Oracle® Big Data Discovery

Installation and Deployment Guide

Version 1.1.3 • May 2016

ORACLE

Copyright and disclaimer

Table of Contents

Part II: Installing Big Data Discovery

Part III: After You Install

Part IV: Uninstalling Big Data Discovery

Appendix A: Optional and Internal BDD Properties

Preface

Oracle Big Data Discovery is a set of end-to-end visual analytic capabilities that leverage the power of Hadoop to transform raw data into business insight in minutes, without the need to learn complex products or rely only on highly skilled resources.

About this guide

This guide describes how to configure, install, and deploy the Oracle Big Data Discovery product. It also provides information on tasks you can perform after deployment and instructions for uninstalling the product.

This guide relates specifically to Big Data Discovery version 1.1. The most up-to-date version of this document is available on the *http://www.oracle.com/technetwork/index.html*.

Note: This guide does *not* describe how to install Big Data Discovery on the Oracle Big Data Appliance. If you want to install on the Big Data Appliance, see the *Oracle Big Data Appliance Owner's Guide Release 4 (4.3 or 4.4.)*. Additionally, see the file `BDD_README.txt` for a workaround for a BDA bug related to installation.

Audience

This guide addresses administrators and engineers who need to install and deploy Big Data Discovery within their existing Hadoop environment.

Conventions

The following conventions are used in this document.

Typographic conventions

The following table describes the typographic conventions used in this document.

Typeface	Meaning
User Interface Elements	This formatting is used for graphical user interface elements such as pages, dialog boxes, buttons, and fields.
`Code Sample`	This formatting is used for sample code segments within a paragraph.
Variable	This formatting is used for variable values. For variables within a code sample, the formatting is *`Variable`*.
`File Path`	This formatting is used for file names and paths.

Symbol conventions

The following table describes symbol conventions used in this document.

Symbol	Description	Example	Meaning
>	The right angle bracket, or greater-than sign, indicates menu item selections in a graphic user interface.	File > New > Project	From the File menu, choose New, then from the New submenu, choose Project.

Path variable conventions

This table describes the path variable conventions used in this document.

Path variable	Meaning
`$ORACLE_HOME`	Indicates the absolute path to your Oracle Middleware home directory, where BDD and WebLogic Server are installed.
`$BDD_HOME`	Indicates the absolute path to your Oracle Big Data Discovery home directory, `$ORACLE_HOME/BDD-<version>`.
`$DOMAIN_HOME`	Indicates the absolute path to your WebLogic domain home directory. For example, if your domain is named `bdd-<version>_domain`, then `$DOMAIN_HOME` is `$ORACLE_HOME/user_projects/domains/bdd-<version>_domain`.
`$DGRAPH_HOME`	Indicates the absolute path to your Dgraph home directory, `$BDD_HOME/dgraph`.

Contacting Oracle Customer Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. This includes important information regarding Oracle software, implementation questions, product and solution help, as well as overall news and updates from Oracle.

You can contact Oracle Customer Support through Oracle's Support portal, My Oracle Support at *https://support.oracle.com*.

Part I

Before You Install

Chapter 1

Introduction

The following sections describe Oracle Big Data Discovery and how it integrates with other software products. They also describe some of the different deployment configurations Big Data Discovery supports.

The Big Data Discovery software package

Oracle Big Data Discovery has a number of distinct components, which are installed and deployed simultaneously.

Studio

Studio is Big Data Discovery's front-end web application. It provides tools that enable you to create and manage data sets and projects, as well as administrator tools for managing user access and other settings. Studio stores its project data and the majority of its configuration in a relational database.

Studio is a Java-based application. It runs inside the WebLogic Server, along with the Dgraph Gateway.

Dgraph Gateway

The Dgraph Gateway is a Java-based interface that routes requests to the Dgraph instances and provides caching and business logic. It also uses Hadoop ZooKeeper to handle cluster services for the Dgraph instances.

The Dgraph Gateway runs inside WebLogic Server, along with Studio.

Data Processing

Data Processing collectively refers to a set of processes and jobs that perform discovery, sampling, profiling, and enrichment of source data. Many of the processes run within Hadoop, so Data Processing must be deployed to Hadoop nodes.

Data Processing CLI

The Data Processing Command Line Interface (CLI) provides a way to manually launch Data Processing jobs and invoke the Hive Table Detector (see below). Because the CLI shares configuration information with Studio, it is automatically deployed to all Managed Servers and Dgraph nodes. It can later be moved to any node that has access to the Big Data Discovery deployment.

Hive Table Detector

The Hive Table Detector is a Data Processing component that monitors the Hive database for new or deleted tables, and launches a Data Processing workflow when it discovers one.

The Hive Table Detector is invoked by the CLI, either manually by the Hive administrator or via the CLI cron job. If you enable the CLI to run as a cron job, the Hive Table Detector runs at each invocation of the cron job.

Dgraph

The Dgraph indexes the data sets produced by Data Processing and stores them on a shared NFS. It also responds to requests users make for records in the data sets.

The Dgraph is designed to be stateless, which allows each Dgraph instance to respond to requests independently of others. Queries are routed to the Dgraph instances by the Dgraph Gateway.

The Dgraph can be hosted on any node in the Big Data Discovery deployment, although it is recommended that you dedicate specific nodes to hosting it. The nodes that host Dgraph instances form a Dgraph cluster inside the BDD cluster.

Dgraph HDFS Agent

The Dgraph HDFS Agent acts as a data transport layer between the Dgraph and the HDFS environment. It exports records to HDFS on behalf of the Dgraph, and imports records from HDFS during data ingest operations.

The HDFS Agent is dependent on the Dgraph. It is deployed to the same nodes the Dgraph is deployed to, starts when the Dgraph starts, and shuts down when the Dgraph shuts down.

Integration with Hadoop

Hadoop provides a number of components and tools that BDD requires to process and manage data; for example, the Hadoop Distributed File System (HDFS) stores your source data and Hadoop Spark on YARN runs all Data Processing jobs.

BDD supports two Hadoop distributions:

- Cloudera Distribution for Hadoop (CDH) CDH 5.3.x , 5.4.x, 5.5.2+
- Hortonworks Data Platform (HDP) 2.2.4 - 2.3.x

Your cluster must be running one of these before you install BDD. This is because the configuration of your Hadoop cluster determines where some of the BDD components will be installed. However, Hadoop doesn't need to be installed on every node that will host BDD, as some BDD components don't require Hadoop to function. For more information, see *Hadoop requirements on page 18*.

Note: You can't connect BDD to more than one Hadoop cluster.

Integration with WebLogic

The WebLogic Server provides a J2EE container for hosting and managing Studio and the Dgraph Gateway, which are J2EE applications. Additionally, WebLogic's Admin Server plays an important role in the installation process, as well as BDD administration after deployment.

The installation package for WebLogic Server 12c (12.1.3) is included in the BDD media pack. When the BDD installer runs, it automatically installs WebLogic Server on all nodes that will host Studio and the Dgraph Gateway, and deploys both components inside of it.

Note: BDD does not currently support integration with an existing installation of WebLogic. You must use the version you download with the BDD packages.

The Admin Server serves as a central point of control for your BDD cluster. Before you install, you'll select a node to be the Admin Server and perform the entire installation from it. After installation, you can perform script-based administrative tasks—such as starting individual components and updating the cluster configuration—from this node.

You can also use the WebLogic Administration Console and WLST (WebLogic Server Scripting Tool) for starting and stopping the Managed Servers that host Studio and the Dgraph Gateway.

Deployment configurations and diagrams

BDD supports many different deployment configurations. Before installing, you can configure your deployment to have one that best supports your needs. This topic describes three types of deployments suitable for demonstration purposes, development, and production.

While this topic illustrates three types of deployments and lists their possible variations, you can deploy BDD into any configuration that meets your data processing needs; you are not limited to the configurations described in this topic.

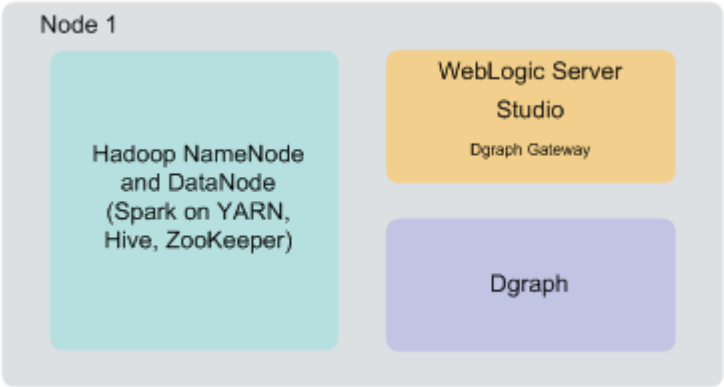
Consider the following deployment options:

- *Single-node deployment for a demo environment on page 11*
- *Two-node deployment for a development environment on page 12*
- *Six-node deployment for a production environment on page 12*

Single-node deployment for a demo environment

You can deploy BDD to a demo environment running on a single physical or virtual machine. This configuration can only handle a limited amount of data, so it is recommended solely for demonstrating the product's functionality with a small sample index.

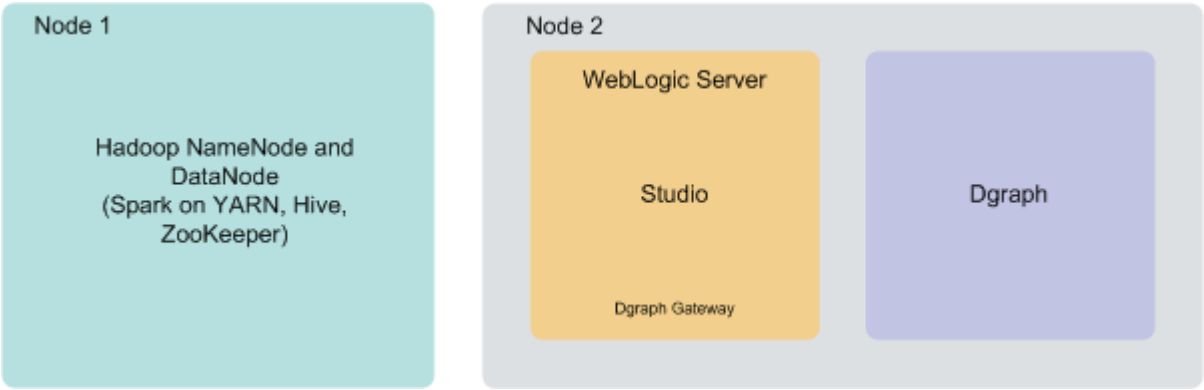
In a single-node deployment, Hadoop (including the NameNode and one DataNode), the WebLogic Server with Studio and Dgraph Gateway, and the Dgraph instance are all hosted on the same node.



