Contents

[1 Table of Contents 1](#_Toc354048180)

[3 Introduction 2](#_Toc354048181)

[4 Installation 3](#_Toc354048182)

[4.1 Prerequisites 3](#_Toc354048183)

[4.1.1 Tibco Silver Fabric 3](#_Toc354048184)

[4.1.2 Engine Operating System 3](#_Toc354048185)

[4.1.3 Hadoop 1.0.4 Distribution Gridlib 3](#_Toc354048186)

[4.1.4 Engine Host Configuration 3](#_Toc354048187)

[5 Configuration Guidelines 3](#_Toc354048188)

[5.1 Name Node Persistence 3](#_Toc354048189)

[5.1.1 Namenode Host Designation 3](#_Toc354048190)

[5.1.2 Name Directory Persistence 4](#_Toc354048191)

[5.1.3 Managed Decommissioning 4](#_Toc354048192)

[5.1.4 RackAwareness Configuration Policies 5](#_Toc354048193)

[5.1.5 CONFIG\_FILE Policy 6](#_Toc354048194)

[5.1.6 ENV\_VARIABLE Policy 6](#_Toc354048195)

[5.2 Automatic ReBalancing 6](#_Toc354048196)

[5.3 UI Routing 7](#_Toc354048197)

[5.4 Customizing Hadoop Parameters 8](#_Toc354048198)

# Introduction

The Hadoop Enabler is actually a set of related enablers designed to support various processes implemented within the Hadoop installation. This includes both of Hadoop’s primary sub-projects, Hadoop File Sytem (HDFS) and MapReduce.

The following individual enablers are included:

|  |  |
| --- | --- |
| Enabler Name | Process Enabled |
| Namenode | The HDFS Namenode daemon |
| Secondary Namenode | The HDFS Secondary Namenode daemon |
| Datanode | The HDFS Datanode daemon |
| Jobtracker\* | The MapReduce Jobtracker daemon |
| Tasktracker\* | The MapReduce Tasktracker daemon |
| Balancer | The HDFS Balancer script |
| Psuedo Cluster | An single-host Hadoop *Psuedo Cluster* as implemented by start-all.sh script in the Hadoop install. |

\* Only works with version of MapReduce version 1 (e.g. 1.1.2).

Note: YARN (a.k.a. Map Reduce version 2 *or* MRv2) is not currently supported by this enabler.

This enabler set allows you to instantiate Hadoop daemons in different combinations to create the cluster you require. For example you may elect to instantiate just the HDFS enablers without the MapReduce enablers.

In addition to enabling Hadoop’s out-of-the box processes, this enable set also automates or simplifies some of the common administrative tasks of a Hadoop cluster.

|  |  |
| --- | --- |
| Automated Re-Balancing | When one or new Datanodes are added, the Balancer will automatically be invoked to distribute data to the new node(s) |
| Managed Decommissioning | When stopping datanodes on an active cluster, data blocks are automatically offloaded to other nodes before shutdown is complete. |
| RackAwareness Configuration Policies | Allow the users to control RackAwareness via pre-built, configurable policies as opposed to developing their own scripts. |
| UI Routing | Maps well-known and persistent URLs to Hadoop Web User Interfaces so that Hadoop processes may be moved between nodes without disrupting clients |

# Installation

## Prerequisites

### Tibco Silver Fabric

This Enabler was originally developed and tested on Silver Fabric 5.0.1.

### Engine Operating System

The Hadoop enabler was originally built and tested on Linux.

It might work on similar OS’s but the current version specifically will not work on Windows.

### Hadoop Distribution

A Silver Fabric gridlib containing the Hadoop version you wish to use must be created and copied onto the Broker nodes at [Fabric Install Directory]/webapps/livecluster/deploy/resources/gridlib/.

This enabler supports multiple versions of Hadoop including but not limited to

1.0.4, 1.1.2 0.23.7 and 2.03-alpha. Note that for builds that contain MRv2 (YARN) and HA Namenodes that these features are not supported by the current enabler.

See the Readme in the [Hadoop Enabler Project Directory]/distribution for instructions on how to create the distirubtion gridlib.

### Engine Host Configuration

Hadoop requires that hosts be able to communicate via passphraseless SSH. And this Hadoop Enabler assumes that this configuration has been or will be performed prior to run time. See the Hadoop documentation for more information on how to configure passphraseless SSH.

# Configuration Guidelines

## Name Node Persistence

### Namenode Host Designation

When restoring a failed Namenode, Hadoop requires that you use the original IP address and port (see https://issues.apache.org/jira/browse/HDFS-34). In Silverfabric, this means you will want to add Resource Preferences to the Namenode Component for a specific host as well as a specific Engine Instance.

