OpenNode2

Installation Guide
(Java)

Revision Date: 10/3/2016

Prepared By:

WINDSOR SOLUTIONS

4386 SW Macadam Ave, Suite 101
Portland, OR 97239
(503) 675-7833

Environmental Information

eXchange Network

Revision History

Date	Author	Changes
April 2009	Windsor	Initial version
July 2009	Windsor	Fixed error in WebSphere configuration instructions, p.14.
September 2009	Windsor	Added Node upgrade instructions.
May 2011	Windsor	Changes to support OpenNode2 v2.0
September 2011	Windsor	Added 2.02 upgrade instructions
October 2012	Windsor	Added NAAS Admin note
September 2013	Windsor	Changes for OpenNode2 2.08
October 2013	Windsor	Updated cover page
February 2015	Windsor	Update for v2.10
April 2015	Windsor	Minor upgrade change to fix missing database script step
October 2016	Windsor	Minor updates for v2.11. Removed prior version upgrade steps.

Contents

Document Overview

This document describes the installation and configuration of the Exchange Network OpenNode2 open-source Node for Java. The primary audience is a System Administrator or Deployment Specialist experienced with installing and managing Java Web applications. The sections entitled "Relational Database" and "Install the Node Administration Database" are intended for Database Administrators. Network managers and security personnel will likely be interested in the "Network and Firewall Configuration" section.

The step-by-step instructions in much of this document, especially the "Configure the Node" and "Configure Node Administration Application, Web Service Endpoint 1.1, and Web Service Endpoint 2" sections, apply specifically to installing and configuring the free, open source Node.

The main components of the OpenNode2 implementation are:

1. Perform prerequisite software installations.
2. Install the OpenNode2 database.
3. Install and configure the four Web applications:
    o Node Administration Utility (WNA)
    o Node Service v1.1 Endpoint (WNE)
    o Node Service v2.1 Endpoint (WNE2)
    o Node REST Service Endpoint (WNREST)
    o Node Orchestration Service (WNOS)
4. Test the installation.

Assumptions

Single Server Deployment

Although the OpenNode2 architecture allows for deploying the component applications across multiple physical machines, for clarity of this document, these guidelines assume that all components will be installed on a single machine, even when deploying to a managed cluster of application servers.

Because of the additional complexity inherent in a distributed environment, and the potential variability in configuration, additional assistance should be sought if installing in these environments. Please see the Exchange Network Website for additional support information.

Network Connectivity

Due to the Service Oriented Architecture (SOA) of the Exchange Network, and due to the Node's dependency on external connectivity to the Network Authentication and Authorization Services (NAAS), this document assumes that the server on which the Node will be deployed has already been configured on the local network and is accessible from the Internet.

Installation Prerequisites

Before proceeding with the OpenNode2 installation and configuration, ensure that the components that follow are installed on the physical server machine. While some steps can be performed at the same time that the Node is installed, most should be completed prior to installation. Those steps that can occur at the same time as the Node installation are noted.

Hardware and software requirements for the Node are to a large degree dictated by the Java application server (e.g., Apache Tomcat) to which OpenNode2 will be deployed.

NOTE: WebLogic is no longer supported by OpenNode2.

Hardware Requirements

Java application servers are available for a wide variety of processor architectures. A single-core Pentium IV processor with a clock speed of 1 gigahertz (GHz) is required at a minimum, though faster processors capable of addressing over 4 gigabytes (GB) of random-access memory (RAM) are preferred. OpenNode2 has been tested on 64-bit Intel-architecture CPUs. NOTE: 32 bit architecture CPUs and Operating systems are discouraged due to their inability to provide enough RAM for many plugins to operate when generating very large files. SPARC chips are no longer tested due to the waning deployments of these machines.

The minimum RAM requirement for a full OpenNode2 application server deployment is 2GB of heap space (a max of 3GB recommended) and 512MB of PermGen Space (a max PermGen of 1024MB is recommended). This is in addition to RAM required by the operating system and application server software.

The minimum hard disk space required is 300 MB. In order to accommodate typical growth in OpenNode2's log and temporary files, 2GB of available storage is recommended. As with RAM, these storage requirements are in addition to the disk space needed by the operating system and application server.