Two-node deployment for a development environment

You can deploy BDD to two nodes for a development environment. This configuration can handle a slightly larger index than a single-node configuration, but is not recommended for production as it does not provide high availability of Dgraph or Studio services and also has limited capacity for processing queries on high volumes of data.

In a two-node configuration, Hadoop (including the NameNode and one DataNode) is hosted on the first node. The WebLogic Server (including Studio and the Dgraph Gateway) and the Dgraph instance are hosted on the second node.



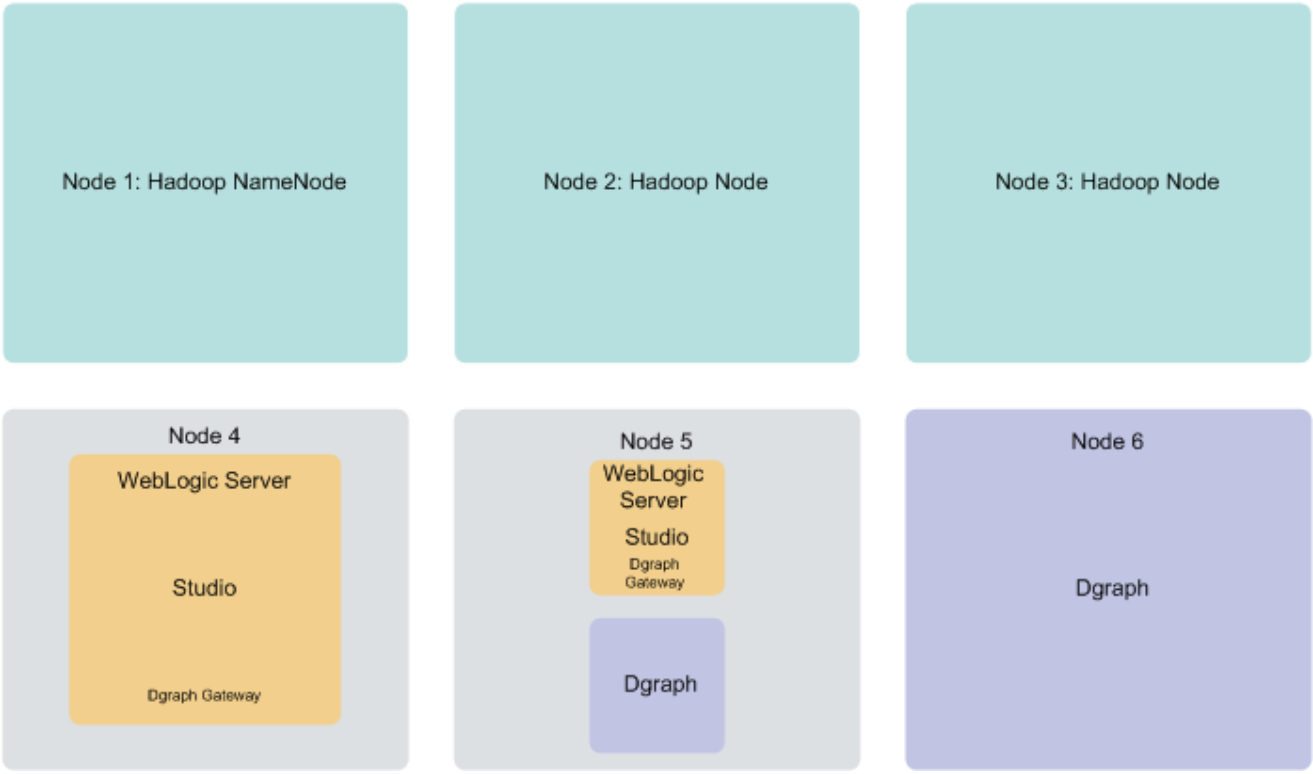
Six-node deployment for a production environment

A production environment can consist of any number of nodes required for scale; however, a cluster of six nodes, with BDD deployed on at least three Hadoop nodes, provides maximum availability guarantees.

In this six-node cluster deployment of BDD:

- Nodes 1, 2 and 3 are running Hadoop. Note that BDD is also deployed on these nodes. After the installation, Data Processing jobs are launched from these nodes and run on other BDD nodes. Having three Hadoop nodes ensures enhanced availability of BDD services, including query processing performed by the Dgraph.
- Nodes 4 and 5 are running WebLogic Server with Studio. This ensures minimal redundancy of the Studio instances.

- Nodes 5 and 6 are running the Dgraph instances. This creates a Dgraph cluster within the BDD cluster, which in turn increases the availability of query processing.



Note: You can also set up a multi-node BDD cluster in ways that differ from the suggested multi-node layout. For example, at deployment time, you can add more nodes to each category — additional Hadoop, WebLogic Server, or Dgraph nodes. You can also decide to co-locate Hadoop and WebLogic Server on some nodes, instead of dedicating separate nodes to running WebLogic Server. Similarly, you can decide to co-locate Hadoop and the Dgraph on the same node. Such decisions may have an impact on overall performance and are dependent on your site's resources and deployment requirements. See section *About co-locating Hadoop, WebLogic Server, and the Dgraph on page 14* in this topic.

About the number of nodes

This documentation does not provide sizing recommendations. To determine an appropriate size for your deployment, use the following guidelines along with your site's specific requirements.

Important: You should determine the number of WebLogic Server and Dgraph nodes your cluster will include before installing BDD, as you won't be able to add more without reinstalling. You'll be able to add Hadoop nodes after you install, though; see the *Data Processing Guide* for more information.

The following statements provide high-level guidance on the number of nodes in each category — Hadoop nodes, WebLogic Server nodes with Studio, and Dgraph nodes:

- Hadoop nodes. Your BDD deployment must include at least one Hadoop node. For high availability, Oracle recommends having at least three. (Note: Your pre-existing Hadoop cluster may have more than

three nodes. The Hadoop nodes that are discussed here are those BDD has also been deployed on.) The BDD installer will automatically install Data Processing on all qualified Hadoop nodes in the cluster.

- WebLogic Server nodes. Your deployment must include at least one WebLogic Server node running Studio and Dgraph Gateway. There is no recommended number of Studio instances, but if you expect to have a large number of end users generating concurrent query requests to BDD, it may be desirable to run two Studio instances (and thus configure two WebLogic Server nodes). If you have more than one WebLogic Server node, Oracle recommends configuring an external load balancer that is connected to the Studio instances running on these nodes. You must specify the number of WebLogic Server nodes in the installer's configuration file before installing.
- Dgraph nodes. Your deployment must include at least one Dgraph instance. If it includes more than one, the Dgraph instances will be run as a cluster within the BDD cluster. Having a cluster of Dgraphs is desirable because it enhances high availability of query processing. You must specify the number of Dgraph nodes in the installer's configuration file before installing.

About co-locating Hadoop, WebLogic Server, and the Dgraph

One way to configure your cluster is to co-locate different components on the same nodes. For example, a single node in your BDD cluster deployment can host any combination of Hadoop, the Weblogic Server, and the Dgraph, including all three components together.

Co-locating components enables you to use your hardware more efficiently, since you don't have to devote an entire server to any specific BDD component. However, it does mean that the co-located components must compete for memory, which can have a negative impact on performance.

The decision to host different components on the same nodes depends on your site's production requirements and the capacity of the machines running each component.

Possible component combinations include:

- The Dgraph and Hadoop. For best performance, Oracle recommends dedicating specific nodes to running the Dgraph (one Dgraph per machine); however, it's possible to host the Dgraph on Hadoop DataNodes. If you decide to co-locate the Dgraph and Hadoop, Oracle recommends that you use a node that isn't running Spark on YARN. Additionally, you should allocate a specific amount of memory to the Dgraph process using Linux cgroups (control groups) and Dgraph flags to prevent it from crashing. For more information on Dgraph flags, see the *Administrator's Guide*.
- The Dgraph and WebLogic Server. The Dgraph and WebLogic Server can be hosted on the same node. If you do this, you should configure the WebLogic Server to consume a limited amount of memory to ensure the Dgraph process has access to sufficient resources for its query processing.
- WebLogic Server and Hadoop. WebLogic Server and Hadoop can be co-located. If do this, you should configure the WebLogic Server to consume a limited amount of memory to ensure that Hadoop has access to sufficient resources for processing.

Security

You have the following options for securing your BDD cluster.

Kerberos and Sentry

Kerberos is a third-party tool that enables secure communication between the individual nodes in your cluster. Sentry is a Hadoop component that controls access to your data within Hive. BDD supports integration with both to ensure the security of your cluster and data.

Note: If you're deploying BDD to a production environment, Oracle strongly recommends enabling both Kerberos and Sentry.

If you want to enable Kerberos and/or Sentry for your BDD cluster, you must set them up on your Hadoop cluster before you install BDD. You must also configure BDD to integrate with them to ensure it can interact with Hadoop and access the data it requires. For more information, see *Kerberos and Sentry requirements on page 29*.

SSL

Currently, you can't configure SSL for the inward-facing ports between BDD components. Oracle therefore recommends that you deploy BDD behind a firewall. You can, however, enable SSL on Studio's outward-facing ports in one or both of the following ways:

- Enable encryption through WebLogic Server. You can do this in the BDD configuration file. This method activates WebLogic's default demo keystores, which you should replace with your own certificates after deployment. For more information, see *Replacing certificates on page 39*.
- Set up a reverse-proxy server. For instructions on how to do this, see *About reverse proxies on page 47*.

Note: These methods don't enable encryption on the inward-facing port on which the Dgraph Gateway listens for requests from Studio.

Administration

After deployment, you have two options for administering Big Data Discovery: the `bdd-admin` script and Oracle Enterprise Manager.

The bdd-admin script

The `bdd-admin` script enables you to perform a number of administrative operations from the command line, such as starting and stopping individual components and updating the configuration of the entire BDD cluster. This script is installed with the rest of the BDD components and can be run from the WebLogic Admin Server. For more information on the `bdd-admin` script, see the *Administrator's Guide*.

Enterprise Manager Plug-in

The Enterprise Manager Plug-in for Big Data Discovery extends Oracle Enterprise Manager Cloud Control to add support for monitoring, diagnosing, and managing BDD components. This is a separate product that enables you to administer your BDD cluster using a graphical user interface. It integrates with BDD but is not included in the installation.

For more information on Enterprise Manager, see the *Oracle Enterprise Manager documentation*.

A note about component names

Some of the installation files and scripts may contain references to the Endeca Server, which is a legacy name for the Dgraph Gateway. This document refers to the component as the Dgraph Gateway, and notes any discrepancies to avoid confusion.

