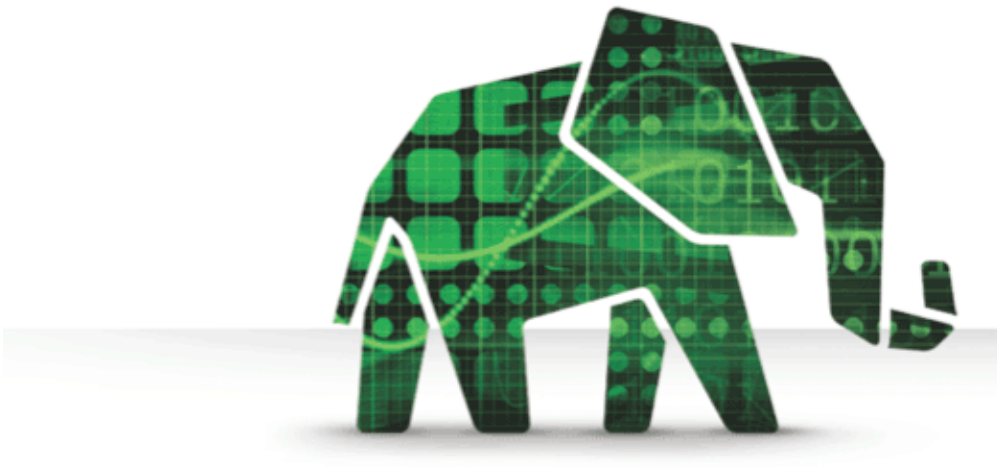
Hortonworks Architecting the Future of Big Data

Hortonworks Technical Preview for Apache Falcon

Released: 11/20/2013



Architecting the Future of Big Data

Welcome to Hortonworks Inc, technical preview for Apache Falcon. The Technical Preview provides early access to upcoming features in the Hortonworks product, letting you test and review during the development process. These features are considered under development.

Although your feedback is greatly appreciated these features are not intended for use in your production systems and not considered Supported by Hortonworks.

Have fun and please send feedback to us on the Community forums:

http://hortonworks.com/community/forums/

Apache Falcon Introduction

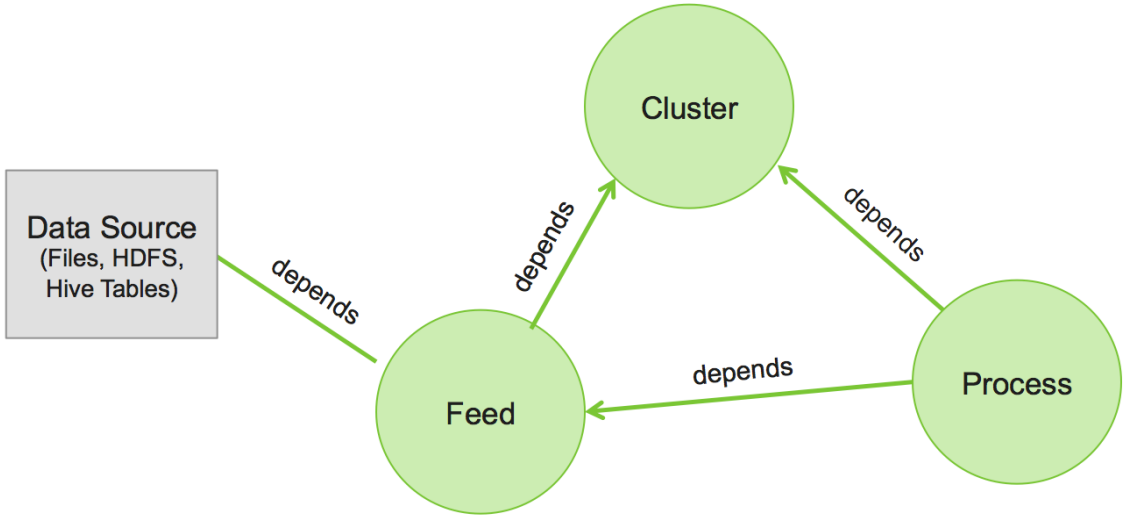
Apache Falcon provides a framework for simplifying the development of data management applications in Apache Hadoop. Falcon enables users to automate the movement and processing of data sets. Instead of hard-coding complex data set and pipeline processing capabilities, Hadoop applications can now rely on the Apache Falcon framework for these functions.

