupAtlas
- CleanupCollector
- CleanupLogs
- CleanupRawRecords
- CleanupSegment
- CleanupSubcrBytes
- CleanupTBADaily
- CleanupTBAWeekly
- CleanupTethering
- CleanupTopSubcr
We cannot continuously feed data to the system because we cannot run cleanup jobs in replay mode. As a result, the hdfs disk is likely to get full if we feed unlimited data to the system.

a. Run `hadoop dfsadmin -report` on the master Collector node.

b. Ensure that the value for DFS Used% is less than 70%.

c. If DFS Used% is higher than 70%, some data cleanup is required and you need to contact Technical Support.

6. Start UI tomcat instances only after at least 2 hours of data have been pushed into the Insta node.

a. Check the data in Insta:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# _shell
admin@host# cli -t "en" "show run full" | grep "cubes-database" | awk '{print $NF}' DATABASE_MURAL
```

b. Check the data in bulkstats:

```
admin@host# idbmysql edr-db-name -e "select * from bin_metatable"
admin@host# idbmysql bs-db-name -e "select * from bin_metatable"
```

Where:

- The `mints` field value of first row in each output signifies the epoch time data processing started.
- The `maxts` field value of first row in each output signifies the epoch time till which data has been processed.
- `edr-db-name` is the name of the EDR database.
- `bs-db-name` is the name of the bulkstats database

7. Log into master UI node and start the processes:
Where `process-name` is replaced with:

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

**Note:** Wait for two minutes between starting each process.

8. Repeat the previous step on the standby UI node.

9. Access the UIs by going to the URL `https://<domainName>:8443/` through your browser.

The domain name to be used is the one which was provided at the time of initial configuration via GMS for the UI nodes configuration details.
Update Databases for URL Categories and IMEI Numbers

Databases for URL categories and IMEI numbers are updated either by means of a new software release or a patch release.

If you receive updates to URL categories and IMEI numbers as a patch, you apply the patch, apply the updated IBs, and fetch the IBs.

1. If the database updates are received in a patch release, install the patch according to the instructions in the Release notes.

If the database updates are received in a new software release, install the new software version on the servers. See the MURAL Installation Guide.

2. Apply the updated IBs to the system.

   Note: If updates are received in a new software release, complete this step after the upgrades are completed on the Insta nodes but before starting data processing (restarting the oozie jobs).

If the data processing jobs were stopped during step 1, complete the following steps before starting data processing (restarting the oozie jobs).

   a. On the master Collector node, 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
```

   b. Go to the pmx aggregation_center subshell:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
```
MURAL Operations and Troubleshooting Guide

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

c. Fetch IBs from the image:

```
pm extension (aggregation_center)> update ib ib-name from image
```

Where `ib-name` is the name of the Information Base to be updated, such as:

- IBStore.tab
- ivFile.txt
- symmetricKey.txt
- privateKey.txt
- manufacturer.list
- category.list
- url.list
- sp.list
- model.list
- mobileappname.list
- mobileappcategory.list
- mobileapptype.list
- mime.list
- device_man_dg_os.map
- os.list
- contentType.map

d. Generate and push all IBs:

```
pm extension (aggregation_center)> generate
pm extension (aggregation_center)> push all ibs
```

e. Log into the standby Collector node and go to the pmx aggregation_center subshell:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
```
f. Fetch the updated IBs:

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

pm extension (aggregation_center)> fetch all ibs from inbox
```
Backup Strategy

If any of these steps fail, do not proceed. Instead, contact Cisco 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 189 of this guide.

Procedures for how to perform backups of the Insta Database can be found in "Backing Up and Restoring Computed Data" on page 164 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 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/backup-file-name
admin@host# cp /config/gms/* /data/GMS_backup/backup-file-name
```

Where `backup-file-name` is the name of the configuration backup file.

2. Repeat the previous step on the Compute node.

**Note:** The above folder `/data/GMS_backup` can be backed up on external storage, such as a tape drive.
Back up and Restoring Computed Data

This topic describes how to take backup of a database on the Insta node running in a high-availability environment, and how to subsequently restore it. If the database is very large (a TB or more), see "Back up and Restoring a Large (TB) Insta Database" on page 170.

If any of these steps fail, do not proceed. Instead, contact Cisco Technical Support to resolve the issue.

You can restore the backup on the same node or on another newly configured node. In this section, run all commands on pm1, the master Insta database, unless otherwise specified. Dropping the old database is not necessary and is not covered in this document.

**Note:** A backup or restore may not be successful if a switchover occurs during this period.

**Before You Begin**

Verify there is enough disk space to store the backup, using one of these two methods: **Detailed View** or **Summary View**.

The Detailed View method may take 1-2 hours to complete, depending upon the size of the databases. It is recommended you use the Summary View if individual database sizes are not required.

**Detailed View**

1. This command gives you the size for all databases on the system as well as the total for all databases. Note the size of the database you want to take a backup of.

For example:

```
admin@host# /usr/local/Calpont/bin/databaseSizeReport.sh
```

The resulting output may resemble:
2. The total size of the databases was approximately 9.5GB. To verify that the amount of space needed for the backup is available in the location where you want to store the database backup, run:

   admin@host# df -kh /data

The resulting output may resemble:

<table>
<thead>
<tr>
<th>Filesystem</th>
<th>Size</th>
<th>Used</th>
<th>Avail</th>
<th>Use%</th>
<th>Mounted on</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/sda11</td>
<td>497G</td>
<td>341M</td>
<td>472G</td>
<td>1%</td>
<td>/data</td>
</tr>
</tbody>
</table>

This step shows there are 472GB available which is more than enough memory to store the 9.5GB databases.

**Summary View**

If you do not need to know the size of each individual database:

1. Log into the master Insta node and run:

   host [cluster : master]> en
   host [cluster : master]# _shell
   admin@host# df -h

The resulting output may resemble:
The sum of the values shown under the 'Used' Column is the total size of Insta Db in compressed format and so 3 times of this space should be available on the disk whose path we are giving in the backup command.

For the above example, the total would be 65G+57G = 122G.

Therefore, the total space required will be approximately 3*122G = 366G.

2. If we want to store it on the /data folder, there needs to be at least 1185GB available. Use the same verification as done in the Detailed View method:

```bash
admin@host# df -kh /data
Filesystem  Size  Used  Avail  Use% Mounted on
/dev/sda11  497G  341M   472G   1%      /data
```

3. In this example, the above step shows there are 472GB available which is not enough memory for the database we checked in step 1.

**Verifying Presence of Data in Database**

These steps will take a close look at the values for one or more minimum timestamp (mints) and maximum timestamp (maxts). Later, when verifying restore operations for each database, you will need to reference the values of the mints and maxts generated by these commands.