Chapter 2

Prerequisites

The following sections describe the hardware and software requirements your environment must meet before you can install BDD.

Supported platforms

At a high level, BDD supports the following platforms.

Note that this is not an exhaustive list of BDD's requirements. Be sure to read through the rest of this chapter before installing for more information about the components and configuration changes BDD requires.

Component	Supported versions
Operating system	• Oracle Enterprise Linux 6.4+ • Red Hat Enterprise Linux 6.4+

Component	Supported versions
Hadoop distribution	• Cloudera Distribution for Hadoop (CDH) 5.3.x, 5.4.x, 5.5.2+ • Hortonworks Data Platform (HDP) 2.2.4-2.3.x
Application server	• WebLogic Server 12c 12.1.3
JDK	• HotSpot JDK 7u67+ x64 • HotSpot JDK 8u45+ x64
Studio database server	• Oracle 11g • Oracle 12c 12.1.0.1.0+ • MySQL 5.5.3+ • HSQLDB (Hypersonic), for development/non-production environments only
Browser	• Internet Explorer 10 and 11 • Firefox ESR • Chrome for Business • Safari Mobile 8.x

Supported operating systems

BDD supports the following operating systems:

- Oracle Enterprise Linux 6.4+ x86_64
- Red Hat Enterprise Linux 6.4+ x86_64

One of these must be installed on all nodes in the cluster, including Hadoop nodes.

Hadoop requirements

You must install one of the following Hadoop distributions on your cluster before you install BDD:

- Cloudera Distribution for Hadoop (CDH) 5.3.x, 5.4.x, 5.5.2+
- Hortonworks Data Platform (HDP) 2.2.4 - 2.3.x

Note: You can switch to a different version of your Hadoop distribution after you install, if necessary. See the *Administrator's Guide* for more information.

BDD doesn't require all of the components each distribution provides, and the components it does require don't need to be installed on all nodes. The following table lists the required Hadoop components and the node(s) they must be installed on.

Note: If you are installing on a single machine, that machine must have all required Hadoop components installed.

Component	Description
Cloudera Manager (CDH)/Ambari (HDP)	The BDD installer uses a RESTful API to query Cloudera Manager (if you're using CDH) or Ambari (If you're using HDP) for information about specific Hadoop nodes, such as their hostnames and port numbers. Cloudera Manager/Ambari must be installed on at least one node in your cluster, although it doesn't have to be on any that will host BDD.
ZooKeeper	BDD uses ZooKeeper services to manage the Dgraph instances and ensure high availability of Dgraph query processing. ZooKeeper must be installed on at least one node in your cluster, although it doesn't have to be on any that will host BDD. For more information on ZooKeeper and how it affects the cluster deployment's high availability, see the *Administrator's Guide*. All Managed Servers must be able to connect to a node running ZooKeeper.
HDFS	BDD stores the Hive tables that contain your source data in HDFS. HDFS must be installed on at least one node in your cluster, although it doesn't need to be on any that will host BDD. HDFS must be installed on all nodes that will run Data Processing.
HCatalog	The Data Processing Hive Table Detector monitors HCatalog for new and deleted tables that require processing. HCatalog must be installed on at least one node in your cluster, although it doesn't have to be one that will host BDD.
Hive	All of your data is stored as Hive tables on HDFS. When BDD discovers a new or modified Hive table, it launches a Data Processing workflow for that table.
Spark on YARN	BDD uses Spark on YARN to run all Data Processing jobs. Spark on YARN must be installed on all nodes that will run Data Processing.
Hue	You can use Hue to load your source data into Hive and to view data exported from Studio. **Note:** HDP doesn't include Hue. If you have an HDP cluster, you must install it separately and set the `HUE_URI` property in BDD's configuration file. You can also use the `bdd-admin` script to update this property after installation, if necessary. For more information, see the *Administrator's Guide*.
YARN	YARN worker nodes run all Data Processing jobs. YARN must be installed on all nodes that will run Data Processing.

Note: Data Processing will automatically be installed on nodes running the following Hadoop components:

- Spark on YARN

- YARN
- HDFS

Additionally, if you will be co-locating the Dgraph and Hadoop, you must enable cgroups on that node and limit the Dgraph's memory consumption.

You must also make a few changes within your Hadoop cluster to ensure that BDD can communicate with your Hadoop nodes. These changes are described below.

YARN setting changes

Required Hadoop client libraries

HDP-specific requirements

YARN setting changes

To ensure that each YARN worker node has access to sufficient resources during processing, you need to update the following YARN-specific Hadoop properties.

You can access these properties in Cloudera Manager/Ambari. If you need help finding them, refer to the documentation for your Hadoop distribution.

Property	Description
`yarn.nodemanager.resource.memory-mb`	The total amount of memory available to your entire YARN cluster. This should be at least 16GB, although you might need to set it higher depending on the amount of data you plan on processing.
`yarn.scheduler.maximum-allocation-vcores`	The maximum number of virtual CPU cores allocated to each YARN container per request. If your cluster contains only one YARN worker node, this should be less than or equal to half of that node's cores. If your cluster contains multiple YARN worker nodes, this should be less than or equal to each node's total number of cores.
`yarn.scheduler.maximum-allocation-mb`	The maximum amount of RAM allocated to each YARN container per request. If your cluster contains only one YARN worker node, this should be less than or equal to half of that node's RAM. If your cluster contains multiple YARN worker nodes, this should be less than or equal to each node's total amount of RAM.
`yarn.scheduler.capacity.maximum-applications`	The maximum number of concurrently-running jobs allowed on each node. This can be between 2 and 8. Note that setting this value higher could cause jobs submitted at the same time to hang indefinitely.

Required Hadoop client libraries

BDD requires a number of client libraries to interact with Hadoop. When the installer runs, it adds these libraries to a single jar, called the Hadoop fat jar, which it distributes to all BDD nodes.

How you obtain the client libraries depends on your Hadoop distribution. If you have CDH, the installer will download them automatically. Note that this requires an internet connection on the install machine. If the script can't download all of the client libraries, it will fail and you will have to download them manually. See *Failure to download the Hadoop client libraries on page ???* for more information.

If you have HDP, you must manually copy the client libraries from your Hadoop nodes to the install machine. The specific libraries you need depend on the version of HDP you have.

HDP 2.2.4

If you have HDP 2.2.4, locate the following directories on your Hadoop nodes and copy them to the install machine:

Note: These directories might not all be on the same node.

- `/usr/hdp/<version>/pig/lib/h2/`
- `/usr/hdp/<version>/hive/lib/`
- `/usr/hdp/<version>/spark/lib/`
- `/usr/hdp/<version>/spark/external/spark-native-yarn/lib/`
- `/usr/hdp/<version>/hadoop/`
- `/usr/hdp/<version>/hadoop/lib/`
- `/usr/hdp/<version>/hadoop-hdfs/`
- `/usr/hdp/<version>/hadoop-hdfs/lib/`
- `/usr/hdp/<version>/hadoop-yarn/`
- `/usr/hdp/<version>/hadoop-yarn/lib/`
- `/usr/hdp/<version>/hadoop-mapreduce/`
- `/usr/hdp/<version>/hadoop-mapreduce/lib/`

HDP 2.3.x

If you have HDP 2.3.x, locate the following directories on your Hadoop nodes and copy them to the install machine:

Note: These directories might not all be on the same node.

- `/usr/hdp/<version>/hive/lib/`
- `/usr/hdp/<version>/spark/lib/`
- `/usr/hdp/<version>/hadoop/`
- `/usr/hdp/<version>/hadoop/lib/`

- `/usr/hdp/<version>/hadoop-hdfs/`
- `/usr/hdp/<version>/hadoop-hdfs/lib/`
- `/usr/hdp/<version>/hadoop-yarn/`
- `/usr/hdp/<version>/hadoop-yarn/lib/`
- `/usr/hdp/<version>/hadoop-mapreduce/`
- `/usr/hdp/<version>/hadoop-mapreduce/lib/`

HDP-specific requirements

If you have HDP, there are a few additional things you need to do to enable BDD to work with your Hadoop cluster.

Required JARs

You also need to make sure that a few JAR files are present on all of your Hadoop nodes. The specific files depend on the version of HDP you have.

Note: This isn't required if you have CDH.

If you have HDP 2.2.4, verify that the following are present on all of your Hadoop nodes:

- `/usr/hdp/2.2.4.X/hive/lib/hive-exec.jar`
- `/usr/hdp/2.2.4.X/spark/lib/spark-assembly-1.2.1.2.2.4.X-hadoop2.6.0.2.2.4.X.jar`

If you have HDP 2.3.x, verify that the following are present on all of your Hadoop nodes:

- `/usr/hdp/2.3.0.0-X/hive/lib/hive-metastore.jar`
- `/usr/hdp/2.3.0.0-X/spark/lib/spark-assembly-1.2.1.2.3.X-hadoop2.6.0.2.3.X.jar`

If any are missing, you can copy them over from one of your Hive or Spark nodes.

Hardware requirements

The hardware requirements for your specific BDD deployment depend on the amount of data you will process. Oracle recommends the following minimum requirements:

Note: In this guide, the term "x64" refers to any processor compatible with the AMD64/EM64T architecture. You might need to upgrade your hardware, depending on the data you are processing. All run-time code must fit entirely in RAM. Likewise, hard disk capacity must be sufficient based on the size of your data set. Please contact your Oracle representative if you need more information on sizing your hardware.

- x86_64 2-core CPU for nodes that will run the Dgraph and HDFS Agent
- x86_64 4-core CPU cores for WebLogic Managed Servers, which will run Studio and the Dgraph Gateway

Note: Oracle recommends turning off hyper-threading for Dgraph nodes. Because of the way the Dgraph works, it is actually detrimental to cache performance to use hyper-threading.

Network requirements

Any host machine in a BDD cluster must have a hostname that is externally-resolvable and accessible using the network (IP) address resolved for that hostname. Oracle recommends that host names be fully qualified.

Physical memory and disk space requirements

The physical memory and disk space requirements for each node depend on its type.