- **Dataset Replication** - Replicate data sets (whether HDFS or Hive Tables) as part of your Disaster Recovery, Backup and Archival solution. Falcon can trigger processes for retry and handle late data arrival logic.
- **Dataset Lifecycle Management** – Establish retention policies for datasets and Falcon will schedule eviction.
- **Dataset Traceability / Lineage** - Use Falcon to view coarse-grained dependencies between clusters, datasets, and processes.

Framework Entities

The Falcon framework defines the fundamental building blocks for data processing applications using entities such as Feeds, Processes, and Clusters. A Hadoop user can establish entity relationships and Falcon handles the management, coordination, and scheduling of data set processing.

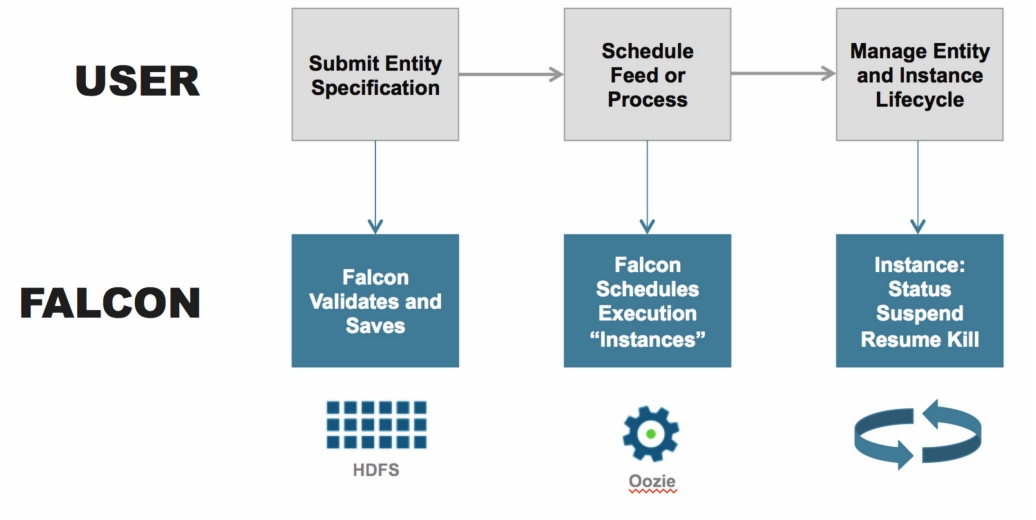
- **Cluster -** Represents the "interfaces" to a Hadoop cluster.
- **Feed –** Defines a dataset with location, replication schedule, and retention policy.
- **Process -** Consumes and processes Feeds.



Usage

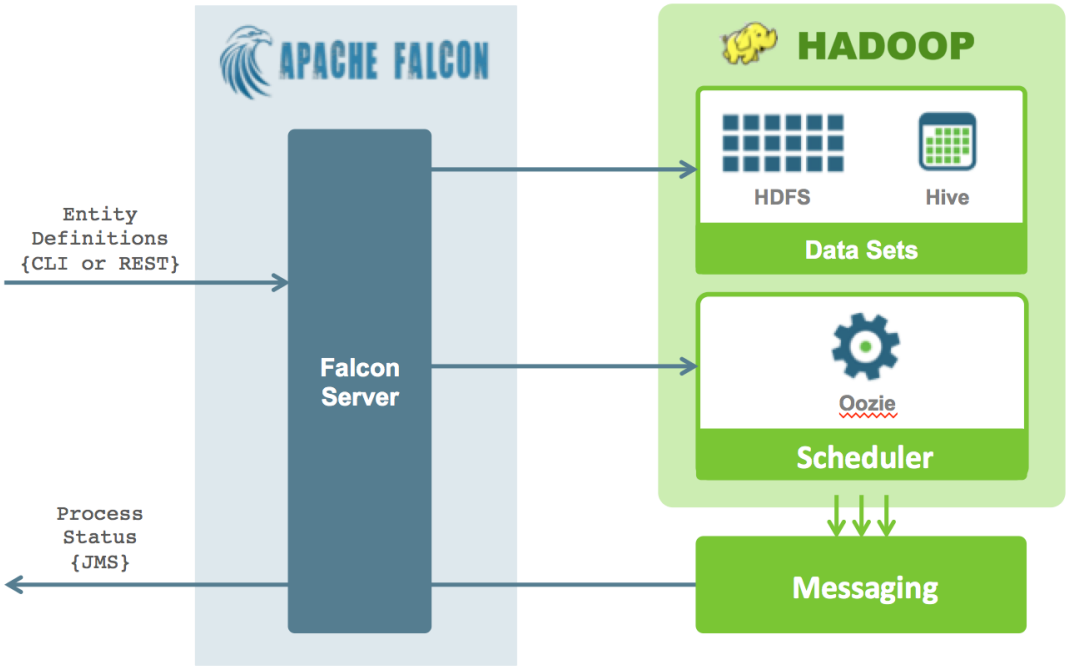
Using the entity definitions, you can create entity specifications (http://falcon.incubator.apache.org/docs/EntitySpecification.html) and submit to Falcon for execution. The high-level Falcon operations are:

1. Submit entity specifications into Falcon for Clusters.
2. Design Feeds and Processes and associate with Clusters.
3. Designate replication schedule and retention policies on the Feeds.
4. Submit entity specifications into Falcon for Feeds and Processes.
5. Instruct Falcon to schedule execution of the Feed and Process entities.
6. Use Falcon to manage instances of execution (status, suspend, resume, kill, re-run).



Architecture

Falcon essentially transforms entity definitions into repeated actions through a workflow scheduler. All the functions and workflow state management requirements are delegated to the scheduler. By default, Falcon uses Apache Oozie for the scheduler.

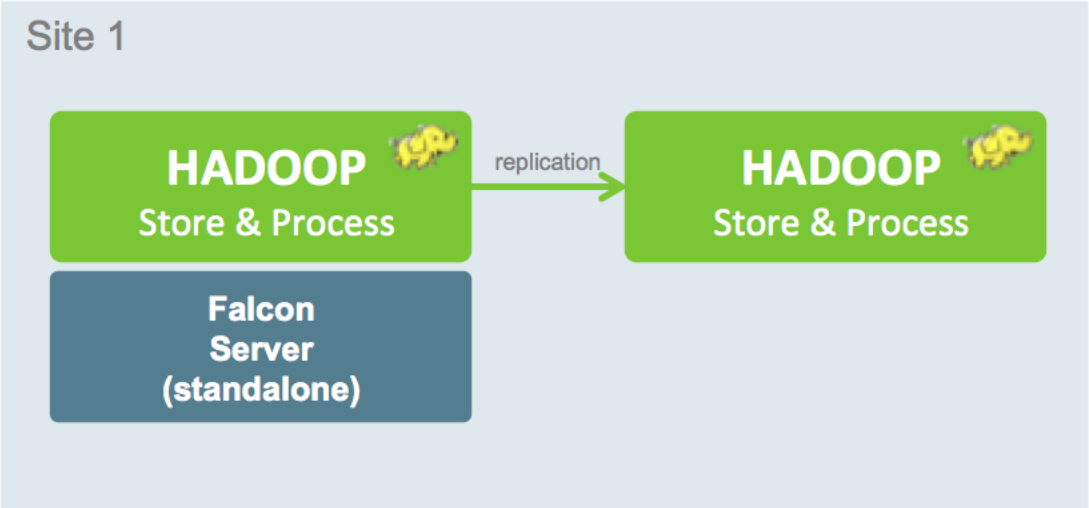


Deployment Options

Falcon can run in either Standalone or Distributed mode. In Standalone mode, a single Falcon Server is installed and managed. This Falcon Server can work with and process data sets for one or more clusters in one or more data centers. If you plan to manage multiple clusters, across data centers, and you plan to have multiple instances of Falcon Server, you can deploy a Falcon Prism Server for distributed management of the Falcon Servers.