1. Verify the database contains some data, and note the values mints and maxts to use later in verifying the restore operation for each database.
   a. Find the names of the databases configured in the Insta configuration—there are two database names against each instance number, one for EDR and one for bulkstats:

```bash
admin@host> en
admin@host# _shell
```
admin@host# cli -t "en" "conf t" "show run full" | grep -I "cubes-database"

The resulting output may resemble:

| insta instance 0 cubes-database mural |
| insta instance 1 cubes-database bulkstats |

**Note:** It is recommend that you back up both instances.

b. Go to _shell:

```bash
admin@host> en
admin@host# _shell
```

c. Run the following command for each database and verify that there are values for one or more minimum timestamp (mints) and maximum timestamp (maxts):

```bash
admin@host# idbmysql database mural -e "select * from bin_metatable"
admin@host# idbmysql bulkstats -e "select * from bin_metatable"
```

The resulting output may resemble:

```
admin@host# idbmysql database mural -e "select * from bin_metatable"
+-------------------+-------------------+-------------------+-------------------+-------------------+
| binclass | aggregationinterval | mints | maxts | bintype |
+-------------------+-------------------+-------------------+-------------------+-------------------+
| 60min   | -1 | 1358326800 | 1358359200 | NULL |
| 60min   | | 400 | 0 | 0 | NULL |
| 60min   | | 604800 | 0 | 0 | NULL |
| 60min   | | 2419200 | 0 | 0 | NULL |
+-------------------+-------------------+-------------------+-------------------+-------------------+
admin@host# idbmysql bulkstats -e "select * from bin_metatable"
```
2. Verify that the Insta process is running:

```
admin@host> en
admin@host# conf t
host [cluster : master|standby](config)# _shell
host [cluster : master|standby](config)# cli -t "en" "config t" "show pm process insta" | grep "Current status"
```

The resulting output may resemble:

```
| Current status: running |
```

3. Stop all oozie jobs on the Collector node in order to avoid creating an inconsistent state (caused by pushing data to the database while a backup is being taken):

```
admin@instanode# cli -m config
admin@instanode(config)# pmx
pm extension> subshell oozie
pm extension (oozie)> stop jobname all
```

4. Verify that all oozie jobs are stopped:

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

**Note:** Make a note of these `mints` and `maxts` values.
Verify that the message *No Jobs match your criteria!* is displayed. If not and there are still jobs running, wait a short time and try again before proceeding to the next step.
Back 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 scenarios:

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:

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   host [cluster : master|standby](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:
3. Stop the Insta process from the master Insta node.

```plaintext
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# pm process rubix terminate
```

4. On the master Insta node, verify that the node is designated as 1:

```plaintext
[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 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:

```plaintext
> en
# _shell
[admin@host ~]# cd /data/Calpont
```
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.

### 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:

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

```bash
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:

```bash
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:

```bash
> 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:

   ```bash
   > 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:
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:

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

**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:

```bash
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
```
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
host [cluster : master](config)# insta infinidb disk-mgmt
    clean schema
host [cluster : master](config)# write mem
```

3. Mount `dbroot` folder for backup on new active:

```
host [cluster : master]# _shell
[admin@host ~]# mount /dev/mapper/dbroot1 /data/Calpont/data1/
[admin@host ~]# mount /dev/mapper/dbroot2 /data/Calpont/data2/
```

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/
```

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
```

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 the 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.

```shell
host [cluster : standby]> en
host [cluster : standby]# _shell
[admin@host ~]$ mount -o remount,rw /
```

7. On the new master Insta node, install `infinidb`:

```shell
host [cluster : master](config)# insta infinidb install
host [cluster : master](config)# write mem
```

8. Verify the status of the installation:

```shell
host [cluster : master](config)# insta infinidb get-status-info
```

The resulting output may resemble:

```
Infinidb Install status : INSTALLED
and Infinidb Adaptor status : Adaptor Running
```

9. Restart the insta process:

```shell
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:
10. Restart the jobs:
   
   a. Go to the pmx oozie subshell:

   ```
   host [cluster : master](config)# write mem
   host [cluster : master](config)# insta infinidb get-status-info
   ```
   
   b. Run:

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

11. Restart Rubix on all UI nodes. Log into each UI node one by one and repeat this step:

   ```
   host [cluster : master](config)# write mem
   host [cluster : master](config)# insta infinidb get-status-info
   ```
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 IB

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 Cisco 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_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:

<table>
<thead>
<tr>
<th>total 8</th>
<th>drwxr-xr-x 2 admin root 4096 Jan 31 13:56 work</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>drwxr-xr-x 2 admin root 4096 Jan 31 13:56 inbox</td>
</tr>
</tbody>
</table>

4. Copy the /data/ib_backup directory 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 folder, 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:

<table>
<thead>
<tr>
<th>total 8</th>
<th>drwxr-xr-x 2 admin root 4096 Jan 31 13:56 work</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>drwxr-xr-x 2 admin root 4096 Jan 31 13:56 inbox</td>
</tr>
</tbody>
</table>

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/
   ```

   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. Log into the master Collector node and change the port to 22222.

   **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. Go to the pmx bulkstats subshell.

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   host [cluster : master|standby](config)# pmx subshell subshell-name
   ```

3. Provide the control network IP addresses of the master Collector node, standby Collector node, and all UI nodes.

   ```
   pm extension (bulkstats)> add ib_
   destination IP-Address
   ```

   Where `IP-Address` is the network IP address of nodes you are adding to the IB database.

4. Then, 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]:
   ```
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 hours to complete.

   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.

6. Log into the standby Collector node and fetch IBs from the pmx subshell inbox:
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)> 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 hours to complete.

3. Once the above push command succeeds, log into the standby Collector node and execute the following commands. They will fetch all IBs received by the inbox from the push conducted in the previous step and will sync the standby Collector node with the IBs:

   ```
   pm extension (subshell-name)> fetch all ibs from inbox
   ```

4. 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.
Backs 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 a period of seven days.

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"

Backs 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):

<table>
<thead>
<tr>
<th>Directory Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>/data/collector/bulkstats_files_backup/GMPLAB1/</td>
</tr>
<tr>
<td>/data/collector/bulkstats_files_backup/GMPLAB2/</td>
</tr>
<tr>
<td>/data/collector/edrflow_backup</td>
</tr>
<tr>
<td>/data/collector/edrhttp_backup</td>
</tr>
</tbody>
</table>

2. Back up the raw data for any adapter to a new folder locally or onto a new SAN drive.

For example, to back up http files:

```bash
[admin@host ~] # mkdir -p /data/http_rawdata
[admin@host ~] # cp -R /data/collector/edrhttp_backup /data/http_rawdata
```

3. 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 Postgres (PGSQL) database data on a master Insta node.

First, verify that the PGSQL database is running. To do this, log into the UI and create a random filter on any of the tabs on the UI.

If any of these steps fail, do not proceed. Instead, contact Cisco Technical Support to resolve the issue.

Backing-Up the PGSQL Database

To back-up an PGSQL database:

1. Log into the master UI node and go to the CLI mode.

2. Stop all java processes running on the system:

   ```
   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
   - bulkstats
   - reportAtlas
   - rge
   - httperror
   - launcher

3. On the standby UI node, repeat the previous step.

4. Log into the master Insta node and backup the database:

   ```
   pg_dump -U postgres -c -o <DB name> -f <file_path>
   ```

   For example:
5. Save the backup as a tar file:

```
cd /data
tar -cvzf pgsqlDB.tgz pgsqlDB
```

**Restart Processes on the UI Nodes**

1. Log into the master UI node and go to the CLI mode.

2. Restart all 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
- launcher

**Note:** Wait for two minutes between starting each process.

3. Log into the standby UI node, repeat the previous step.

**Restoring a PGSQL Database**

To restore a PGSQL database:

1. Log into the master UI node and go to the CLI mode.

2. Stop all java processes on the system:

```
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
- bulkstats
- reportAtlas
- rge
- httperror
- launcher

3. Log into the standby UI node, repeat the previous step.

4. Check that PGSQL process is running on the master Insta Node:

   host [cluster : master](config)# show tps pgsql status

   The resulting output may resemble:

   postmaster (pid 28344) is running...