Type of node	Requirements
Install machine/WebLogic Admin Server	The install machine, which will become the Admin Server after you install, has the following requirements: • At least 512MB of free swap space. If you need to install WebLogic Server on nodes that don't meet this requirement, be sure to set the `WLS_NO_SWAP` property in BDD's configuration file to `TRUE`. • 10GB of available space in the directory defined by the `TEMP_FOLDER_PATH` property in BDD's configuration file. • 6GB of available space in the directory defined by the `ORACLE_HOME` property in BDD's configuration file. • At least 10GB of available space in the directory defined by the `INSTALLER_PATH` property in BDD's configuration file.
WebLogic Managed Server	All Managed Servers have the following requirements: • At least 5GB of RAM. Your system might require more, depending on the amount of data plan on processing. • At least 512MB of free swap space. If you need to install WebLogic Server on nodes that don't meet this requirement, be sure to set the `WLS_NO_SWAP` property in BDD's configuration file to `TRUE`. • 10GB of available space in the directory defined by the `TEMP_FOLDER_PATH` property in BDD's configuration file. • 6GB of available space in the directory defined by the `ORACLE_HOME` property in BDD's configuration file.
Dgraph	Dgraph nodes have the following requirements: • At least 5GB of RAM. Your system might require more, depending on the amount of data you plan on processing. • 10GB of available space in the directory defined by the `TEMP_FOLDER_PATH` property in BDD's configuration file. • 1GB of available space in the directory defined by the `ORACLE_HOME` property in BDD's configuration file.

Type of node	Requirements
YARN worker nodes	YARN worker nodes, which will run Data Processing jobs, have the following requirements: • At least 16GB of RAM for the entire YARN cluster combined (not per node). Your system might require more, depending on the amount of data you plan on processing. • 3GB of available space in the directory defined by the `TEMP_FOLDER_PATH` property in BDD's configuration file. • 2GB of available space in the directory defined by the `ORACLE_HOME` property in BDD's configuration file. You must also update some YARN-specific properties in your Hadoop cluster to ensure that each YARN worker node has access to sufficient resources. These are described in *YARN setting changes on page 20*.

Screen resolution requirements

BDD has the following screen resolution requirements:

- Minimum: 1366x768
- Recommended: 1920x1080

User access requirements

The Linux user that performs the installation must be the one that will run all BDD processes after the installation and can't be the root user. The following must be configured for this user:

- Password-less sudo-to-root enabled on all nodes in the cluster, including Hadoop nodes.
- Bash set as the default shell on all nodes in the cluster, including Hadoop nodes.
- Permission to create the directory in which BDD and the WebLogic Server will be installed on all nodes in the cluster, including Hadoop nodes. (This directory is defined by the `ORACLE_HOME` property in the BDD configuration file.)
- Passwordless SSH enabled so they can log into all other nodes in the cluster (including Hadoop nodes) from the install machine (instructions are available in the following topic).

Enabling passwordless SSH

Enabling passwordless SSH

You must enable passwordless SSH on all nodes in the cluster for the user who will perform the installation.

To enable passwordless SSH:

1. Generate SSH keys on all nodes in the cluster, including Hadoop nodes.

2. Copy the keys to the install machine to create `known_hosts` and `authorized_keys` files.
3. Copy the `known_hosts` and `authorized_keys` files to all servers in the cluster.

JDK requirements

BDD requires one of the following JDK versions:

Note: BDD requires a JDK that includes the HotSpot JVM. This will be included in any version you download using the following links, as long as you *don't* select a version from the JRockit Family.

- *JDK 7u67+ x64*
- *JDK 8u45+ x64*

The JDK must be installed in the same location on all nodes.

Note: If one of the supported JDKs is installed on your Hadoop nodes, you can copy it to your BDD nodes.

Additionally, you must set the `$JAVA_HOME` environment variable on all nodes. If you have multiple versions of the JDK installed, be sure that this points to the correct one. Additionally, if the path is set to or contains a symlink, the symlink must be identical on all other nodes.

Required Linux utilities

The BDD installer requires several Linux utilities.

The following must be present in the `/bin` directory:

```
basename
cat
chgrp
chown
date
dd
df
mkdir
more
rm
sed
tar
true
```

The following must be present in the `/usr/bin` directory:

```
awk
cksum
cut
dirname
expr
gzip
head
id
netcat
perl (see below)
printf
sudo (Note: This is the default version on OEL 6.)
```

```
tail
tr
unzip
wc
which
```

BDD requires Perl 5.10+ with multithreading. This must be set as the default version on all BDD nodes. Additionally, the install machine requires a few specific Perl modules; see the following section for instructions on installing these.

Finally, `tty` must be disabled for sudo. If it's currently enabled, comment out the line `Defaults requiretty` in `/etc/sudoers` on all nodes:

```
#Defaults requiretty
```

Installing the required Perl modules

Installing the required Perl modules

You must install the following Perl modules on the install machine:

- `Mail::Address`
- `XML::Parser`
- `JSON-2.90`

Note: You only need to perform this procedure on the install machine. These modules aren't required on any other nodes.

To install the required Perl modules:

1. Install `Mail::Address`:
   (a) Download `Mail::Address` from *http://pkgs.fedoraproject.org/repo/pkgs/perl-MailTools/MailTools-2.14.tar.gz/813ae849683367bb75e6be89e4e8cc46/MailTools-2.14.tar.gz*.
   (b) Extract `MailTools-2.14.tar.gz`:
   ```
   tar -xvf MailTools-2.14.tar.gz
   ```
   This creates a directory called `/MailTools-2.14`.
   (c) Go to `/MailTools-2.14` and run the following commands to install the module:
   ```
   perl Makefile.PL
   make
   make test
   sudo make install
   ```
2. Install `XML::Parser`:
   (a) Download `XML::Parser` from *http://search.cpan.org/CPAN/authors/id/T/TO/TODDR/XML-Parser-2.44.tar.gz*.
   (b) Extract `XML-Parser-2.44.tar.gz`:
   ```
   tar -xvf XML-Parser-2.44.tar.gz
   ```
   This creates a directory called `/XML-Parser-2.44`.

(c) Go to `/XML-Parser-2.44` and run the following commands to install the module:

```
perl Makefile.PL
make
make test
sudo make install
```

3. Install `JSON-2.90`:

   (a) Download `JSON-2.90` from *http://search.cpan.org/CPAN/authors/id/M/MA/MAKAMAKA/JSON-2.90.tar.gz*.

   (b) Extract `JSON-2.90.tar.gz`:

   ```
   tar -xvf JSON-2.90.tar.gz
   ```

   This creates a directory called `/JSON-2.90`.

   (c) Go to `/JSON-2.90` and run the following commands to install the module:

   ```
   perl Makefile.PL
   make
   make test
   sudo make install
   ```

Database requirements

Studio requires a relational database to store configuration and state, including component configuration, user permissions, and system settings.

BDD supports the following database types:

- Oracle 11g
- Oracle 12c 12.1.0.1.0+
- MySQL 5.5.3+

Note: BDD does not currently support database migration. If you decide to switch to a different type of database later on, you must reinstall BDD with a new database instance.

If you're installing BDD in a production environment, you must create the following:

- A database of one of the types listed above.
- A database username and password.
- An empty schema. The name of this is arbitrary, but the default is `studio`.

Note: All Studio instances must be able to connect and write to the same database. To avoid performance issues, Oracle recommends that you connect them over the same local network.

Sample commands for creating Oracle and MySQL database users and schemas are available in *Sample commands for a production database on page 28*.

Additionally, you must install the database client on the install machine. For MySQL, this should be MySQL client. For Oracle databases, this should be Oracle Database Client, installed with a type of Administrator. Note that the Instant Client is not supported.

If you have an Oracle database, you must set the `ORACLE_HOME` environment variable to the directory one level above the `/bin` directory that the `sqlplus` executable is located in. For example, if the `sqlplus` executable is located in `/u01/app/oracle/product/11/2/0/dbhome/bin`, you should set `ORACLE_HOME` to `/u01/app/oracle/product/11/2/0/dbhome`. Note that this is different from the `ORACLE_HOME` property in BDD's configuration file.

If you have a MySQL database, you must set UTF-8 as the default character set.

Demo environment database requirements

If you are installing BDD in a demo environment, you can use one of the databases listed above or a Hypersonic (HSQL) database.

Hypersonic is an embedded database running inside the JVM. It is useful for getting Studio up and running quickly, but can't be used in a production environment due to performance issues and its inability to support multiple Studio nodes.

Important: If you install in a demo environment with a Hypersonic database and later decide to scale up to a production environment, you must reinstall BDD with one of the supported MySQL or Oracle databases listed above.

Sample commands for a production database

Sample commands for a production database

Below are sample commands you can use to create users and schemas for Oracle and MySQL databases. You are not required to use these exact commands when setting up your database—these are just examples to help get you started.

Oracle database

You can use the following commands to create a user and schema for an Oracle 11g or 12c database.

```
CREATE USER <username> PROFILE "DEFAULT" IDENTIFIED BY <password> DEFAULT TABLESPACE "USERS"
TEMPORARY TABLESPACE "TEMP" ACCOUNT UNLOCK;
GRANT CREATE PROCEDURE TO <username>;
GRANT CREATE SESSION TO <username>;
GRANT CREATE SYNONYM TO <username>;
GRANT CREATE TABLE TO <username>;
GRANT CREATE VIEW TO <username>;
GRANT UNLIMITED TABLESPACE TO <username>;
GRANT CONNECT TO <username>;
GRANT RESOURCE TO <username>;
```

MySQL database

You can use the following commands to create a user and schema for a MySQL database.

Note: MySQL databases must use UTF-8 as the default character encoding.

```
create user '<username>'@'%' identified by '<password>';
create database <database name> default character set utf8 default collate utf8_general_ci;
grant all on <database name>.* to '<username>'@'%' identified by '<password>' with grant option;
```

```
flush privileges;
```

Index requirements

The Dgraph requires an index to store the contents of the data sets it can query. It's stored on a shared NFS, which all Dgraph nodes must have read/write access to. Note that at any given time, only one Dgraph instance, the leader, has write access.

You can install BDD with an existing BDD-formatted index if you have one you'd like to use. To do this, you must put it on the NFS before installing and update BDD's configuration file to point to it. For more information, see *Configuring BDD on page ???*.

If you don't have an existing index, the installer can create an empty one for you. This is also configured in BDD's configuration file.

Kerberos and Sentry requirements

BDD supports integration with Kerberos 5+ and the version of Sentry included with your Hadoop distribution. If either or both of these are enabled for your Hadoop cluster, you must also enable them for BDD to ensure it can access the data it requires.

For more information on Kerberos and Sentry, see *Security on page 15*.

This procedure assumes you already have Kerberos and/or Sentry installed.

To enable Kerberos and Sentry:

1. Create the following directories in HDFS:
   - `/user/<BDD user>`, where `<BDD user>` is the name of the Linux user that will install BDD (the BDD user).
   - `/user/<HDFS_DP_USER_DIR>`, where `<HDFS_DP_USER_DIR>` is the value of `HDFS_DP_USER_DIR` in BDD's configuration file.

   The owner of both directories must be the BDD user and their group must be `supergroup`.
2. Add the BDD user to the `hive` group.
3. For Kerberos:
   (a) Create a BDD principal.

   The primary component must be the name of the BDD user and the realm must be your default realm.

   (b) Generate a keytab file for the BDD principal and copy it to the install machine.