### Name Directory Persistence

By default, the Namenode Enabler will use a local drive location for the Namenode’s Name Directory (the file location where HDFS stores metadata). Use a local drive is optimal for performance, but it also means that the data can be lost in the event of a Namenode Component restart. To ensure that this metadata survives a Namenode restart, Hadoop allows users to specify a second location (e.g. a highly available SAN elsewhere in the datacenter) for storing a second copy of the metadata.

The SilverF Fabric runtimecontext variable for defining this second location is ${hadoop\_enabler\_NAMENODE\_NAME\_DIR\_REMOTE}. When you start you first start your Hadoop cluster, the directory defined by this variable should not exist. The Hadoop Enabler will then create and initialize both the remote and local copies.

**Warning:** It is important during initial restart that the remote directory does not exist. An empty directory at the defined location will confuse Hadoop and cause unpredictable results. If there is a leftover directory from a previous execution of your Hadoop cluster you should delete it manually before starting your Hadoop cluster.

**Note:** A complete loss of the Name Directory’s metadata would mandate a complete reloading your Hadoop cluster. It is best to minimize the chane of any accidental deletion. For this reason, and given that complete Hadoop restarts are intended to be rare, the deletion of any previous Name Directory has been left as a manual task in this version of the Enabler.

### Managed Decommissioning

Hadoop Datanode *decommissioning* process allows administrators shutdown individual datanodes in an orderly fashion and to offload data blocks from the datanode(s) to prevent a loss of data. It requires a number of manual steps on both the datanode and namenode. The Hadoop enabler set automates this process.

When one or more datanoes are stopped or restarted on an active cluster, a request is sent from the Datanode enabler to the Namenode enabler to initiate the decommissioning process. This request is persistent and will survive until the decommissioning process is complete or the entire cluster is shut down. The Namenode enabler will initiate the decommissioning process and monitor it to completion. After that the datanode will be completely stopped and the namenode and datanode enablers will clean up the request queue.

Generally speaking, Managed Decommissioning does not require any special configuration by the user. However, it is important to understand that Hadoop will *not* fully decommission a datanode if doing so would violate it’s replication policies by under-replicating one or more blocks.

For example, if you have 3 active datanodes and the default hdfs replication factor (dfs.replication) is set to 3, then stopping any one of the datanodes would create a policy violation. In this case the datanode you are attempting to stop will enter a status of “Decommissioning in Progress” and will remain there indefinitely. It will never reach the completed status of “Decommissioned”.

With the SF Enabler, the enabler will remain in a Deallocating mode indefinitely or until the Maximum Deactivation period for your component has been reached. To resolve the problem, additional datanodes should be added until Hadoop can successfully offload the replicas.

### RackAwareness Configuration Policies

The Hadoop enablers allows users to pick from a set of pre-defined RackAwareness policies or to create their own policy.

To implement a custom policy, the user should implement the policy in a scipt (see the Hadoop documentation) and override the enabler supplied scripts which is at Hadoop-1.0.4/bin/hadoop\_enabler\_RACKAWARENESS.sh (see the SF documentation for details on how to attach content files to your component definitions).

The pre-defined RackAwareness policies provide default customizable set of behaviors that can be controlled through RuntimeContextVariables. If no configuration is supplied, all nodes will be assigned to a default rack

|  |  |
| --- | --- |
| RackAwarenes Policy | Description |
| CONFIG\_FILE | Sets the location based on a user configuration files |
| ENV\_VARIABLE | Sets the location based on an environment variable from the host |
| DEFAULT | Sets the location of all nodes to a default value |

The following variables on the Namenode Enabler are used to control the Rackawareness policy definition.

|  |  |  |
| --- | --- | --- |
| Variable | Description | Default Value |
| hadoop\_enabler\_RACKAWARENESS\_POLICY | The nameo of the policy to be used. | DEFAULT |
| hadoop\_enabler\_RACKAWARENESS\_CONFIG\_FILE | The configuration file to be used with the CONFIG\_FILE policy. | [Hadoop Install]/conf/ hadoop\_enabler\_RACKAWARENESS.cfg |
| hadoop\_enabler\_RACKAWARENESS\_DEFAULT | The default value to be used with the DEFAULT POLICY or when another policy is unable to create a value. | /default/rack |
| hadoop\_enabler\_RACKAWARENESS\_ENV\_VARIABLE | The environment variable on the remote hosts to be used with ENV\_VARIABLE policy. | RACK |

#### CONFIG\_FILE Policy

When the RackAwareness Policy is set to CONFIG\_FILE (the default), the Hadoop Enabler attempts to retrieve the rack value from a file. The file contains a list of regular expressions followed by the rack value to be used. For example:

###################################################

# Hadoop Rackaware Configuration

####################################################

192.168.247.132 /dc1/rack1

192.168.247.133 /dc1/rack1

192.168.247.134 /dc1/rack1

192.168.247.\* /dc2/rack2

192.168.248.\* /dc2/rack3

If multiple matches exist, the first one located, starting from the top of the file will be used. If no match is found the default value will be used.