5. Restore the backed up PGSQL DB on the master Insta node:

   tar -xvzf /data/pgsqlDB.tgz
   (config)# tps pgsql file-path path-to-backup-file-to-restore
dbname database-name restore

   For example:

   (config)# tps pgsql file-path /data/pgsqlDB dbname rubixdb
   restore

6. Log into the master UI node and restart 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
- launcher

**Note:** Wait for two minutes between starting each process.

7. Log into the standby UI node and repeat the previous step.

8. 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
   ```

9. 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.

10. Verify that filters are available.
Troubleshooting

Troubleshooting User Interface Access Problems

This topic describes the most common causes of trouble 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 is reachable. If it is 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:

   ```
   ps - ef | grep atlas
   ```

3. If you see errors of the type `- Error retrieving data / Module Load error`.

4. 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
   ```

5. After two minutes check that atlas app is getting listed under running processes. Go to the _shell prompt and run:

   ```
   ps - ef | grep atlas
   ```
6. 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 postmaster process is running on the master Insta node:

```plaintext
host [cluster : master] > en
host [cluster : master] # conf t
host [cluster : master] (config) # show tps psql status
```

2. If the postmaster is stopped, start the postmaster process using:

```plaintext
host [cluster : master] (config) # tps psql restart
host [cluster : master] (config) # show tps psql status
```

The resulting output may resemble:

```
postmaster (pid7320) is 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.
Troubleshooting Abrupt Failure of Jobs

This topic provides information about preliminary diagnosis and analysis if data processing has stopped on the 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 will stop moving ahead every hour.

Debugging Insta Nodes

Follow these steps for both master and standby Insta nodes to rule out the possibility of issues on those nodes.

Checking the Status

1. Log into the master Insta node and check if the Insta process is running:

   ```
   host [cluster : master]# cli -m config
   host [cluster : master](config)# _shell
   [admin@host ~]# cli -t "en" "config t" "show pm process insta"
   | grep "Current status"
   Current status: running
   ```

2. Repeat step 1 on the standby Insta node.

3. Log into the master Insta node and check the status of the Insta database. It should be in a **RUNNING** state. It should show all Insta node IPs in Active Insta nodes.

   ```
   host [cluster : master]# cli -m config
   host [cluster : master](config)# insta infinidb get-status-info
   ```

   The resulting output may resemble:
MURAL Operations and Troubleshooting Guide

getsysteminfo  Thu Jun 13 13:57:19 2013
System CLUSTER-inst
System and Module statuses
Component Status Last Status Change
------------- --------------- ---------------------
System ACTIVE Thu Jun 13 11:15:47 2013
Module pm1 ACTIVE Tue Jun 11 10:20:07 2013
Module pm2 ACTIVE Tue Jun 11 10:19:31 2013
Active Parent OAM Performance Module is 'pm1'

Calpont Process statuses
Process Module Status Last Status Change
------------- ------ ------- ---------------------
ProcessMonitor pml ACTIVE Tue Jun 11 10:18:16 2013
  27913
ProcessManager pml ACTIVE Tue Jun 11 10:18:26 2013
  28498
DBRMControllerNode pml ACTIVE Tue Jun 11 10:19:13 2013
  28932
ServerMonitor pml ACTIVE Tue Jun 11 10:19:15 2013
  29004
DBRMWorkerNode pml ACTIVE Tue Jun 11 10:19:17 2013
  29038
...
Active Alarm Counts: Critical = 0, Major = 0, Minor = 0,
Warning = 0, Info = 0 Infinidb Install status : INSTALLED
Total instances configured: 2
Insta instance 0 service status : RUNNING
Insta instance 1 service status : RUNNING
Infinidb Adaptor status : Adaptor Running
Active insta nodes : 192.168.147.15, 192.168.147.16,

4. Repeat step 3 on the standby Insta node. It should show Active Insta
nodes as 0.0.0.0.

5. If processes are already shown in a **RUNNING** status, then continue to the next step.

**Checking Port Connections**

Run the following commands to verify that the 11111|22222 ports, of the master and standby Insta nodes, are in **LISTEN** mode on their respective Insta servers.

```bash
host [cluster : master|standby]> en
host [cluster : master|standby]# _shell
[admin@host dir]# netstat -nap | grep "11111|22222" | egrep "LISTEN"
```

The resulting output may resemble:

```
tcp 0 0 0.0.0.0:11111 0.0.0.0:* LISTEN 19243/insta
tcp 0 0 0.0.0.0:22222 0.0.0.0:* LISTEN 19243/insta
```

If both ports are already in **LISTEN** mode, then continue to the next step.

**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:

```bash
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:
2. Run the following commands and make sure the `mints` and `maxts` are not all zero for each DB:

```bash
host [cluster : master] > en
host [cluster : master] # _shell
[admin@host dir]# idbmysql <edr_database_name> -e "select * from bin_metatable";
```

For example,

```bash
[admin@host dir]# idbmysql database mural -e "select * from bin_metatable"
```

<table>
<thead>
<tr>
<th>binclass</th>
<th>aggregationinterval</th>
<th>mints</th>
<th>maxts</th>
<th>bintype</th>
</tr>
</thead>
<tbody>
<tr>
<td>60min</td>
<td>-1</td>
<td>1358326800</td>
<td>1358359200</td>
<td>NULL</td>
</tr>
<tr>
<td>60min</td>
<td>86400</td>
<td>0</td>
<td>0</td>
<td>NULL</td>
</tr>
<tr>
<td>60min</td>
<td>604800</td>
<td>0</td>
<td>0</td>
<td>NULL</td>
</tr>
<tr>
<td>60min</td>
<td>2419200</td>
<td>0</td>
<td>0</td>
<td>NULL</td>
</tr>
</tbody>
</table>

```bash
[admin@host dir]# idbmysql bulkstats -e "select * from bin_metatable";
```

<table>
<thead>
<tr>
<th>binclass</th>
<th>aggregationinterval</th>
<th>mints</th>
<th>maxts</th>
<th>bintype</th>
</tr>
</thead>
<tbody>
<tr>
<td>5min</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>NULL</td>
</tr>
<tr>
<td>5min</td>
<td>900</td>
<td>1358327700</td>
<td>1358366400</td>
<td>NULL</td>
</tr>
<tr>
<td>5min</td>
<td>3600</td>
<td>1358326800</td>
<td>1358362800</td>
<td>NULL</td>
</tr>
<tr>
<td>5min</td>
<td>86400</td>
<td>0</td>
<td>0</td>
<td>NULL</td>
</tr>
<tr>
<td>5min</td>
<td>604800</td>
<td>0</td>
<td>0</td>
<td>NULL</td>
</tr>
<tr>
<td>5min</td>
<td>2419200</td>
<td>0</td>
<td>0</td>
<td>NULL</td>
</tr>
</tbody>
</table>
```
a. In this example, output from the first table (EDR) which needs to be considered are the \texttt{mints} and \texttt{maxts} values in the row which has \texttt{aggregationinterval} = -1 and \texttt{binclass} = 60\text{min}.

For example, \texttt{mints} = 1358326800, \texttt{maxts} = 1358359200

b. Similarly, output from the second table (bulkstats) which needs to be considered are the \texttt{mints} and \texttt{maxts} values in the row which has \texttt{aggregationinterval} = 900 and \texttt{binclass} = 5\text{min} will be considered.

For example, \texttt{mints} = 1358327700, \texttt{maxts} = 1358366400

3. Check the function of the \texttt{date} command using the \texttt{mints} and \texttt{maxts} values identified in step 2a and 2b, above:

<table>
<thead>
<tr>
<th>Command</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{[admin@host dir]# date -d @1358326800}</td>
<td>Wed Jan 16 09:00:00 UTC 2013</td>
</tr>
<tr>
<td>\texttt{[admin@host dir]# date -d @1358359200}</td>
<td>Wed Jan 16 18:00:00 UTC 2013</td>
</tr>
</tbody>
</table>

\textbf{Note:} This shows we have processed data from 09:00 to 18:00 on Jan 16 in Insta DB for EDR.

<table>
<thead>
<tr>
<th>Command</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{[admin@host dir]# date -d @1358327700}</td>
<td>Wed Jan 16 09:15:00 UTC 2013</td>
</tr>
<tr>
<td>\texttt{[admin@host dir]# date -d @1358366400}</td>
<td>Wed Jan 16 20:00:00 UTC 2013</td>
</tr>
</tbody>
</table>

\textbf{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 \texttt{maxts} of both databases should show as 1 hour less than the \texttt{maxts} shown on UI.

After performing all steps in this section on "Debugging Insta Nodes" on page 195, 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 34.

**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`:

   ```shell
   host [cluster : master]# cli -m config
   host [cluster : master]
   (config)# _shell
   ```

2. Check the Collector processes by running:

   ```shell
   [admin@host dir]# cli -t "en" "config t" "show pm process tps"
   | grep "Current status"
   Current status: running
   ```

   ```shell
   [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:

   ```shell
   [admin@host ~]# ps -ef | grep -i collector_file_generator | grep -v grep
   ```

   The resulting output may resemble:

   ```plaintext
   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 34
- 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 primary hdfs processes by running:

```
[admin@host dir]# ps -ef | grep java | egrep -v "backup|grep" | awk '{print $NF}'
```

The resulting output may resemble:

```
org.apache.hadoop.hdfs.server.namenode.NameNode
org.apache.hadoop.hdfs.server.namenode.SecondaryNameNode
org.apache.hadoop.mapred.JobTracker
```

7. While in the master Collector node, run the following command to verify the backup HDFS processes:

```
[admin@host dir]# ps -ef | grep -v "grep" | grep -i backup | awk '{print $NF}'
```

The resulting output may resemble:

```
org.apache.hadoop.hdfs.server.datanode.DataNode
org.apache.hadoop.hdfs.server.namenode.NameNode
```

8. Repeat the primary and backup process verifications on the standby Collector node.

9. 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]# hadoop 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 those on Insta, 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-freeded-bin

Where process-name is:

- 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:
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 on /data and /data/collector by running:

```
[admin@host dir]# df -kh
```

The resulting output may resemble:

<table>
<thead>
<tr>
<th>Filesystem</th>
<th>Size</th>
<th>Used</th>
<th>Avail</th>
<th>Use%</th>
<th>Mounted on</th>
</tr>
</thead>
<tbody>
<tr>
<td>rootfs</td>
<td>4.0G</td>
<td>2.0G</td>
<td>1.9G</td>
<td>52%</td>
<td>/</td>
</tr>
<tr>
<td>/dev/root</td>
<td>4.0G</td>
<td>2.0G</td>
<td>1.9G</td>
<td>52%</td>
<td>/</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>/dev/sdal1</td>
<td>498G</td>
<td>84G</td>
<td>389G</td>
<td>18%</td>
<td>/data</td>
</tr>
</tbody>
</table>
3. Log into the NameNode and go to _shell:

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

4. Ensure that there is enough disk space on HDFS. Also verify that HDFS processes are up and running in the report:

```bash
[admin@host dir] # hadoop dfsadmin -report
```

The resulting output may resemble:

```
DFS Used: 2895297981108 (2.63 TB)
DFS Used%: 93.95%
...
Datanodes available: 3 (3 total, 0 dead)
Name: 10.10.2.18:50010
Decommission Status: Normal
...
DFS Used%: 92.47%
DFS Remaining%: 2.43%
...
Name: 10.10.2.14:50010
...
DFS Used%: 83.79%
DFS Remaining%: 11.14%
...
Name: 10.10.2.13:50010
...
```
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]# hadoop dfs -du /data
```

The resulting output may resemble:

<table>
<thead>
<tr>
<th>DFS Used%: 91.26%</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFS Remaining%: 3.66%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Found 39 items</th>
</tr>
</thead>
<tbody>
<tr>
<td>18 hdfs://Namenode-VIP:9000/data/AggregationJobDaily</td>
</tr>
<tr>
<td>18 hdfs://Namenode-VIP:9000/data/AggregationJobWeekly</td>
</tr>
<tr>
<td>18 hdfs://Namenode-VIP:9000/data/BSAgg15min</td>
</tr>
<tr>
<td>18 hdfs://Namenode-VIP:9000/data/BSAgg1Day</td>
</tr>
<tr>
<td>18 hdfs://Namenode-VIP:9000/data/BSAgg1hour</td>
</tr>
<tr>
<td>18 hdfs://Namenode-VIP:9000/data/BulkStat</td>
</tr>
<tr>
<td>...</td>
</tr>
</tbody>
</table>

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:

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

<table>
<thead>
<tr>
<th>FAILED</th>
<th>KILLED</th>
<th>PREP</th>
<th>RUNNING</th>
<th>SUCCEEDED</th>
<th>all</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job ID</td>
<td>App Name</td>
<td>App Path</td>
<td>Console URL</td>
<td>User</td>
<td>Group</td>
</tr>
<tr>
<td>Run</td>
<td>Created</td>
<td>Started</td>
<td>Status</td>
<td>Last Modified</td>
<td>Ended</td>
</tr>
</tbody>
</table>

3. Run this command to determine the job ID for the job you are checking on:

```bash
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.

<table>
<thead>
<tr>
<th>FAILED</th>
<th>KILLED</th>
<th>PREP</th>
<th>RUNNING</th>
<th>SUCCEEDED</th>
<th>all</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job ID</td>
<td>App Name</td>
<td>App Path</td>
<td>Console URL</td>
<td>User</td>
<td>Group</td>
</tr>
<tr>
<td>Run</td>
<td>Created</td>
<td>Started</td>
<td>Status</td>
<td>Last Modified</td>
<td>Ended</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2012-12-12 06:15</td>
<td>RUNNING 2012-12-12 06:15</td>
<td>-</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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:

```bash
[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 successfully.
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:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cd folder-name</td>
<td>Change to the folder directory</td>
</tr>
<tr>
<td>ls <em>std</em></td>
<td>List files containing stdout and stderr</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ls -l <em>/std</em></td>
<td>List all files containing stdout and stderr</td>
</tr>
</tbody>
</table>

**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 then the Collectors also 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:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>en</td>
<td>Enable shell access</td>
</tr>
<tr>
<td>conf t</td>
<td>Configure terminal access</td>
</tr>
<tr>
<td>pmx subshell subshell-name</td>
<td>Enter subshell for the selected configuration profile</td>
</tr>
</tbody>
</table>
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 205.

1. Log into the nodes whose memory is 100% full and check the processes:

```
host [cluster : master|standby]# ps -ef| grep -i "datanode|tasktracker" |grep -v grep |awk '{print $NF}'
```

The resulting output may resemble:

```
org.apache.hadoop.hdfs.server.datanode.DataNode
org.apache.hadoop.mapred.TaskTracker
```

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% /  
```
4. If any of the partitions are full, notify Cisco Technical Support and provide the log files as explained in "System Logs" on page 34.
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 from a Single Node Failure" on page 1.

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.

Follow the section "Removing the Faulty or Inaccessible Node From Cluster" on page 1 to install each new Compute node in place of the failed one.

Insta Node

Follow the steps on "Recovering an Insta Node " on page 1 to install the new Insta node in place of the failed one.

Rubix Node

Follow the steps within "Recovering a Failed UI Node" on page 1 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 data is not received on the Collector for two or three hours.

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 Cisco 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, 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 ASR and the collector blades. If one is found, contact your local IT support group to resolve the problem with the
ASR 5000 server. If one is not found, determine if there is a failure at the intermediary network devices connecting ASR 5000 and the Collector.

Once connectivity is restored, verify that Collector data has resumed on SNMP traps for each of the configured adaptors edrhttp, edrflow or bulk-stats).

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:

<table>
<thead>
<tr>
<th>Size</th>
<th>Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>28M</td>
<td>/data/08</td>
</tr>
<tr>
<td>4.0K</td>
<td>/data/Base Cubes.json.30rc1</td>
</tr>
<tr>
<td>4.0K</td>
<td>/data/ReservedNames.json</td>
</tr>
<tr>
<td>7.4M</td>
<td>/data/apache-tomcat</td>
</tr>
<tr>
<td>4.0K</td>
<td>/data/atlasData</td>
</tr>
<tr>
<td>14G</td>
<td>/data/backup-hadoop-admin</td>
</tr>
<tr>
<td>7.6G</td>
<td>/data/backup_hadoop_logs</td>
</tr>
<tr>
<td>4.0K</td>
<td>/data/changeInstaIPInExporterJobs</td>
</tr>
<tr>
<td>264K</td>
<td>/data/cli_config.txt</td>
</tr>
<tr>
<td>6.7G</td>
<td>/data/collector</td>
</tr>
</tbody>
</table>

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, 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 ASR 5000 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, 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 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",
```
d. For each DC or gateway, check the input directory configured here. If files are coming 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 16805 1673 2 Nov01 ?   02:28:27 /platform_latest/collector/bin/collector
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 man-
   ager.
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 any of these steps fail, do not proceed. Instead, contact Cisco 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 and activate the updated XML:

   ```
   > en
   # conf t
   (config)# gms config mural.xml activate
   ```

   **Note:** The mural.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:

```bash
(config)# image fetch scp://admin@192.168.147.18/data/mfgcd-guavus-x86_64-20140315-133212.iso
(config)# image mount mfgcd-guavus-x86_64-20140315-133212.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 3.4.

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:
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 server. We are assuming here that LUN isn't faulty and previous LUN has been assigned to the new node as well.

8. 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 COLL-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.

9. 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
```

10. Wait for the node successfully installed message to appear before proceeding.

11. Apply any patches to the Collector nodes that have been released for this build. See the Mural Release Notes for a list of patches and installation instructions.

**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 \texttt{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 sub-shell:

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

\textit{subshell-name}
```

2. Identify the start time for the dataset, as the system uses this in the \texttt{job-start} script.

```
pm extension (oozie)> show config dataset

atlas_edrflow_1
```

The resulting output may resemble:

```
Attributes:

-----------
doneFile : \_DONE
doneFile : \_DONE
doneFile : \_DONE
doneFile : \_DONE
doneFile : \_DONE
doneFile : \_DONE
doneFile : \_DONE
doneFile : \_DONE
doneFile : \_DONE
doneFile : \_DONE
doneFile : \_DONE
doneFile : \_DONE
doneFile : \_DONE
doneFile : \_DONE
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
```

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 \texttt{\_shell} prompt:

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

### 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:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# _shell
host [cluster : master|standby]# vi /var/home/root/.ssh/known_hosts
```

### For Bulkstats

1. Log into the master Collector node and change the port to 22222:

   **Note:** Go to `_shell` and invoke a new CLI shell before changing the port. Otherwise the port change does not take effect.
2. Go to the pmx bulkstats subshell:

```
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
```

3. Generate and push all IBs:

```
pm extension (bulkstats)> generate all ibs
pm extension (bulkstats)> push all ibs
```

4. Log into the restored Collector node and go to the pmx bulkstats subshell as in step 2.

5. Fetch IBs from Inbox:

```
pm extension (subshell-name)> fetch all ibs from inbox
```

**For EDR**

1. Log into the current master Collector node 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. Go to the pmx aggregation_center subshell.
3. Generate and push all IBs.

```
pm extension (subshell-name) > generate all ibs
pm extension (subshell-name) > push all ibs
```

4. Show Collector IPs and push gateway information.

```
pm extension (aggregation_center) > show collector IPs
pm extension (aggregation_center) > push gateway configuration
```

5. Log into the restored Collector node and go to the pmx aggregation_center subshell.

```
host [cluster : master|standby] > en
host [cluster : master|standby]# conf t
host [cluster : master|standby] (config)# pmx subshell subshell-name
```

6. Fetch IBs from the inbox.

```
pm extension (subshell-name) > fetch all ibs from inbox
```

### 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 MURAL 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 -
2. Use SCP to transfer these files from the GMS to the new Collector node:

- /data/work/serverFile_tethering
- /data/work/serverFile_uncatReports

3. Create a work directory on the new Collector node:

```bash
host [cluster : master|standby](config)# _shell
host [cluster : master|standby]# cd /data
host [cluster : master|standby]# mkdir work
host [cluster : master|standby]# cd work
```

4. Login into the master Collector node and go to _shell:

```bash
admin@host(config)# _shell
```

5. Change your location to the newly created work directory:

```bash
admin@host# cd /data/work
```

6. Copy files from master Collector to the new Collector server:

```bash
[admin@host work]# scp serverFile_tethering admin@restored-mgmt-ip:/data/work/serverFile_tethering
[admin@host work]# scp serverFile_uncatReports admin@restored-mgmt-ip:/data/work/serverFile_uncatReports
```

7. Start the Collector process on the new Collector node:

```bash
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config) # pm process tps restart
```

**Note:** Wait for 10 min to let the processes initialize the system.
Setting Up a New User for the ASR 5000 in the Collectors

To set up a new user for the ASR 5000 in the Collectors:

1. Log on to the master Collector node:

```
host [cluster : master]
```

2. Create the user:

```
username user-id password password
write memory
```

Where the username and password should be the same ones configured for EDR and bulkstats files transfer on the ASR 5000.

3. Return to _shell:

```
_shell
```

4. Edit /etc/ssh/sshd_config to set the following parameters, as indicated:
5. Return to the configuration terminal:

```
admin@host> UsePAM no
PasswordAuthentication yes
```

6. Restart the process:

```
admin@host# en
admin@host> conf t
```

```
host [cluster : master|standby](config)> pm process sshd restart
host [cluster : master|standby](config)> _shell
```
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 Cisco 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]# hadoop dfsadmin -report
   ```

   The resulting output may resemble:

   ```
   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: 10.10.2.18:50010
   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%
   ```
Like the output shown above, a Compute node that is down or is not accessible shows values of zero for:

- DFS Used
- Non DFS Used
- DFS Remaining

**Removing the Faulty or Inaccessible Node From Cluster**

1. Log into the master Name (Collector) node and go to the pmx hadoop sub-shell:

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   host [cluster : master|standby](config)# pmx subshell 
   subshell-name
   ```

2. Run this command, where `ip-address-ctrl-interface` is the IP address of the Compute node control interface.

   ```
   pm extension (hadoop)> remove slave ip-address-ctrl-interface
   ```

**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 all UCS nodes:

   ```
   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:

```bash
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:

```bash
host [cluster : master|standby](config)# image fetch
scp://admin@192.168.147.18/data/mfgcd-guavus-x86_64-20130913-203452.iso
host [cluster : master|standby](config)# image mount mfgcd-guavus-x86_64-20130827-085941.iso
```

The resulting output may resemble:

```text
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 30 minutes to get all the blades manufactured with image 3.4.

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**

To configure either the original master or the standby node:

1. 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
   ```

2. Start the installation on the Compute node:

   ```
   host [cluster : master|standby](config)#
   install appliance mural.xml cluster cluster-name ComputeNode-CLUS Compute-GMS-1
   ```

   Where:
MURAL Operations and Troubleshooting Guide

- **ComputeNode-CLUS** is the name of the cluster used while configuring GMS. It comes up automatically when you press tab after typing cluster-name.

- **Compute-GMS-1** is the name of the Compute node specified in the mural.xml file. This name is read from the mural.xml file and should come up automatically when you press tab.

3. Monitor the installation status:

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

<table>
<thead>
<tr>
<th>ComputeNode-CLUS</th>
<th>Compute-GMS-1</th>
</tr>
</thead>
</table>

Wait for the message `Node successfully installed` before proceeding.

4. Apply any patches that have been released for this build. See the MURAL Release Notes for a list of patches and installation instructions.

### 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. Add the newly configured Compute node to HDFS:

```
pm extension (hadoop)> add slave 10.10.2.18
```

Where 10.10.2.18 is the IP address of the Compute node control interface. Wait 10 minutes for processes to come up.
4. 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
```

5. Verify that the Compute node was added to the cluster:

```
host [cluster : master]# hadoop 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. There are different actions depending on whether it is functioning as a master or standby; both are addressed in this section.