   The name and location of this file are arbitrary as the installer will rename it `bdd.keytab` and copy it to all BDD nodes.

   (c) Copy the `krb5.conf` file from one of your Hadoop nodes to the install machine.

   The location you put it in is arbitrary as the installer will copy it to `/etc` on all BDD nodes.

   (d) Install the `kinit` and `kdestroy` utilities on all BDD nodes.

   (e) Add the BDD user to the `hdfs` group on all BDD nodes.

(f) Update the Kerberos-related properties in BDD's configuration file.

(g) If you use HDP, set the `hadoop.proxyuser.hive.groups` property in `core-site.xml` to `*`.

You can do this in Ambari.

4. For Sentry, create a new role for BDD:

```
create role <BDD_role>;
grant all on server server1 to role <BDD_role>;
show grant role <BDD_role>;
grant role <BDD_role> to group hive;
```

After installation, you can use the `bdd-admin` script to update your Kerberos configuration, if necessary. For more information, see the *Administrator's Guide*.

Supported Web browsers

Studio supports the following Web browsers:

- Firefox ESR
- Internet Explorer 10 and 11 (compatibility mode is not supported)
- Chrome for Business
- Safari 8+ (for mobile)

Studio support for iPad

You can use the Safari Web browser on an iPad running iOS 7 or later to sign in to Studio and view projects. You cannot use an iPad to create, configure, or export projects.

While the iPad can support most component functions, the component export option is disabled.

Part II

Installing Big Data Discovery

Part III

After You Install

Chapter 3

Post-Installation Tasks

The following sections describe tasks you can perform after you install BDD, such as verifying your installation and increasing Linux file descriptors.

Navigating the BDD directory structure

Your BDD installation consists of two main directories: `$BDD_HOME` and `$DOMAIN_HOME`.

$BDD_HOME

`$BDD_HOME` is the root directory of your BDD installation. Its default path is:

```
$ORACLE_HOME/BDD-<version>
```

`$BDD_HOME` contains the following subdirectories.

Directory name	Description
`/BDD_manager`	Directories related to the `bdd-admin` script: • `/bin`: Contains `bdd-admin` and other relevant scripts. You can use `bdd-admin` to administer your cluster from the command line. • `/conf`: Contains `bdd.conf`. • `/linux`: Additional scripts required by the `bdd-admin` script. • `/log`: Contains the `bdd-admin` log files. • `version.txt`: Contains version information for `bdd-admin`. More information on the `bdd-admin` script is available in the *Oracle Big Data Discovery Administrator's Guide*. **Note:** Because this directory is required for updating the cluster configuration after deployment and uninstalling BDD, it is created on all Managed Servers, Dgraph nodes, and YARN NodeManager nodes in the cluster. However, the `bdd-admin` script can only be run from the Admin Server.
`/common/hadoop`	Files and directories BDD requires to communicate with Hadoop: • `/conf`: The configuration files for your Hadoop cluster. • `/lib`: The Hadoop fat jar generated from the Hadoop client libraries.
`/dataprocessing`	Executables and packages for Data Processing, as well as the following subdirectories: • `/edp/log`: Data Processing log files. • `/edp_cli`: The CLI and related files. This directory is only available on the Managed Servers.

Directory name	Description
`/dgraph`	Files and directories related to the Dgraph, including: • `/bin`: Scripts for administering the Dgraph. • `/bin/trace_logs`: The Dgraph Tracing Utility logs. • `/conf`: Stylesheets for Dgraph statistics pages and schemas for Dgraph queries and responses. • `/dgraph-hdfs-agent`: Scripts for administering the HDFS Agent and its libraries. • `/doc` • `/empty_indexes`: The empty indexes the installer uses to create the `base` index. This directory is only used if you set `DGRAPH_INDEX_NAME` to `base`. • `/lib` and `/lib64`: Dgraph libraries. • `/msg`: Localized messages for EQL queries. • `/nls` and `/olt`: Files related to the OLT. • `/ssl`: File for configuring SSL. • `version.txt`: Contains version information for the Dgraph and HDFS Agent components. • `/xquery`: XQuery documents for communications between the Dgraph and other services.
`/logs`	Log files for the Dgraph Dgraph HDFS Agent, and Data Processing components.
`/server`	Files and directories related to the Dgraph Gateway, including: • `/endeca-cmd`: Contains the `endeca-cmd` command line utility, which you use to administer the Dgraph Gateway from the command line. • `/endeca-server`: EAR file for the Dgraph Gateway application. • `README_BDD.txt`: The BDD release notes. • `version.txt`: Contains version information for the Dgraph Gateway component.
`/studio`	Contains the EAR file for the Studio application, and a version file for Studio.
`/uninstall`	The uninstall script and its required utilities.
`version.txt`	Contains version information for your BDD installation.

$DOMAIN_HOME

`$DOMAIN_HOME` is the root directory of Studio, the Dgraph Gateway, and your WebLogic domain. Its default path is:

```
$ORACLE_HOME/user_projects/domains/bdd-<version>_domain
```

`$DOMAIN_HOME` contains the following subdirectories.

Directory name	Description
`/autodeploy`	Provides a way to quickly deploy applications to a development server. You can place J2EE applications in this directory; these will be automatically deployed to the WebLogic Server when it is started in development mode.
`/bin`	Scripts for migrating servers and services, setting up domain and startup environments, and starting and stopping the WebLogic Server and other components.
`/config`	Data sources and configuration files for Studio and the Dgraph Gateway.
`/console-ext`	Console extensions. This directory is only used on the Admin Server.
`edit.lok`	Ensures can only edit the domain's configuration one at a time. Don't edit this file.
`/EndecaServer`	Contains files and libraries used by the Dgraph Gateway.
`fileRealm.properties`	Configuration file for the file realm.
`/init-info`	Schemas used by the Dgraph Gateway.
`/lib`	The domain library. The JAR files in this directory are dynamically added to the end of the Dgraph Gateway's classpath when the Dgraph Gateway is started. You use this directory to add application libraries to the Dgraph Gateway's classpath.
`/nodemanager`	Files used by the Node Manager. `nodemanager.domains` lists the locations of directories created by the configuration wizard, and `nodemanager.properties` configures the Node Manager.
`/pending`	Stores pending configuration changes.
`/security`	Files related to domain security.
`/servers`	Log files and security information for each server in the cluster.
`startWebLogic.sh`	Script for starting the WebLogic Server.
`/tmp`	Temporary directory.

Verifying your installation

Once the installer completes, you can verify that each of the major BDD components were installed properly and are running.

Verifying your cluster's health

Use the `bdd-admin` script to verify the overall health of your cluster.

More information on the `bdd-admin` script is available in the *Administrator's Guide*.

To verify the deployed components:

1. On the Admin Server, open a new terminal window and navigate to the `$BDD_HOME/BDD_manager/bin` directory.
2. Run the following:

```
./bdd-admin.sh status --health-check
```

If your cluster is healthy, the script's output should be similar to the following:

```
[2015/06/19 04:18:55 -0700] [Admin Server] Checking health of BDD cluster...
[2015/06/19 04:20:39 -0700] [web009.us.example.com] Check BDD functionality......Pass!
[2015/06
/19 04:20:39 -0700] [web009.us.example.com] Check Hive Data Detector health......Hive Data Detector
has previously run
[2015/06/19 04:20:39 -0700] [Admin Server] Successfully checked statuses.
```

Verifying Data Processing

To verify that Data Processing is running, you must launch a Data Processing workflow. You can do this in two ways:

- Use the CLI to launch a Data Processing workflow. For more information, please refer to the *Data Processing Guide*.
- Create a data set in Studio. For more information, please refer to the *Data Exploration and Analysis Guide*.

Note: If you use the CLI to verify Data Processing, you must first add the table(s) you want processed to the CLI whitelist. For more information, see *Updating the CLI whitelist and blacklist on page 37*.

Updating the CLI whitelist and blacklist

In order to create data sets from existing Hive tables, you must update the CLI white- and blacklists that define which tables are processed by Data Processing.

The CLI whitelist specifies which Hive tables should be processed. Tables not included in this list are ignored by the Hive Table Detector and any Data Processing workflows invoked by the CLI. Similarly, the blacklist

specifies the Hive tables that should not be processed. You can use one or both of these lists to control which of your Hive tables are processed and which are not.

Once you have updated the whitelist and/or blacklist as needed, you can either wait for the Hive Table Detector to process your tables automatically or use the CLI to start a Data Processing workflow immediately.

For information on the CLI white- and blacklists, see the *Data Processing Guide*.

Signing in to Studio as an administrator

After you complete the BDD installation and deployment, you can sign in to Studio as an administrator, begin to create new users, explore data sets, re-configure Studio settings as necessary, and so on.

To sign in to Studio as an administrator:

1. Ensure the WebLogic Server on the Admin Server node is running.

   (This is the WebLogic instance running Studio.)

2. Open a Web browser and load Studio.

   By default, the URL is `http://<Admin Server Name>:7003/bdd`.

3. Specify the admin username and password set during the installation and click **Sign In**.

   If the admin username and password weren't set, login with the default values.

   Table 3.1: Sign in Values

Field	Value
Login	`admin@oracle.com`
Password	`Welcome123`

4. Reset the password, if prompted.

   The new password must contain:

   - At least 6 characters
   - At least one non-alphabetic character

Now you can add additional Studio users. There are several ways to add new Studio Users:

- Integrate Studio with an Oracle Single Sign On (SSO) system. For details, see the *Administrator's Guide*.
- Integrate Studio with an LDAP system. For details, see the *Administrator's Guide*.
- Or, while you are signed in as an administrator, you can create users manually in Studio from the **Control Panel>Users** page.

Backing up BDD

Oracle recommends that you back up your BDD cluster immediately after deployment.

You can do this with the `bdd-admin` script. For more information, see the *Administrator's Guide*.

Replacing certificates

Enabling SSL for Studio activates WebLogic Server's default Demo Identity and Demo Trust Keystores. As their names suggest, these keystores are untrusted and meant for demo purposes only. After deployment, you should replace them with your own certificates.

More information on WebLogic's demo keystores is available in section *Configure keystores* of WebLogic's *Administration Console Online Help*.

Increasing Linux file descriptors

You should increase the number of file descriptors from the 1024 default.

Having a higher number of file descriptors ensures that the WebLogic Server can open sockets under high load and not abort requests coming in from clients.

To increase the number of file descriptors on Linux:

1. Edit the `/etc/security/limits.conf` file.
2. Modify the **nofile** limit so that **soft** is 4096 and **hard** is 8192. Either edit existing lines or add these two lines to the file:

   ```
   *      soft      nofile      4096
   *      hard      nofile      8192
   ```

   The "*" character is a wildcard that identifies all users.