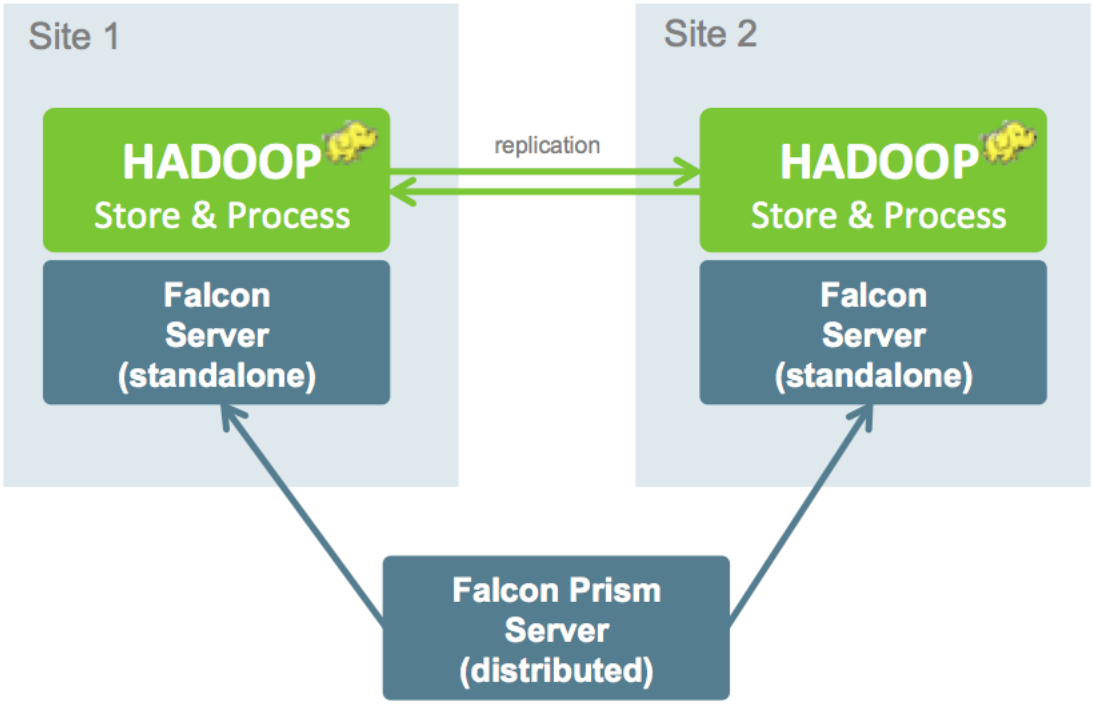
Standalone

When replicating data in the same cluster or a different cluster (in the same colo), you can run a single Falcon Server instance standalone.



Distributed

When running multiple Falcon Server instances across clusters and colos, you can install a Falcon Prism Server for "distributed" Falcon Server instance management.



Utilities and APIs

Whether you are managing the standalone Falcon Servers, or a Falcon Prism Server in a distributed deployment, Falcon provides a Command Line Interface CLI (http://falcon.incubator.apache.org/docs/FalconCLI.html) and a REST API (http://falcon.incubator.apache.org/docs/restapi/ResourceList.html). The CLI and REST API both expose Entity Management, Instance Management, and Admin operations.

System Requirements

The Falcon Technical Preview has the following minimum system requirements:

- Hortonworks Data Platform (HDP)
- Operating Systems
- Software Requirements
- JDK Requirements

Hortonworks Data Platform

Falcon requires HDP 2.0 GA.

Note: You must have Oozie installed and configured on your cluster.

Operating systems

- 64-bit RHEL 6
- 64-bit CentOS 6
- 64-bit Oracle Linux 6

Software Requirements

- yum
- rpm
- wget
- java (see JDK Requirements)

JDK Requirements

Your system must have the correct JDK installed on all hosts that will run Falcon. The following JDKs are supported:

- Oracle JDK 1.7 64-bit
- Oracle JDK 1.6 update 31 64-bit

Installation

Install the Standalone Falcon Server (recommended for first-time Falcon users) or the Distributed Falcon Server on a host that has the Hadoop packages installed. You can run the following command to confirm the Hadoop packages are installed:

```
hadoop version
```

Install Standalone Falcon Server (Recommended)

Complete the following steps to install the Falcon Standalone server:

1. Download the Falcon Server package.

   ```
   wget http://public-repo-1.hortonworks.com/HDP-LABS/Projects/Falcon/2.0.6.0-76/rpm/falcon-0.4.0.2.0.6.0-76.el6.noarch.rpm
   ```

2. Install the Package using RPM.

   ```
   rpm -Uvh falcon-0.4.0.2.0.6.0-76.el6.noarch.rpm
   ```

   The Falcon Server software is installed here:

   ```
   /usr/lib/falcon
   ```

Install Distributed Falcon Prism Server

Complete the following steps to install the Distributed Falcon Prism server:

Note: Do not install the Distributed Falcon Prism Server package on the same host as a Falcon Server.