If any of these steps fail, do not proceed. Instead, contact Cisco Technical Support to resolve the issue.

Checking Status of Insta Nodes

If a blade fails, regardless of whether the node is functioning as a master or standby, the initial action is the same.

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.

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 all UCS nodes:

```
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-guavus-x86_64-20130913-
  203452.iso
host [cluster : master|standby](config)# image mount mfgcd-
  guavus-x86_64-20130827-085941.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 net-
work 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 30 minutes to get all the blades manufactured with image 3.4.

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
   ```

   **Note:** This command will take approximately 5-10 minutes to stop all jobs.

   b. Log into the master UI node and stop the tomcat 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
- bulkstats
- reportAtlas
- rge
- httperror
- launcher

c. Repeat this on the standby UI node.
d. Log into the master Rubix node and stop processes:

```
(config)# pm process rubix terminate
(config)# write mem
```
e. Repeat this on the standby Rubix node.

**Configuring the New Insta Node**

To configure either the original master or the standby node:

1. 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
```

2. 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.

3. Monitor the installation status:
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](config)#
install appliance show installation-status cluster INSTA-CLUS
```

**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
```

4. Apply any patches that have been released for this build. See the MURAL Release Notes for a list of patches and installation instructions.

5. Log into each of the Insta nodes and run the following commands to ensure that the Insta process is up and running:

   a. Verify that all required instances are up and running:

```
host [cluster :
master|standby]# cli -t "en" "conf t" "insta infinidb
get-status-info" | grep "RUNNING" | wc -l
```

   If this command returns a value of three, all required instances are up and running.

   b. This command should return status as running:
c. Check the status of postgres on both nodes. It should be up and running.

```
host [cluster : master|standby]> en
host [cluster : master|standby]# _shell
admin@host# ps -ef |grep postmaster
```

If any of these steps fail, do not proceed. Instead, contact Cisco 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:

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   host [cluster : master|standby](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. 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
- launcher

**Note:** Wait for two minutes between starting each process.

4. Write changes to memory:

host [cluster : master] (config)# write mem

5. Repeat the above steps on the standby Rubix node.
Recovering a Failed UI Node: Scenario 1

The Rubix node in a cluster has gone down due to a hardware failure and 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 any of these steps fail, do not proceed. Instead, contact Cisco 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 all UCS nodes:

   ```
   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-guavus-x86_64-20130913-203452.iso
host [cluster : master|standby](config)# image mount mfgcd-guavus-x86_64-20130827-085941.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 30 minutes to get all the blades manufactured with image 3.4.

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. 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
   ```

2. 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.

3. 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.**
4. Apply any patches that have been released for this build. See the MURAL Release Notes for a list of patches and installation instructions.

**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 Collector node:

   Log into the GMS and go to the _shell prompt:

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# _shell
   host [cluster : master|standby]# vi /var/home/root/.ssh/known_hosts
   ```

2. Log into the master Collector node and go to the pmx bulkstats subshell:

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

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

5. Push all IBs:

   ```
   pm extension (aggregation_center)> push all ibs mgmt-ip-of-restored-rubix-node
   pm extension (aggregation_center)> quit
   pm extension> quit
   ```
Pushing Certificates to the New Node

If single certificate files were originally installed on the UI Nodes, restore the files on the restored UI node.

1. Log into the master UI node and copy the keystore to the restored UI standby node:

```
> en
# _shell
# scp /data/apache-tomcat/apache-tomcat-7.0.27/keystore
admin@control-ip-of-restored-rubix-node:/data/apache-tomcat/apache-tomcat-7.0.27/
```

Where control-ip-of-restored-rubix-node is _____.

2. Set the correct keystore password. Log into the master UI node and check the password:

```
host [cluster : master]> en
host [cluster : master]# _shell
admin@host# cat /data/apache-tomcat/apache-tomcat-7.0.27/conf/server.xml | grep "keystorePass"
keystoreFile="keystore" keystorePass="rubix123"
```

3. Log into the restored standby UI node and edit the file to set the password to the same as on the master UI node determined in the previous step:

```
host [cluster : standby]> en
host [cluster : standby]# _shell
admin@host# vi /data/apache-tomcat/apache-tomcat-7.0.27/conf/server.xml
```

4. Specify the same password as above in the file for keystorePass. Modify the ciphers with SSL_RSA_WITH_RC4_128_MD5, SSL_RSA_WITH_RC4_128_SHA and save the file:

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
maxThreads="150" scheme="https" secure="true"
```
keystoreFile="keystore" keystorePass="rubix123"
clientAuth="false" ciphers=" SSL_RSA_WITH_RC4_128_MD5, SSL_RSA_WITH_RC4_128_SHA" sslProtocol="TLS" />

Starting Processes on Standby UI Node

1. Log into the standby UI node and restart the Rubix process:

```plaintext
dir [cluster : standby]> en
dir [cluster : standby]# conf t
dir [cluster : standby](config)# pm process rubix restart
```

2. Start tomcats:

```plaintext
dir [cluster : master|standby](config)# rubix modify-app
    process-name enable

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

Where `process-name` is replaced with:

- `atl`as
- `reportAtlas`
- `bulkstats`
- `rge`
- `httperror`
- `launcher`

**Note:** Wait for two minutes between starting each process.

3. Write changes to memory:

```plaintext
dir [cluster : standby](config)# write mem
```

Recovering a Failed UI Node: Scenario 2

One UI node in a cluster has rebooted due to a power failure but it comes up afterwards.
1. Log into the the rebooted UI node and restart the Rubix process:

```plaintext
code
host [cluster : master|standby]> en
host [cluster : master|standby]# config terminal
host [cluster : master|standby](config)# pm process rubix restart
```

2. Start tomcats:

```plaintext
code
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
- launcher

**Note:** Wait for two minutes between starting each process.

3. Write changes to memory:

```plaintext
code
(config)# write mem
```
Recovering from a Single Node Failure
Restarting UI Processes After a UI Node Reboots

This topic describes how to restart UI nodes after an involuntarily reboot or a power failure.

Failover of UI nodes is not supported:

- If the master UI node becomes standby after a reboot, you must reset the node to be master again.
- You must manually restart tomcat processes if either the standby or master UI reboots.

In the MURAL environment, the UI node hosts both the RGE (Report Generation Engine) and the Cube (or Rubix) engine. The RGE responds to HTTP requests from the user interface, whereas the Cube engine pre-fetches and locally caches data and forwards requests to the Insta node.

Restarting Tomcat Instances on Standby UI Nodes

In certain circumstances, such as after a power failure, you must manually restart the tomcat processes on the UI nodes. Restarting the tomcat process on the master UI node is covered in the following sections.

To restart the tomcat process:

1. Log into the standby UI node and go to _shell:

   host [cluster : standby]> en
   host [cluster : standby]# _shell

2. Start the *tomcat instance* apache-tomcat by running:

   host [cluster : master|standby]# cd /data/tomcat-instance/apache-tomcat-7.0.27/bin/
   host [cluster : master|standby]# ./startup.sh

3. Re-run the previous step for the *tomcat-instance* apache-tomcat2.

Verifying That the Master Node Rebooted

Before you begin, verify that the original master UI node has rebooted.
Note: If a UI node has failed, see "Recovering from a Single Node Failure" on page 1.

To confirm which UI node has involuntarily rebooted:

1. Log into the virtual IP of the UI cluster using an SSH connection and go to _shell:

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

2. Verify that the HSQLDB process is not running:

```
host [cluster : master|standby]# ps -ef | grep -i hsql | grep -v grep
```

The resulting output may resemble:

```
admin 2817 1 0 May16 ? 00:15:10 java -cp /opt/tms/rubix-atlas3.2/WEB-INF/lib/hsqldb-2.2.9.1.jar org.hsqldb.Server --props /opt/tms/rubix-atlas3.2/WEB-INF/classes/server.properties --silent true --trace true
```

If the output does not include the hsqldb process as shown above, the master UI node has rebooted and you should complete the following procedure to restart UI processes.

**Restarting UI Processes on the Master UI Node**

To restart UI processes after a reboot of the master UI node:

1. Wait for the master UI node to come up as the new standby.

2. Reboot the current master UI node to make the original master the master again.

3. Log into the virtual IP of the UI (Rubix) cluster and verify that the original master node (the one which had first rebooted) is again the master UI node:
MURAL Operations and Troubleshooting Guide

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf host [cluster : master|standby](config)# show cluster global brief

The resulting output may resemble:

Global cluster state summary
=================================
Cluster ID: COL-CLUS-2
Cluster name: COL-CLUS-2
Management IP: 10.10.2.23/24
Cluster master IF: eth3
Cluster node count: 2
ID Role State Host External Addr Internal
---------------------------------------------
1* master online GMS2-2 192.168.147.19 10.10.2.19
2 standby online GMS2-1 192.168.147.17 10.10.2.17

4. Log into the master UI node, go to _shell, and change your location to the /data folder:

```
host [cluster : master]> en
host [cluster : master]# _shell
[admin@host ~]# cd /data
```

5. Start HSQLDB:

```
[admin@host data]# nohup java -cp /opt/tms/rubix-atlas3.2/WEB-INF/lib/hsqldb-2.2.9.1.jar org.hsqldb.Server --props /opt/tms/rubix-atlas3.2/WEB-INF/classes/server.properties --silent false --trace true &
```

6. Save the password in the HSQLDB database on the master UI node:

```
[admin@hostdata]# java -jar /opt/tms/rubix-atlas3.2/WEB-INF/lib/sqltool-2.2.9.1.jar --driver=org.hsqldb.jdbcDriver test
```
7. Log into the master UI node of the cluster and go to _shell:

```sql
alter user SA SET password "gU@9us";
commit;
\q
```

Start **tomcat-instance** *apache-tomcat*:

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

8. Repeat the previous step on the:

- standby UI node for instance *apache-tomcat*
- standby UI node for instance *apache-tomcat2*
- master UI node for instance *rge/apache-tomcat*
- master UI node for instance *apache-tomcat-bulkstats*

**Note:** Wait at least 5 minutes between starting *rge/apache-tomcat* and *apache-tomcat-bulkstats*. 
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 Cube (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 under "Recovering from a Single Node Failure" on page 1.

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 from a Single Node Failure" on page 1, 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 an Insta 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 cluster-name DN-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:
   a. Now follow the below sections on both Collector and Compute nodes by referring to the MURAL Installation Guide:
      - Make Performance Related Modifications
      - Check the Processes are running
      - Generate and Push the Information Bases
   b. Follow the section on verifying the blades and nodes have the most recent patches applied according to the MURAL Release Notes.

**Starting Node Processes**

1. Start sending data to the MURAL platform, if the timestamp on the data pushed to the MURAL platform is greater than or equal to the current date and time, and is not an old time stamp.

2. Start the ASR data feed.

3. Send the EDR and bulk stats data feeds to the MURAL Platform. If ASR is used as an input, note the start time from the filename that is created in the Collector folder as mentioned in input.txt. 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:
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 --verbose
   ```

7. Execute the Set Job Time Script for both the master and standby Collector node.


### Start the Data Processing

1. Start oozie jobs:

   a. Log into the master Collector node and go to the pmx oozie subshell:

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

   ```
   host [cluster : master|standby]# mount -o remount,rw /
   ```

   ```
   host [cluster : master|standby]# cd /opt/deployment/Mural_setStartTime/
   ```

   ```
   host [cluster : master|standby]# ./setOozieTime --dataStartTime data-start-time --node collector-mgmt-ip --password admin-user-pwd --verbose
   ```

   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.

2. If recovering from multiple Insta UI node failures, complete sub-steps are provided here:

a. Log into the master Collector node and go to the pmx oozie subshell:

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

b. Stop CubeExporter Jobs:

```
pm extension (subshell-name)> stop jobname job-name
```

Where job-name is:

- CubeExporter
- CubeExporterDaily
- CubeExporterWeekly
- CubeExporter4Weekly

c. Log into each UI Node.

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

d. Stop the processes atlas, reportAtlas, bulkstats, and rge:

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

e. Log into GMS node and activate the XML which was used for installing the setup:

```
admin@host> en
admin@host# conf t
admin@host(config)# gms config mural.xml activate
```
3. 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 cluster cluster-name INSTA-CLUS-GMS
   ```

4. Generate IBs from Collector master for both EDR and bulkstats.
   a. Change the port to 22222 for pushing bulkstats IB’s to Insta node:
      
      **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
      ```

   b. Go to the pmx bulkstats subshell:

      ```
      host [cluster: master|standby]> en
      host [cluster: master|standby]# conf t
      host [cluster: master|standby](config)# pmx subshell subshell-name
      ```

   c. Generate IBs and push them to required nodes:

      ```
      pm extension (subshell-name)> generate all ibs
      pm extension (subshell-name)> push all ibs
      ```

   d. Change the port to 11111 for pushing EDR IBs to the Insta node.
**Note:** Go to _shell and invoke a new CLI shell before changing the port. Otherwise the port change does not take effect.

```bash
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
```

e. Go to the pmx aggregation_center subshell:

```bash
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# pmx subshell subshell-name
```

f. Generate IBs, and push them to required nodes:

```bash
pm extension (subshell-name)> generate all ibs
pm extension (subshell-name)> push all ibs
```

5. Once above operations return success, start pushing data from Collector.

Start the CubeExporter jobs from master Collector so that processed mapreduce data gets pushed to Insta.

a. Log into the master Collector node and go to the pmx oozie subshell:

```bash
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# pmx subshell subshell-name
```

b. Start CubeExporter Jobs:

```bash
pm extension (subshell-name)> run jobname job-name
```

Where `job-name` is:
6. 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. Log into GMS node:

```
admin@host> en
admin@host# conf t
```

2. 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
```

3. 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 cluster cluster-name RUBIX-CLUS-GMS
```

4. 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:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
```
b. Push the IBs from the master Collector node to the two UI nodes.

Run the following commands on the pmx aggregation_center subshell, then backout and run it on the bulkstats subshell:

<table>
<thead>
<tr>
<th>pm extension (bulkstats)</th>
<th>pm extension (bulkstats)</th>
</tr>
</thead>
<tbody>
<tr>
<td>push all ibs Control-IP-UI-NODE-1</td>
<td>push all ibs Control-IP-UI-NODE-2</td>
</tr>
</tbody>
</table>

Replacing Control-IP-UI-NODE-1 and Control-IP-UI-NODE-1.

c. If you use a browser other than Safari, you must accept certificates by accessing these URLs in following order prior to attempting the next step:

- Bulkstats (https://<domainName>:20443/)
- RGE (https://<domainName>:30443/)

d. Reinstall single certificate files, following the steps in MURAL Installation Guide, to access EDR, bulkstats and RGE on UI.

e. This can be avoided by putting production certificates on UI nodes Tomcat for all three Tomcat instances - bulkstats, RGE and bulkstats.

See details in the Mural Installation Guide on how to generate and install the certificate. Use that as a reference to get certificates from an authorized Certificate Signing authority like Verisign Inc.

f. Start UI processes and verify data following the steps in the section of the same name under "Recovering from a Single Node Failure" on page 1.
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 Cisco 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 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:

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# pmx subshell
subshell-name
```

2. Stop all jobs:

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

**Note:** This command will take approximately 5-10 minutes to stop all 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 into the master UI node and go to `_shell`: 

---

Copyright © 2014, Cisco Systems, Inc. 261
5. What is this command?