Customizing the WebLogic JVM heap size

You can change the default JVM heap size to fit the needs of your deployment.

The default JVM heap size for WebLogic is 3GB. The size is set in the `setDomainEnv.sh` file, which is in the `$DOMAIN_HOME/bin` directory. The heap size is set with the `-Xmx` option.

To change the WebLogic JVM heap size:

1. Open the `setDomainEnv` file in a text editor.
2. Search for this comment line:

   ```
   # IF USER_MEM_ARGS the environment variable is set, use it to override ALL MEM_ARGS values
   ```

3. Add the following line immediately after the comment line:

   ```
   export USER_MEM_ARGS="-Xms128m -Xmx3072m ${MEM_DEV_ARGS} ${MEM_MAX_PERM_SIZE}"
   ```

4. Save and close the file.

5. Re-start WebLogic Server.

Chapter 4

Creating Multiple Studio Instances

For a larger production environment, you may want to configure a number of Studio instances.

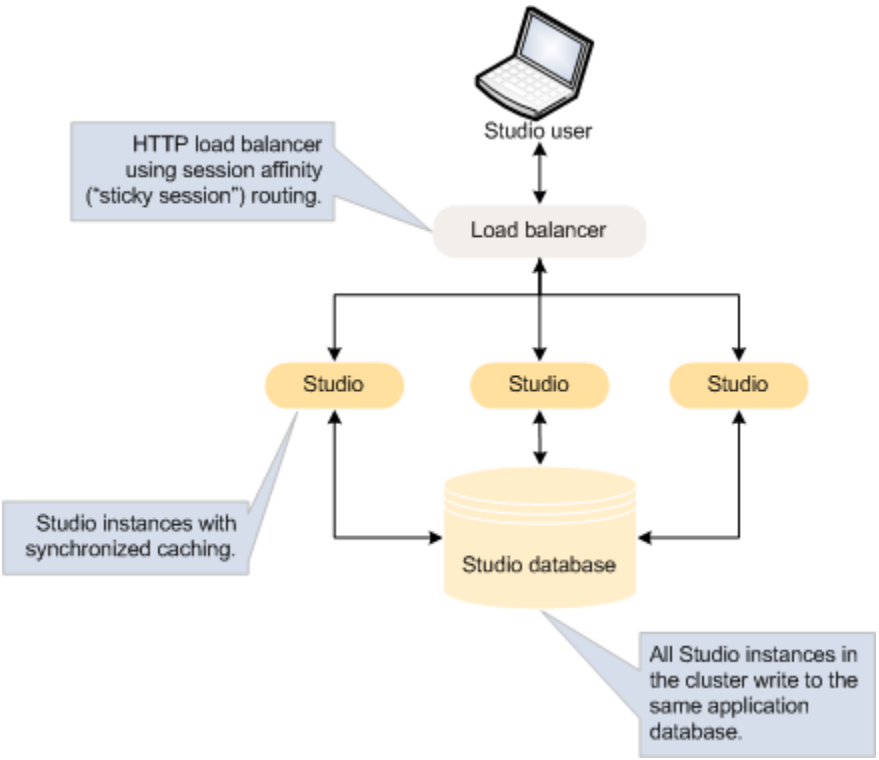
About multiple Studio instances

Studio allows you to create multiple Studio instances. In a cluster of Studio instances, changes made to one instance are automatically made to the other instances. For a large production environment, using clustering provides redundancy and support for higher throughput, allowing for more concurrent users.

A Studio cluster is made up of Studio instances configured to write to the same application database. For a clustered implementation, you cannot use a Hypersonic database.

The Studio instances also must be configured to use synchronized caching, so that information cached on one instance is available to all of the other instances in the cluster. Studio uses Ehcache (*www.ehcache.org*), which uses RMI (Remote Method Invocation) multicast to notify each member of the cluster when the cache has been updated.

In a Studio cluster, requests are routed to the Studio instances by an HTTP load balancer. The load balancer must use session affinity (also known as "sticky session") load balancing. If a member of the Studio cluster is down, the load balancer routes requests to another instance in the cluster.



Setting up multiple Studio instances

To configure multiple Studio instances, you connect each instance to the same application database, and then configure a shared cache for those instances.

Installing the Studio instances

Configuring synchronized caching for the Studio instances

Installing the Studio instances

Each instance in the cluster of Studio nodes is first installed as a standalone instance and then modified to share certain configuration.

Connecting each instance to the same Studio database

Each instance in the Studio cluster must be connected to the same Studio application database.

Optionally, you could use a clustered database configuration. For clustering, Oracle 11g uses RAC and MySQL has MySQL Cluster. For details on setting up a clustered database configuration, see the documentation for your database system.

Using the same configuration for each instance

In a clustered configuration, each instance should have the same configuration, to ensure that users have the same experience no matter which instance in the cluster they are connected to.

Most of the application settings are stored in the database. Because each instance writes to the same database, those settings remain constant among the cluster instances.

Also make sure that each instance has the same settings in `portal-ext.properties`. This includes any framework settings that you set in the file instead of from the **Control Panel** user interface.

Configuring synchronized caching for the Studio instances

Studio instances in a multiple node environment must use synchronized caching.

About synchronized caching

Synchronized caching ensures that the information cached by one Studio instance is available to all of the Studio instances in the environment.

This reduces the number of times each instance needs to query the Studio database, which allows for faster response times and better performance. Studio uses Ehcache (*www.ehcache.org*) for caching synchronization.

Updating portal-ext.properties to synchronize caching for Studio instances

The `portal-ext.properties` file for each instance includes commented-out settings for synchronizing the caches for Studio instances.

For each Studio instance, uncomment the following clustering settings in `portal-ext.properties`. You should be able to use the default values provided.

```
##
## Cluster
##
# Uncomment the following properties to enable clustering
# Note: Clustering will not work with Hypersonic.  Configure a common database for all cluster nodes.

#net.sf.ehcache.configurationResourceName=/ehcache/hibernate-clustered.xml
#ehcache.multi.vm.config.location=/ehcache/liferay-multi-vm-clustered.xml
#org.quartz.jobStore.isClustered=true
```

This table lists settings in `portal-ext.properties` used to enable synchronized caching of Studio instances (in a cluster of Studio instances). For each setting, the table provides a description of the required value.

Setting	Description
`net.sf.ehcache.configurationResourceName`	The name and location of the XML configuration file for Hibernate caching. Hibernate is used by Studio to read from and write to the Studio application database. In the default `portal.properties` file, the configuration file is set to `hibernate.xml`, to implement caching in a non-clustered Studio implementation. When you uncomment this property in `portal-ext.properties`, which changes the configuration file to `hibernate-clustered.xml`, then Hibernate synchronizes the cache with the other Studio instances in the Studio cluster.
`ehcache.multi.vm.config.location`	The name and location of the XML configuration file for Ehcache. In the default `portal.properties` file, the file is set to `liferay-multi-vm.xml`, to implement caching in a non-clustered Studio implementation. When you uncomment this property in `portal-ext.properties`, which changes the configuration file to `liferay-multi-vm-clustered.xml`, then the cache is synchronized with the other Studio instances in the Studio cluster.
`org.quartz.jobStore.isClustered`	Enables clustering of Studio instances on the built-in Quartz job scheduling engine.

These configuration files are configured to automatically detect the other Studio instances in the Studio cluster, and to use IP address 233.0.0.1 and port 4446 to send the updated cache information.

Customizing the shared cache configuration files

The default versions of the shared cache configuration files should work in most cases. However, you can if needed create and deploy customized versions.

The most likely customization that might be needed would be to the IP address and port number configured near the top of each file:

```
<cacheManagerPeerProviderFactory
        class="net.sf.ehcache.distribution.RMICacheManagerPeerProviderFactory"
        properties="peerDiscovery=automatic,multicastGroupAddress=230.0.0.1,multicastGroupPort
=4446,timeToLive=1"
        propertySeparator=","
    />
```

If you make any changes to these configuration files, make sure to make the same changes for all of the instances in the cluster.

To customize the clustered cache configuration files:

1. Extract the default files from the `ehcache` directory in `portal-impl.jar`.

   The file is in the `WEB-INF/lib` directory, which is located in `endeca-portal.war`, which is in `bdd-studio.ear`.

2. Make the necessary updates to the files.

   To ensure that Studio uses the correct files, you may want to rename the customized files to something like:

   - `hibernate-clustered-custom.xml`
   - `liferay-multi-vm-clustered-custom.xml`

3. To deploy the customized files:

   (a) Undeploy `bdd-studio.ear`.

   Use the appropriate method to undeploy the file based on whether you auto-deployed the .ear file or installed it.

   (b) Update `bdd-studio.ear` to add a subdirectory `APP-INF/classes/ehcache/` that contains the customized XML files.

   (c) Redeploy the updated .ear file.

4. If needed, update `portal-ext.properties` to reflect the customized file names:

```
net.sf.ehcache.configurationResourceName=/ehcache/hibernate-clustered-custom.xml
ehcache.multi.vm.config.location=/ehcache/liferay-multi-vm-clustered-custom.xml
```

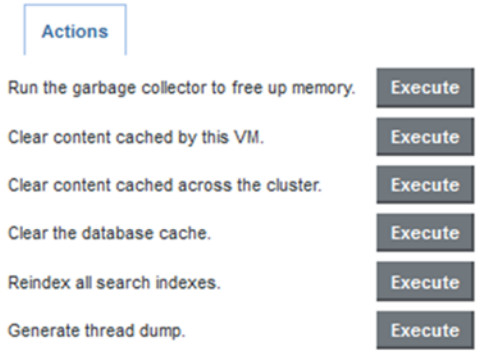
Clearing the cache for multiple Studio instances

As part of troubleshooting issues with a multi-instance Studio implementation, you can clear the cache for Studio instances. From the Studio **Control Panel**, you can clear the cache for either the current instance or for the entire Studio cluster.

To clear the Studio cache:

1. Click the **Control Panel** icon.
2. On the **Control Panel** menu, in the **Server** section, click **Server Administration**.

3. At the bottom of the page, on the **Actions** tab:



- To clear the cache for the current instance only, click the **Execute** button next to **Clear content cached by this VM**.
- To clear the cache for the entire Studio cluster, click the **Execute** button next to **Clear content cached across the cluster**.

Chapter 5

Using Studio with a Reverse Proxy

Studio can be configured to use a reverse proxy.

About reverse proxies

A reverse proxy provides a more secure way for users to get access to application servers.

What is a reverse proxy?

A reverse proxy retrieves resources on behalf of a client from one or more servers, and then returns these resources to the client as though they came from the server itself.