1. Download the Falcon Distributed package.

   ```
   wget http://public-repo-1.hortonworks.com/HDP-LABS/Projects/Falcon/2.0.6.0-76/rpm/falcon-distributed-0.4.0.2.0.6.0-76.el6.noarch.rpm
   ```

2. Install the Package using RPM.

   ```
   rpm -Uvh falcon-distributed-0.4.0.2.0.6.0-76.el6.noarch.rpm
   ```

   The Falcon Prism Server software is installed here:

   ```
   /usr/lib/falcon-distributed
   ```

Install Falcon Documentation

To install the Falcon project documentation:

1. Download the Falcon Documentation package.

```
wget http://public-repo-1.hortonworks.com/HDP-LABS/Projects/Falcon/2.0.6.0-76/rpm/falcon-doc-0.4.0.2.0.6.0-76.el6.noarch.rpm
```

2. Install the Package using RPM.

```
rpm -Uvh falcon-doc-0.4.0.2.0.6.0-76.el6.noarch.rpm
```

3. Browse to the Falcon Documentation index page:

```
/usr/share/doc/falcon-0.4.0.2.0.6.0-76/index.html
```

Setup

This section covers:

- Configuring Ooozie
- Using the CLI
- Using Basic Commands
- Using the REST API
- Removing Falcon

Configuring Oozie

After installing Falcon, you must make the following configuration changes to Oozie:

1. Add the following properties in **bold** to the `/etc/oozie/conf/oozie-site.xml` file:

```
 <property>
        <name>oozie.service.ProxyUserService.proxyuser.falcon.hosts</name>
        <value>*</value>
 </property>
 <property>
        <name>oozie.service.ProxyUserService.proxyuser.falcon.groups</name>
        <value>*</value>
 </property>
   <property>
   <name>oozie.service.URIHandlerService.uri.handlers</name>
<value>org.apache.oozie.dependency.FSURIHandler,org.apache.oozie.dependency.HCatURIHandler</value>
   </property>
   <property>
       <name>oozie.services.ext</name>
       <value>
           org.apache.oozie.service.JMSAccessorService,
           org.apache.oozie.service.PartitionDependencyManagerService,
           org.apache.oozie.service.HCatAccessorService
       </value>
   </property>

   <!-- Coord EL Functions Properties -->
   <property>
       <name>oozie.service.ELService.ext.functions.coord-job-submit-instances</name>
       <value>
```

```
            now=org.apache.oozie.extensions.OozieELExtensions#ph1_now_echo,
            today=org.apache.oozie.extensions.OozieELExtensions#ph1_today_echo,
            yesterday=org.apache.oozie.extensions.OozieELExtensions#ph1_yesterday_echo,
            currentMonth=org.apache.oozie.extensions.OozieELExtensions#ph1_currentMonth_echo,
            lastMonth=org.apache.oozie.extensions.OozieELExtensions#ph1_lastMonth_echo,
            currentYear=org.apache.oozie.extensions.OozieELExtensions#ph1_currentYear_echo,
            lastYear=org.apache.oozie.extensions.OozieELExtensions#ph1_lastYear_echo,
            formatTime=org.apache.oozie.coord.CoordELFunctions#ph1_coord_formatTime_echo,
            latest=org.apache.oozie.coord.CoordELFunctions#ph2_coord_latest_echo,
            future=org.apache.oozie.coord.CoordELFunctions#ph2_coord_future_echo
        </value>
    </property>

    <property>
        <name>oozie.service.ELService.ext.functions.coord-action-create-inst</name>
        <value>
            now=org.apache.oozie.extensions.OozieELExtensions#ph2_now_inst,
            today=org.apache.oozie.extensions.OozieELExtensions#ph2_today_inst,
            yesterday=org.apache.oozie.extensions.OozieELExtensions#ph2_yesterday_inst,
            currentMonth=org.apache.oozie.extensions.OozieELExtensions#ph2_currentMonth_inst,
            lastMonth=org.apache.oozie.extensions.OozieELExtensions#ph2_lastMonth_inst,
            currentYear=org.apache.oozie.extensions.OozieELExtensions#ph2_currentYear_inst,
            lastYear=org.apache.oozie.extensions.OozieELExtensions#ph2_lastYear_inst,
            latest=org.apache.oozie.coord.CoordELFunctions#ph2_coord_latest_echo,
            future=org.apache.oozie.coord.CoordELFunctions#ph2_coord_future_echo,
            formatTime=org.apache.oozie.coord.CoordELFunctions#ph2_coord_formatTime,
            user=org.apache.oozie.coord.CoordELFunctions#coord_user
        </value>
    </property>

    <property>
        <name>oozie.service.ELService.ext.functions.coord-action-create</name>
        <value>
            now=org.apache.oozie.extensions.OozieELExtensions#ph2_now,
            today=org.apache.oozie.extensions.OozieELExtensions#ph2_today,
            yesterday=org.apache.oozie.extensions.OozieELExtensions#ph2_yesterday,
            currentMonth=org.apache.oozie.extensions.OozieELExtensions#ph2_currentMonth,
            lastMonth=org.apache.oozie.extensions.OozieELExtensions#ph2_lastMonth,
            currentYear=org.apache.oozie.extensions.OozieELExtensions#ph2_currentYear,
            lastYear=org.apache.oozie.extensions.OozieELExtensions#ph2_lastYear,
            latest=org.apache.oozie.coord.CoordELFunctions#ph2_coord_latest_echo,
            future=org.apache.oozie.coord.CoordELFunctions#ph2_coord_future_echo,
            formatTime=org.apache.oozie.coord.CoordELFunctions#ph2_coord_formatTime,
            user=org.apache.oozie.coord.CoordELFunctions#coord_user
        </value>
    </property>

    <property>
        <name>oozie.service.ELService.ext.functions.coord-job-submit-data</name>
        <value>
            now=org.apache.oozie.extensions.OozieELExtensions#ph1_now_echo,
            today=org.apache.oozie.extensions.OozieELExtensions#ph1_today_echo,
            yesterday=org.apache.oozie.extensions.OozieELExtensions#ph1_yesterday_echo,
            currentMonth=org.apache.oozie.extensions.OozieELExtensions#ph1_currentMonth_echo,
            lastMonth=org.apache.oozie.extensions.OozieELExtensions#ph1_lastMonth_echo,
            currentYear=org.apache.oozie.extensions.OozieELExtensions#ph1_currentYear_echo,
            lastYear=org.apache.oozie.extensions.OozieELExtensions#ph1_lastYear_echo,
            dataIn=org.apache.oozie.extensions.OozieELExtensions#ph1_dataIn_echo,
            instanceTime=org.apache.oozie.coord.CoordELFunctions#ph1_coord_nominalTime_echo_wrap
,
            formatTime=org.apache.oozie.coord.CoordELFunctions#ph1_coord_formatTime_echo,
            dateOffset=org.apache.oozie.coord.CoordELFunctions#ph1_coord_dateOffset_echo,
            user=org.apache.oozie.coord.CoordELFunctions#coord_user
        </value>
    </property>

    <property>
        <name>oozie.service.ELService.ext.functions.coord-action-start</name>
        <value>
            now=org.apache.oozie.extensions.OozieELExtensions#ph2_now,
            today=org.apache.oozie.extensions.OozieELExtensions#ph2_today,
            yesterday=org.apache.oozie.extensions.OozieELExtensions#ph2_yesterday,
            currentMonth=org.apache.oozie.extensions.OozieELExtensions#ph2_currentMonth,
            lastMonth=org.apache.oozie.extensions.OozieELExtensions#ph2_lastMonth,
            currentYear=org.apache.oozie.extensions.OozieELExtensions#ph2_currentYear,
            lastYear=org.apache.oozie.extensions.OozieELExtensions#ph2_lastYear,
            latest=org.apache.oozie.coord.CoordELFunctions#ph3_coord_latest,
```

```
        future=org.apache.oozie.coord.CoordELFunctions#ph3_coord_future,
        dataIn=org.apache.oozie.extensions.OozieELExtensions#ph3_dataIn,
        instanceTime=org.apache.oozie.coord.CoordELFunctions#ph3_coord_nominalTime,
        dateOffset=org.apache.oozie.coord.CoordELFunctions#ph3_coord_dateOffset,
        formatTime=org.apache.oozie.coord.CoordELFunctions#ph3_coord_formatTime,
        user=org.apache.oozie.coord.CoordELFunctions#coord_user
    </value>
</property>

<property>
    <name>oozie.service.ELService.ext.functions.coord-sla-submit</name>
    <value>
    instanceTime=org.apache.oozie.coord.CoordELFunctions#ph1_coord_nominalTime_echo_fixed,
        user=org.apache.oozie.coord.CoordELFunctions#coord_user
    </value>
</property>

<property>
    <name>oozie.service.ELService.ext.functions.coord-sla-create</name>
    <value>
        instanceTime=org.apache.oozie.coord.CoordELFunctions#ph2_coord_nominalTime,
        user=org.apache.oozie.coord.CoordELFunctions#coord_user
    </value>
</property>
```