#### ENV\_VARIABLE Policy

When the RackAwareness Policy is set to ENV\_VARIABLE, the Hadoop Enabler attempts to retrieve the rack value from a environment variable set on the slave node.

The environment variable to be used must be setup such that it is visible to the SF user when logging in via an SSH session.

If no match is found the default value will be used.

## Automatic ReBalancing

The Balancer Enabler is designed to recognize certain conditions where a rebalancing is recommended and automatically run Hadoop’s Balancer script. Specifically it executes the Balancer script whenever one or more datanodes are added to the cluster. If a new datanode is added to the cluster while a Balancer is already running, the Balancer Enabler will automatically halt and re-intiate the Balancer Script. And if datanodes are added while the Balancer Enabler is off line, the Balancer script will be executed as soon the Balance Enabler is brought back on-line.

A Balancer Component should configured in SF with *component dependency* (see SF documentation) on the cluster’s Namnode Component. It is not necessary to deploy the Balancer Component on any specific hosts. However, it is recommended to attach Resource Preferences (see the Silver Fabric documentation) to the Balancer Component that will at prevent it being deployed on any slave node where it would compete with a datanode and/or tasktracker for resources.

The current version of the Hadoop enabler does not have a scheduling or calendar feature for executing the balancer on a routine basis (e.g. nightly). To also run the balancer on a routine schedule, cron or some other external scheduling tool should be used.

## UI Routing

All Hadoop Daemons include a Web UI and the Enablers that support these daemons use Silver Fabric’s HTTP Support. HTTP Support allows users to associate well-known URL with HTTP enabled UIs within a cloud of transient hosts. Note: If you Master Nodes (Namenode, Secondary Namenode and Jobtracker)are all configured to run on specific nodes and engine instances (see section 3.1.1 Host) this feature may not be necessary.

To enable this feature, the user must enable the “HTTP Support” feature on their Component definition.

For each such daemon process, the enabler will attach a static prefix (e.g. the name namenode is ‘nn’). The user may optionally supply an additional prefix (see the Silver Fabric User’s Guide for more details on configuring HTTP Support). The complete URL mapping for each daemon process will follow the pattern shown in the table below.

|  |  |
| --- | --- |
| Process | Virtual Router Mapping |
| Namenode | http://[Virtual Router][/userprefix]/nn |
| Secondary Namenode | http://[Virtual Router][/userprefix]/sn |
| Datanode\* | http://[Virtual Router][/userprefix]/dn |
| Jobtracker | http://[Virtual Router][/userprefix]/jt |
| Tasktracker\* | http://[Virtual Router][/userprefix]/tt |

\* You will usually want to access this UI by first accessing the master (Namenode or Jobtracker) and then drilling down to the desired slave node. When multiple instances of a slave node exists in a cluster, using the Virtual Router’s direct mapping will result in a routing to a randomly selected slave node, which is not particularly useful in most cases.

## Customizing Hadoop Parameters

The Hadoop enabler provides a RuntimeContextVariables to hold user hadoop property overrides. There is one such variable for each of the Hadoop xml configuration files. For each file there is a set of Reserved Properties that are controlled by the Hadoop enabler and should not be overridden.

|  |  |  |
| --- | --- | --- |
| Config File | User Overrides Variable | Reserved Properties |
| coresite.xml | hadoop\_enabler\_USERPROPS\_CORE | |
|  |  | fs.default.name hadoop.tmp.dir  topology.script.file.name |
| hdfs-site.xml | hadoop\_enabler\_USERPROPS\_HDFS | |
|  |  | dfs.http.address dfs.secondary.http.address  dfs.datanode.address  dfs.datanode.http.address  dfs.datanode.ipc.address  dfs.hosts.exclude  dfs.name.dir |
| mapred-site.xml | hadoop\_enabler\_USERPROPS\_MAPRED | |
|  |  | mapred.job.tracker  mapred.job.tracker.http.address  mapred.task.tracker.http.address |

For example to change the default replication factor and block size you would set ${hadoop.namenode.userprops.hdfs} to:

<property><name>dfs.replication</name><value>2</value></property>

<property><name>dfs.block.size</name><value>16777216</value></property>