A reverse proxy is located between the client and the proxied server(s). Clients access content through the proxy server. The reverse proxy server assumes the public hostname of the proxied server. The hostname(s) of the actual/proxied servers are often internal and unknown to the client browser.

Some common reasons for implementing a reverse proxy include:

- Security or firewalling
- SSL termination
- Load balancing and failover
- Resource caching/acceleration
- URL partitioning

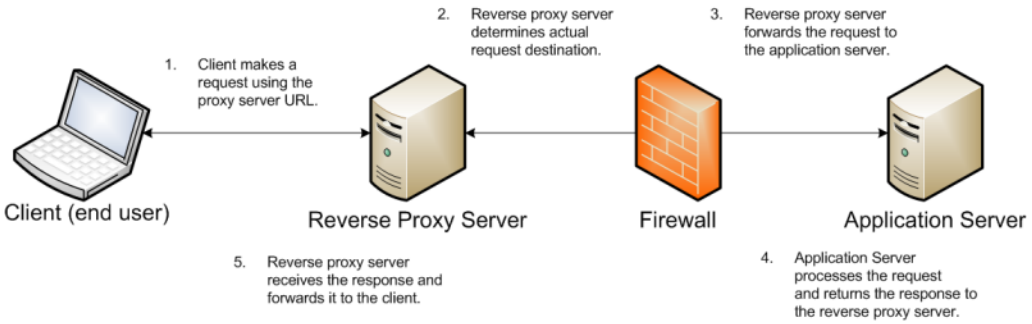
Types of reverse proxies

Reverse proxies may be either devices/appliances or specially configured web servers.

A very popular software-based reverse proxy is the Apache HTTP Server configured with the `mod_proxy` module. Many commercial web servers and reverse proxy solutions are built on top of Apache HTTP Server, including Oracle HTTP Server.

Example sequence for a reverse proxy request

Here is an example of the typical sequence for a request processed using a reverse proxy server.



1. The client makes a request to the public URL.

   For this example, for a Studio project, the request URL might be something like `http://mybdd/bdd/web/myproject`, using the default port 80.

   The hostname resolves to the address of the reverse proxy server. The reverse proxy is listening on this address and receives the request.

2. The reverse proxy server analyzes the URL to determine where the request needs to be proxied to.

   A reverse proxy might use any part of the URL to route the request, such as the protocol, host, port, path, or query-string. Typically the path is the main data used for routing.

   The reverse proxy configuration rules determine the outbound URL to send the request to. This destination is usually the end server responsible for serving the content. The reverse proxy server may also rewrite parts of the request. For example, it may change or make additions to path segments.

   Reverse proxies can also add standard or custom headers to the request.

   For example, the URL `http://mybdd/web/myproject` might be proxied to `http://bddserver1:8080/bdd/web/myproject`. In this case:

   - The hostname of the target server is bddserver1
   - The port is changed to 8080
   - The context path `/bdd/` is added

3. The reverse proxy server sends the request to the target server.
4. The target server sends the response to the reverse proxy server.
5. The reverse proxy server reads the request and returns it to the client.

Recommendations for reverse proxy configuration

Here are some general configuration recommendations for setting up a reverse proxy.

Preserving HTTP 1.1 Host: headers

Enabling the Apache ProxyPreserveHost directive

Preserving HTTP 1.1 Host: headers

HTTP 1.1 requests often include a `Host:` header, which contains the hostname from the client request. This is because a server may use a single IP address or interface to accept requests for multiple DNS hostnames.

The `Host:` header identifies the server requested by the client. When a reverse proxy proxies an HTTP 1.1 request between a client and a target server, when it makes the request, it must add the `Host:` header to the outbound request. The `Host:` header it sends to the target server should be the same as the Host: header it received from the client. It should not be the Host: header that would be sent if accessing the target server directly.

When the application server needs to create an absolute, fully-qualified URL, such as for a redirect URL or an absolute path to an image or CSS file, it must provide the correct hostname to the client to use in a subsequent request.

For example, a Java application server sends a client-side redirect to a browser (HTTP 302 Moved). It uses the `ServletRequest.getServerName()` method to fetch the hostname in the request, then constructs a `Host:` header.

The URL sent by the client is `http://mystudio/web/myapp`. The actual internal target URL generated by the reverse proxy will be `http://studioserver1:8080/bdd/web/myapp`.

If there is no specific configuration for the target server, then if the reverse proxy retains the `Host:` header, the header is:

```
Host: http://mystudio
```

If the reverse proxy does not retain the `Host:` header, the result is:

```
Host: http://studioserver1:8080
```

In the latter case, where the header uses the actual target server hostname, the client may not have access to `studioserver1`, or may not be able to resolve the hostname. It also will bypass the reverse proxy on the next request, which may cause security issues.

If the `Host:` header cannot be relied on as correct for the client, then it must be configured specifically for the web or application server, so that it can render correct absolute URLs.

Most reverse proxy solutions should have a configuration option to allow the `Host:` header to be preserved.

Enabling the Apache ProxyPreserveHost directive

The `ProxyPreserveHost` directive is used to instruct Apache `mod_proxy`, when acting as a reverse proxy, to preserve and retain the original `Host:` header from the client browser when constructing the proxied request to send to the target server.

The default setting for this configuration directive is Off, indicating to not preserve the `Host:` header and instead generate a `Host:` header based on the target server's hostname.

Because this is often not what is wanted, you should add the `ProxyPreserveHost On` directive to the Apache HTTPD configuration, either in `httpd.conf` or related/equivalent configuration files.

Reverse proxy configuration options for Studio

Here are some options for configuring reverse proxy for Studio.

Simple Studio reverse proxy configuration

Here is a brief overview of a simple reverse proxy configuration for Studio. The configuration preserves the `Host:` header, and does not use SSL or path remapping. Studio only supports matching context paths.

In this simple configuration:

- A reverse proxy server is in front of a single Studio application server.
- The reverse proxy server is configured to preserve the `Host:` header.
- The context paths match.
- Neither the reverse proxy nor the application server is configured for SSL.

With this setup, you should be able to access Studio correctly using the reverse proxy without additional configuration.

Studio reverse proxy configuration without preserving Host: headers

If a reverse proxy used by Studio does not preserve the `Host`: header, and instead makes a request with a `Host:` header referring to the target application server, Studio and the application server receive an incorrect hostname. This causes Studio to generate absolute URLs that refer to the proxied application server instead of to the reverse proxy server.

If the reverse proxy cannot be configured to preserve the `Host:` header, you must configure a fixed hostname and port. To do this, you can either:

- Configure the application server to have a fixed hostname and port
- Use `portal-ext.properties` to configure Studio with a fixed hostname and port

Configuring a fixed hostname for the application server

In WebLogic, set up a virtual host with the fixed hostname and port.

Configuring Studio with a fixed hostname

To configure Studio with a fixed hostname and port, add the following properties to `portal-ext.properties`:

```
web.server.host=<reverseProxyHostName>
web.server.http.port=<reverseProxyPort>
```

Configuring Studio to support an SSL-enabled reverse proxy

If Studio is installed behind a reverse proxy that has SSL capabilities, and the client SSL is terminated on the reverse proxy, you must configure Studio to set the preferred protocol to HTTPS, and provide the host and port for the reverse proxy server.

To do this, add the following settings to `portal-ext.properties`:

```
web.server.protocol=https
web.server.host=<reverseProxyHostName>
web.server.https.port=<reverseProxyPort>
```

Where:

- *reverseProxyHostName* is the host name of the reverse proxy server.
- *reverseProxyPort* is the port number for the reverse proxy server.

Part IV

Uninstalling Big Data Discovery

Chapter 6

Uninstalling Big Data Discovery

This section describes how to uninstall BDD.

The uninstallation script

You uninstall BDD by running the `uninstall.sh` script, which is located in `$BDD_HOME/uninstall`.

You must run the script from the Admin Server. It doesn't require any arguments, but it does need access to `bdd.conf`, which it assumes is located in `$BDD_HOME/BDD_manager/conf`.

When the script runs, it:

1. Reads `bdd.conf`.
2. Terminates all currently running processes.
3. Deletes the WebLogic domain.
4. Cleans up the Hive Table Detector cron job.
5. Deletes the Data Processing CLI.
6. Deletes all Data Processing libraries.
7. Deletes the `/user/bdd` directory from HDFS.
8. Deletes the contents of the `$ORACLE_HOME` directory, including WebLogic Server and all BDD components, and the WebLogic domain.
9. Deletes the znode for the Dgraph cluster from the ZooKeeper namespace.

Note: If you upgraded BDD at any point, the script also removes any remaining files from the previous BDD versions.

Although the script deletes most of the BDD data from your system, it leaves behind some BDD-related files and directories, including:

- The empty BDD directories. For example, the script removes everything inside of `$ORACLE_HOME`, but leaves the directory itself. You can remove these manually when the script finishes running, although this isn't required if you're going to reinstall.
- The Dgraph index on the shared NFS, even if you're using the base index created by the installer. If you plan on reinstalling BDD, you can leave this on the NFS and reuse it.
- The sample files created by Data Processing.
- The `/oraInventory` directory and the `oraInst.lok` file.

Running the uninstallation script

You uninstall BDD by running `uninstall.sh` from the Admin Server.

Note: If you upgraded BDD at any point, the script also removes any remaining files from the previous BDD versions.

To run the uninstallation script:

1. On the Admin Server, open a command prompt and navigate to `$BDD_HOME/uninstall`.
2. Run the uninstallation script:

   ```
   ./uninstall.sh
   ```

3. Enter `yes` or `y` when asked if you're sure you want to uninstall BDD.

Appendix A

Optional and Internal BDD Properties

The following sections describe the optional and internal properties in `bdd.conf`.

Optional settings

The second part of `bdd.conf` contains optional properties. You can update these if you want, but the default values will work for most installations.

General settings

This section configures settings relevant to all components and the installation process itself.

Configuration property	Description
`FORCE`	Determines whether the installer will remove files and directories left over from previous installations when it runs. Use `FALSE` if this is your first time installing BDD. Use `TRUE` if you're reinstalling after either a failed installation or an uninstallation. Note that this property only accepts UPPERCASE values.
`ENABLE_AUTOSTART`	Determines whether the BDD cluster will automatically restart after its servers are rebooted: • `TRUE`: WebLogic (including Studio and the Dgraph Gateway), the Dgraph, and the HDFS Agent will automatically restart after their host servers are rebooted. • `FALSE`: WebLogic, the Dgraph, and the HDFS Agent must be restarted manually. Note that this property only accepts UPPERCASE values.

WebLogic settings

This section configures the WebLogic Server, including the Admin Server and all Managed Servers. It doesn't configure Studio or the Dgraph Gateway.