2. Stop your Oozie server.

   ```
   su oozie
   cd /usr/lib/oozie
   bin/oozie-stop.sh
   ```

3. Confirm the Oozie Share Lib is installed on your cluster. You can install the Oozie Share Lib using the Oozie Community Quick Start docs:

   http://oozie.apache.org/docs/4.0.0/DG_QuickStart.html#Oozie_Share_Lib_Installation

4. Copy the existing Oozie WAR file to `/usr/lib/oozie/oozie.war`. This will make sure all existing items in the WAR file are still present after the current update.

   ```
   su root
   cp {$CATALINA_BASE}/webapps/oozie.war /usr/lib/oozie/oozie.war
   ```

   Where `{$CATALINA_BASE}` is the path for the Oozie web app. By default, `{$CATALINA_BASE}` is `/var/lib/oozie/oozie-server`.

5. Add the Falcon EL extensions to Oozie. Copy the extension JAR files provided with the Falcon Server to a temporary directory on the Oozie server. For example, if your standalone Falcon Server is on the same machine as your Oozie server, you can just copy the JAR files.

   ```
   mkdir /tmp/falcon-oozie-jars
   cp /usr/lib/falcon/oozie/ext/falcon-oozie-el-extension-0.4.0.2.0.6.0-76.jar \
   /tmp/falcon-oozie-jars/
   ```

6. Package the Oozie WAR file.

```
su oozie
cd /usr/lib/oozie/bin
./oozie-setup.sh prepare-war -d /tmp/falcon-oozie-jars
```

7. Start your Oozie Server.

```
su oozie
cd /usr/lib/oozie
bin/oozie-start.sh
```

Using Basic Commands

Commands are different between Standalone and Distributed Falcon and have been arranged accordingly.

- Standalone
- Distributed

Verify that you are running the right commands for your system.

Standalone

Starting the Falcon Server (standalone):

```
su falcon
/usr/lib/falcon/bin/falcon-start
```

Stopping the Falcon Server (standalone):

```
su falcon
/usr/lib/falcon/bin/falcon-stop
```

Browsing the Falcon logs (standalone):

```
/var/log/falcon/
```

Running the Falcon CLI (standalone):

```
/usr/lib/falcon/bin/falcon
```

Accessing the Falcon Server (standalone):

```
http://{your.falcon.server}:15000
```

Distributed

Starting the Falcon Prism Server (distributed):

```
su falcon
/usr/lib/falcon-distributed/bin/falcon-start
/usr/lib/falcon-distributed/bin/prism-start
```

Stopping the Falcon Prism Server (distributed):

```
su falcon
/usr/lib/falcon-distributed/bin/falcon-start
/usr/lib/falcon-distributed/bin/prism-stop
```

Browsing the Falcon logs (distributed):

```
/var/log/falcon-distributed/
```

Running the Falcon CLI (distributed):

```
/usr/lib/falcon-distributed/bin/falcon
```

Accessing the Falcon Prism Server (distributed):

```
http://{your.falcon.prism.server}:16000
```

Using the CLI

The Falcon CLI lets you perform Entity Management, Instance Management, and Admin operations from the command line. Run the CLI with the help option to see a list of commands:

```
falcon help
```

Using the REST API

The Falcon REST API lets you perform Entity Management, Instance Management, and Admin operations from a REST utility (for example, curl). When accessing the Falcon Server REST API, be sure to pass the Remote-user header. For example, the following command will retrieve the Falcon version using curl:

```
curl -H "Remote-user: root"
http://{your.falcon.server}:15000/api/admin/version
```

Note: By default, the standalone Falcon Server runs on port 15000 and in distributed mode, runs on port 16000.

You can learn more about the Falcon REST API here: http://falcon.incubator.apache.org/docs/restapi/ResourceList.html.