```
[admin@host]# ps -ef | grep tomcat | grep -v grep | awk '{print $2}' | xargs -I {} kill -9 {}
```

6. Stop all tomcats by running:

```
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
- bulkstats
- reportAtlas
- rge
- httperror
- launcher

7. 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. This LUN will be shared between both the Insta nodes.

2. Reboot both the Insta nodes (master and standby) individually:

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

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

```bash
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 which failed.

   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.
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. 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]> en
   host [cluster : master|standby]# config t
   host [cluster : master|standby](config)# gms config Insta Storage Failure.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,
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
cluster cluster-name INSTA-CLUS-GMS force-format
```

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 "Back ing Up and Restoring a Large (TB) Insta Database" on page 170

**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. Change the port for bulkstats to 22222.

   **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. Go to the pmx bulkstats subshell:

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   ```
3. Provide the control network IP addresses of the master Collector node, standby Collector node, and all UI nodes:

```
mxm extension (bulkstats)> add ib_destination IP-Address
```

Where IP-Address is the network IP address of nodes you are adding to the IB database.

4. Generate IBs on the pmx bulkstats subshell:

```
mxm 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 generated IBs to required nodes:
pm extension (bulkstats) > **push all ibs**

6. Log into the standby Collector node and go to the pmx bulkstats subshell:

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

7. Fetch IBs from the inbox:

```
pm extension (subshell-name) > fetch all ibs from inbox
```

**EDR Configurations**

1. Log into the master Collector node 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. Go to the pmx aggregation_center subshell.

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

3. Generate IBs and push them to required nodes:

```
pm extension (subshell-name) > generate all ibs
pm extension (subshell-name) > push all ibs
```

4. Log into the standby Collector node and go to the pmx aggregation_
5. Fetch IBs from the pmx aggregation_center subshell inbox:

```
pm extension (subshell-name)> fetch all ibs from inbox
```

### Restart Jobs and Processes

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

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

2. Start all jobs:

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

3. Log into the master UI node and go to the configure terminal:

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

4. Start all 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
Note: Wait for two minutes between starting each process.

5. Log into the standby UI node and repeat the above steps to restart the tomcat instances.
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 Cisco 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 ComputeRubix 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
   - bulkstats
   - reportAtlas
   - rge
MURAL Operations and Troubleshooting Guide

- httperror
- 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 below order:

**Note:** Wait for two minutes between starting each process.

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:

- bulkstats
- reportAtlas
- rge

d. Log into the standby UI node and repeat the previous step.

7. Access the UI's by going to the URL https://<domainName>:21443/
   through your browser.

8. The domain name to be used is the one which was provided at the time of
   initial configuration via GMS for the UI nodes configuration details.
Restarting Bulkstats After Updating Start Time

The data start time is the time from which data is sent to the system. If you adjust the start time, you must restart the bulkstats related jobs.

If any of these steps fail, do not proceed. Instead, contact Cisco Technical Support to resolve the issue.

Stopping Bulk Stats Jobs

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

   ```
   host [cluster : master|standby]> en
   host [cluster : master|standby]# conf t
   host [cluster : master|standby](config)# pmx subshell
   subshell-name
   ```

2. Run this command for each bulkstats job:

   ```
   pm extension (oozie)> stop jobname job-name
   ```

   Where job-name is:

   - BulkStat
   - BSAgg15min
   - BSAgg1hour
   - BSAgg1Day
   - BSAgg1Week
   - BSAgg4Week
   - BulkStatExporter_15min
   - BulkStatExporter_1Day
   - BulkStatExporter_1Week
   - BulkStatExporter_4Week

Sending Input Files

Start sending bulkstats files from the ASR to the system every 5 minutes. Let it run for at least an hour. Reference the MURAL Installation Guide for specific guidance on how to perform this step.
Before proceeding, ensure that you have fed continuous input data for at least 1 hour to the system.

**Updating the Start Time**

1. Log into the GMS and go to the `shell`:

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

2. Run:

```bash
[admin@host]# cd /opt/deployment/Mural setStartTime
[admin@host dir]#. /setOozieTime --dataStartTime data_start_time --node Collector_IP --password admin-password --manual --verbose
```

Where:

- `data_start_time` is the time from which you start sending data to the system in the format **YYYY-MM-DDThh:mmZ**. For example, if data is coming to the system from 13th May 10:00, then value would be **2013-05-13T10:00Z**.

- Minutes are in multiples of 5.

- `Collector_IP` is the management IP address of the master or standby Collector Node.

- `Admin-password` is the Collector node administrator password.

- Manual mode is only for selecting bulkstats related jobs

For example,

```bash
[admin@host dir]#. /setOozieTime --dataStartTime 2013-10-16T05:002 --node 192.168.147.14 --password admin@123 --manual --verbose
```

3. Repeat step 1 for the standby Collector node which will display a long list of jobs.

4. Enter 51–60 to execute bulkstats specific commands:
Select the command numbers to execute. Press Q to quit. (ex. ALL / 1-10,16-22 / 1,2,5,7,8,11 / 31,34-40,45,56-60) [ALL] : 51-60

5. The commands you selected will be listed individually. You will be prompted to confirm your selection. Enter a y to continue:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>51.</td>
<td>pmx subshell oozie set job BSAgg15min attribute jobStart 2013-10-16T05:15Z</td>
</tr>
<tr>
<td>52.</td>
<td>pmx subshell oozie set job BSAgg1Day attribute jobStart 2013-10-17T00:00Z</td>
</tr>
<tr>
<td>53.</td>
<td>pmx subshell oozie set job BSAgg1Week attribute jobStart 2013-10-17T00:00Z</td>
</tr>
<tr>
<td>54.</td>
<td>pmx subshell oozie set job BSAgg1hour attribute jobStart 2013-10-16T06:00Z</td>
</tr>
<tr>
<td>55.</td>
<td>pmx subshell oozie set job BSAgg4Week attribute jobStart 2013-11-07T00:00Z</td>
</tr>
<tr>
<td>56.</td>
<td>pmx subshell oozie set job BulkStat attribute jobStart 2013-10-16T05:05Z</td>
</tr>
<tr>
<td>57.</td>
<td>pmx subshell oozie set job BulkStatExporter_15min attribute jobStart 2013-10-16T05:15Z</td>
</tr>
<tr>
<td>58.</td>
<td>pmx subshell oozie set job BulkStatExporter_1Day attribute jobStart 2013-10-17T00:00Z</td>
</tr>
<tr>
<td>59.</td>
<td>pmx subshell oozie set job BulkStatExporter_1Week attribute jobStart 2013-10-17T00:00Z</td>
</tr>
<tr>
<td>60.</td>
<td>pmx subshell oozie set job BulkStatExporter_4Week attribute jobStart 2013-11-07T00:00Z</td>
</tr>
</tbody>
</table>

Kindly confirm. Do you want to execute above commands on node 192.168.147.14 (y/n) [n]: y

Launching connection with 192.168.147.14 ...
Restart the Jobs

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

```
host [cluster : master|standby]> en
host [cluster : master|standby]# conf t
host [cluster : master|standby](config)# pmx subshell subshell-name
```

2. Run this command for each bulkstats job:

```
host [cluster : master|standby](config)#
run job job-name
```

Where job-name is:

- BulkStat
- BSAgg15min
- BSAgg1hour
- BSAgg1Day
- BSAgg1Week
- BSAgg4Week
- BulkStatExporter_15min
- BulkStatExporter_1Day
- BulkStatExporter_1Week
- BulkStatExporter_4Week

3. After 30 min, follow steps in the MURAL Installation Guide to check the bulk-stats related jobs done.txt on the:

- Collector
- Insta bulkstats database
- MURAL UI