Configuration property	Description and possible settings
`WLS_START_MODE`	Defines the mode WebLogic Server will start in: • If set to `prod`, the WebLogic Server starts in production mode, which requires a username and password when it starts. • If set to `dev`, it starts in development mode, which doesn't require a username or password. The installer will still prompt you for a username and password at runtime, but these will not be required when starting WebLogic Server. Note that this property only accepts lowercase values.
`WLS_NO_SWAP`	Controls whether the installer will check for the required amount of free swap space (512MB) on the Admin Server and all Managed Servers before installing WebLogic Server. If set to `TRUE`, the script won't perform the swap space check. Use this value if you're installing WebLogic Server on nodes that don't meet the swap space requirement. For more information, see *Physical memory and disk space requirements on page 23*.
`WEBLOGIC_DOMAIN_NAME`	The name of the WebLogic domain, which Studio and the Dgraph Gateway run in.
`ADMIN_SERVER_PORT`	The Admin Server's port number. This number must be unique.
`MANAGED_SERVER_PORT`	The port used by the Managed Server (i.e., Studio). This number must be unique. This property is still required if you are installing on a single server.
`WLS_SECURE_MODE`	Enables and disables SSL for Studio's outward-facing ports. This can be set to `TRUE` or `FALSE`. When set to `TRUE`, the Studio instances on the Admin Server and the Managed Servers listen for requests on the `ADMIN_SERVER_SECURE_PORT` and `MANAGED_SERVER_SECURE_PORT`, respectively. Note that this property doesn't enable SSL for any other BDD components.
`ADMIN_SERVER_SECURE_PORT`	The secure port on the Admin Server that Studio listens on when `WLS_SECURE_MODE` is set to `TRUE`. Note that when SSL is enabled, Studio still listens on the un-secure `ADMIN_SERVER_PORT` for requests from the Dgraph Gateway.

Configuration property	Description and possible settings
`MANAGED_SERVER_SECURE_PORT`	The secure port on the Managed Server Studio listens on when `WLS_SECURE_MODE` is set to `TRUE`. Note that when SSL is enabled, Studio still listens on the un-secure `MANAGED_SERVER_PORT` for requests from the Dgraph Gateway.
`ENDECA_SERVER_LOG_LEVEL`	The log level used by the Dgraph Gateway: • `INCIDENT_ERROR` • `ERROR` • `WARNING` • `NOTIFICATION` • `TRACE` More information on Dgraph Gateway log levels is available in the *Administrator's Guide*.
`SERVER_TIMEOUT`	The timeout value (in milliseconds) used when responding to requests sent to all Dgraph Gateway web services except the Data Ingest Web Service. A value of `0` means there is no timeout.
`SERVER_INGEST_TIMEOUT`	The timeout value (in milliseconds) used when responding to requests sent to the Data Ingest Web Service. A value of `0` means there is no timeout.
`SERVER_HEALTHCHECK_TIMEOUT`	The timeout value (in milliseconds) used when checking data source availability when connections are initialized. A value of `0` means there is no timeout.
`STUDIO_ADMIN_SCREEN_NAME`	The Studio admin's screen name.
`STUDIO_ADMIN_EMAIL_ADDRESS`	The email address of the Studio admin, which will be their username. This must be a full email address and can't begin with `root@` or `postmaster@`. **Note:** If you set the `BDD_STUDIO_ADMIN_USERNAME` environment variable for a silent installation, you don't need to set this property. If you do, the installer will overwrite this value with the value of `BDD_STUDIO_ADMIN_USERNAME`.
`STUDIO_ADMIN_PASSWORD_RESET_REQUIRED`	Determines whether the Studio admin will be asked to reset their password the first time they log in.
`STUDIO_ADMIN_FIRST_NAME`	The Studio admin's first name.
`STUDIO_ADMIN_MIDDLE_NAME`	The Studio admin's middle name.
`STUDIO_ADMIN_LAST_NAME`	The Studio admin's last name.

Dgraph and HDFS Agent settings

This section configures the Dgraph and the HDFS Agent.

Configuration property	Description and possible settings
`DGRAPH_WS_PORT`	The port the Dgraph listens on for requests.
`DGRAPH_BULKLOAD_PORT`	The port that the Dgraph listens on for bulk load ingest requests.
`DGRAPH_OUT_FILE`	The path to the Dgraph's stdout/stderr file.
`DGRAPH_LOG_LEVEL`	Optional. Defines the log levels for the Dgraph's out log subsystems. This must be in the format `"subsystem1 level1|subsystem2,subsystem3 level2|subsystemN levelN"` (including quotes). You can include as many subsystems as you want. Any you don't include will be set to `NOTIFICATION`. For more information on the Dgraph's out log subsystems and their supported levels, see the *Administrator's Guide*.
`DGRAPH_ADDITIONAL_ARG`	**Note:** This property is only intended for use by Oracle Support. Don't provide a value for this property when installing BDD. Optional. Defines one or more flags to start the Dgraph with. More information on Dgraph flags is available in the *Administrator's Guide*.
`AGENT_PORT`	The port that the HDFS Agent listens on for HTTP requests.
`AGENT_EXPORT_PORT`	The port that the HDFS Agent listens on for requests from the Dgraph.
`AGENT_OUT_FILE`	The path to the HDFS Agent's stdout/stderr file.

Data Processing settings

This section configures Data Processing and the Hive Table Detector.

Configuration property	Description and possible settings
`ENABLE_HIVE_TABLE_DETECTOR`	Enables the DP CLI to automatically run the Hive Table Detector according to the schedule defined by the subsequent properties. When set to `TRUE`, the Hive Table Detector runs automatically on the server defined by `DETECTOR_SERVER`. When it runs, the default behavior performs these two steps: • Provisions any new Hive table in the "default" database, if that table passes the whitelist and blacklist. • Deletes any BDD data set that does not have a corresponding source Hive table. This is an action that you cannot prevent. When set to `FALSE`, the Hive Table Detector does not run.
`DETECTOR_SERVER`	The hostname of the server the Hive Table Detector runs on. This must be one of the WebLogic Managed Servers.
`DETECTOR_HIVE_DATABASE`	The name of the Hive database that the Hive Table Detector monitors. The default value is `default`. This is the same as the default value of `HIVE_DATABASE_NAME`, which is used by Studio and the CLI. You can use a different database for each these properties, but Oracle recommends you start with one for a first time installation. This value can't contain semicolons (;).
`DETECTOR_MAXIMUM_WAIT_TIME`	The maximum amount of time (in seconds) that the Hive Table Detector waits between update jobs.
`DETECTOR_SCHEDULE`	A Cron format schedule that specifies how often the Hive Table Detector runs. This must be enclosed in quotes. The default value is `"0 0 * * *"`, which means the Hive Table Detector runs at midnight, every day of every month.
`ENABLE_ENRICHMENTS`	Determines whether data enrichments are run during the sampling phase of data processing. This setting controls the Language Detection, Term Extraction, Geocoding Address, Geocoding IP, and Reverse Geotagger modules. When set to `true`, all of the data enrichments run. When set to `false`, none of them run. For more information on data enrichments, see the *Data Processing Guide*.

Configuration property	Description and possible settings
`MAX_RECORDS`	The maximum number of records included in a data set. For example, if a Hive table has 1,000,000 records, you could restrict the total number of sampled records to 100,000. Note that the actual number of records in each data set may be slightly higher or less than this value.
`SANDBOX_PATH`	The path to the HDFS directory that the Avro files created when users export data from BDD are stored in.
`LANGUAGE`	Specifies either a supported ISO-639 language code (`en`, `de`, `fr`, etc.) or a value of `unknown` to set the language property for all attributes in the data set. This controls whether Oracle Language Technology (OLT) libraries are invoked during indexing. A language code requires more processing but produces better processing and indexing results by using OLT libraries for the specified language. If the value is `unknown`, the processing time is faster but the processing and indexing results are more generic and OLT is not invoked. For a complete list of the languages BDD supports, see the *Data Processing Guide*.

Internal settings

The third part of `bdd.conf` contains internal settings either required by the installer or intended for use by Oracle Support.

Note: Don't modify any properties in this part unless instructed to by Oracle Support.

Configuration property	Description
`DP_POOL_SIZE`	The maximum number of calls Studio can make to Data Processing at once.
`DP_TASK_QUEUE_SIZE`	The maximum number of jobs Studio can add to the Data Processing queue.

Configuration property	Description
`MAX_INPUT_SPLIT_SIZE`	The maximum partition size for Spark inputs, in MB. This controls the size of the blocks of data handled by Data Processing jobs. Partition size directly affects Data Processing performance. When partitions are smaller, more jobs run in parallel and cluster resources are used more efficiently. This improves both speed and stability. The default value is 32. This amount should be sufficient for most clusters, with a few exceptions: • If your Hadoop cluster has a very large processing capacity and most of your data sets are small (around 1GB), you can decrease this value. • In rare cases, when data enrichments are enabled, the enriched data set in a partition can become too large for its YARN container to handle. If this occurs, you can decrease this value to reduce the amount of memory each partition requires. Note that this property overrides the HDFS block size used in Hadoop.
`SPARK_DYNAMIC_ALLOCATION`	Determines whether Data Processing will dynamically compute the resources allocated to the Spark executors during processing. This value should always be set to `true`. `false` is only intended for use by Oracle Support. When set, Data Processing allocates Spark resources according to the static configuration defined by the following properties: • `SPARK_DRIVER_CORES` • `SPARK_DRIVER_MEMORY` • `SPARK_EXECUTORS` • `SPARK_EXECUTOR_CORES` • `SPARK_EXECUTOR_MEMORY`
`SPARK_DRIVER_CORES`	The number of cores used by the Spark job driver.
`SPARK_DRIVER_MEMORY`	The maximum memory heap size for the Spark job driver. This must be in the same format as JVM memory settings; for example, `512m` or `2g`.
`SPARK_EXECUTORS`	The total number of Spark executors to launch.
`SPARK_EXECUTOR_CORES`	The number of cores for each Spark executor.
`SPARK_EXECUTOR_MEMORY`	The maximum memory heap size for each Spark executor. This must be in the same format as JVM memory settings; for example, `512M` or `2g`.

Configuration property	Description
`DP_ADDITIONAL_JARS`	Optional. A colon-separated list of the absolute paths to additional jars, such as custom SerDe jars, you want to use during data processing. These will be added to the CLI classpath. Note that you must manually copy each SerDe jar to the same location on all cluster nodes before installing.
`BDD_VERSION`	The version of BDD you're installing. This property is intended for use by Oracle Support and shouldn't be changed.
`BDD_RELEASE_VERSION`	The BDD hotfix or patch version you're installing. This property is intended for use by Oracle Support and shouldn't be changed.

Index