Configuring the Store

Once an entity definition is submitted to Falcon, by default, the XML is stored in the local file system where the Falcon Server is running. This is configured in the Falcon Server startup properties:

```
vi /etc/falcon/conf/startup.properties

######### Implementation classes #########
*.config.store.uri=file://${falcon.home}/store
```

The local file store is useful during development and testing. In production, you should change this setting to use HDFS to store entity definitions. One way to do this is to modify the property and restart your Falcon Server.

```
vi /etc/falcon/conf/startup.properties


######### Implementation classes #########
*.config.store.uri=hdfs://{namenode.hostname}:8020/apps/falcon/st
ore
```

Note: Be sure to create the `/apps/falcon` directory in HDFS and set ownership for the falcon user and group with 755 permissions. For example:

```
drwxr-xr-x falcon   hdfs  /apps/falcon
```

Removing Falcon

To completely remove Falcon from a host:

1. Connect to the host running the Falcon Server.
2. Remove the Falcon packages.

   ```
   yum erase falcon –y
   yum erase falcon-distributed -y
   ```

3. Delete the Falcon logs directory.

   ```
   rm -r /var/log/falcon
   rm -r /var/log/falcon-distributed
   ```

4. Delete the Falcon conf directory.

   ```
   rm -r /etc/falcon/conf
   rm -r /etc/falcon-distributed/conf
   ```

5. Delete the Falcon run directory.

   ```
   rm -r /var/run/falcon
   rm -r /var/run/falcon-distributed
   ```

6. Delete the Falcon lib directory.

   ```
   rm -r /var/lib/falcon
   rm -r /var/lib/falcon-distributed
   ```

Known Issues and Limitations

At the time of this release, there is the following known issue for Falcon:

HIVE-5550: Import fails for tables created with default text and sequence file formats using HCatalog API. For more information, see the Falcon documentation included in the falcon-doc package.

Note: Visit the forum for the latest discussions on Hortonworks issues:

http://hortonworks.com/community/forums/

Troubleshooting

The following troubleshooting information is available:

- Falcon Server fails to start with jar: command not found error
- Cluster entity definition submit hangs

Falcon Server fails to start with jar: command not found error

```
bash$ /usr/lib/bin/falcon-start
/usr/lib/falcon/bin/falcon-config.sh: line 82: jar: command not
found
/usr/lib/falcon/bin
Hadoop is installed, adding hadoop classpath to falcon classpath
prism started using hadoop version:  Hadoop 2.2.0.2.0.6.0-76
```

Before starting the server, be sure you have exported Java Home and put the Java binaries in the path:

```
export JAVA_HOME=/usr/jdk64/jdk1.6.0_31
export PATH=$PATH:$JAVA_HOME/bin
```

Alternatively, you can set up your environment to define JAVA_HOME in `/etc/falcon/conf/falcon-env.sh` or `/etc/falcon-distributed/conf/falcon-env.sh`.

Cluster entity definition submit hangs

Check the `/var/log/falcon` (or if running Distributed `/var/log/falcon-distributed`) application log for retry entries. This indicates Falcon was

unable to verify one or more interfaces in the cluster definition. The operation will timeout after the retries are complete.

```
2013-11-13 03:05:00,960 INFO  - [1962083476@qtp-184766585-
0:ambari-qa:POST//entities/submit/cluster 51d22bfc-d54d-4691-
95e4-364eb85c3f0e] ~ Retrying connect to server:
c6402.ambari.apache.org/192.168.64.102:8050. Already tried 47
time(s); retry policy is
RetryUpToMaximumCountWithFixedSleep(maxRetries=50, sleepTime=1
SECONDS) (Client:783)
```

Further Reading

The Falcon Apache docs are available here:

Item	URL
Falcon Project Page	http://falcon.incubator.apache.org/
Falcon Project JIRA	https://issues.apache.org/jira/browse/FALCON
Falcon CLI	http://falcon.incubator.apache.org/docs/FalconCLI.html
Falcon REST API	http://falcon.incubator.apache.org/docs/restapi/ResourceList.html
Falcon Entity Specification	http://falcon.incubator.apache.org/docs/EntitySpecification.html

About Hortonworks

Hortonworks is a leading commercial vendor of Apache Hadoop, the preeminent open source platform for storing, managing and analyzing big data. Hortonworks Data Platform provides an open and stable foundation for enterprises and a growing ecosystem to build and deploy big data solutions. Hortonworks is the trusted source for information on Hadoop, and together with the Apache community, Hortonworks is making Hadoop easier to install, manage and use. Hortonworks provides technical support, training & certification programs for enterprises, systems integrators & technology vendors.

3460 W. Bayshore Rd.
Palo Alto, CA 94303 USA

US: 1.855.846.7866
International: 1.408.916.4121
www.hortonworks.com