Software Requirements

Operating System

OpenNode2 for Java has been tested on the latest LTS version of Ubuntu (64-bit edition only) as well as other, Debian based Linux distributions. New, 64 bit versions of Windows server are also supported. OpenNode2 for Java has also been tested extensively in virtualized environments as well as the cloud.

Java Runtime Environment

OpenNode2 requires a minimum Java Runtime Environment (JRE) version of 1.7. Extensive testing has been performed using this version. Note: Java 1.5 and 1.6 are no longer supported by Oracle, should no longer be used, and OpenNode2 will not function while using them.

Java Application Server

OpenNode2 for Java requires a Java application server that supports JRE version 1.7 or higher, and a minimum version 2.5 of the Java Web Application specification (i.e., version 2.5 of the Java Servlet API). Although most commercial application servers support the Java Platform, Enterprise Edition (Java EE, formerly Java 2, Enterprise edition or J2EE) specification, OpenNode2 itself does not require Java EE.

OpenNode2 for Java has been primarily tested with these application servers:

- Apache Tomcat version 7.x.

Although OpenNode2 should run on other application servers (such as Jboss or WebSphere) with no modifications, the server may need to be configured for non-default values of some deployment parameters.

Relational Database

OpenNode2 uses a standard relational database as its metadata repository. This database contains configuration data and activity logs (including XML documents generated by and submitted to OpenNode2). OpenNode2 requires a user account with read and write permission to this database; it does not require permission to create or drop tables. The OpenNode2 distribution package includes setup scripts and Java Database Connectivity (JDBC) drivers for these database management systems (DBMS):

- Oracle, version 11g and higher

- Microsoft SQL Server, version 2008 R2 and higher

- MySQL, version 5.1 and higher (5.6 or higher recommended)

For each supported DBMS, OpenNode2 provides one Structured Query Language (SQL) script for creating the database or schema, and another script for seeding the database with user account data required to start OpenNode2. It is assumed that a database administrator familiar with the tools for the DBMS and with sufficient permission to execute the scripts will be responsible for this aspect of installation.

Separate databases should be created for each runtime environment, such as development, testing, and production. The database names should follow some convention, such as "OpenNode2_dev", "OpenNode2_test", and "OpenNode2_prod". These names will later be used in the application's configuration files.

SSL Certificate (production installations only)

All production Exchange Network Nodes are required to have a Secure Sockets Layer (SSL) certificate provided by a recognized Certificate Authority (CA). A 128-bit encrypted SSL certificate must be installed and the Node URL must be listening on the standard TCP port 443. This is **not** a requirement for development and test servers; non-production environments may use the Exchange Network certificate (issued by the CDX helpdesk), a self-signed certificate, or no SSL certificate at all.

Procedures for installing (and optionally, creating) SSL certificates vary by application server. This guide assumes that the server administrator is sufficiently familiar with the application server to manage certificates.

Web and Proxy Servers, Firewalls, XML Gateways

In production environments, it is common to use a dedicated HTTP server such as Apache HTTPD or Microsoft Internet Information Server (IIS) to respond to an application's URL and serve content such as static Web pages, images, and other media, while delegating requests for dynamic content to a Java application server. Because OpenNode2 consists primarily of "faceless" Web services, and because the Node Administration application is the only component with a user interface and this has very little static content, this system architecture provides no real performance advantage.

Many organizations use HTTP servers or special-purpose devices as proxies to route requests from the public Internet through network firewalls to application servers in a secure network. In addition, a number

of hardware and software products such as XML gateways and Simple Object Access Protocol (SOAP) intermediaries are available for providing enhanced security, management, and message processing.

Each environment is unique, and it is beyond the scope of this document to provide specific guidance on deploying OpenNode2 in a broad range of network topologies. Because of the high degree of site specific variability in configuration, additional assistance may be required. Please see the Exchange Network Website for additional support information.

User Accounts

NAAS Node Account

An account *for each environment under which OpenNode2 will run* (e.g. "Production," "Test," etc.) will need to be set up with the EPA's Network Authentication and Authorization Services (NAAS). This account is typically named node@NODE_DOMAIN.com, where "*NODE_DOMAIN*" is the same domain that the Node URL uses. The following steps assume that an onsite Node administrator has an active NAAS account for administering their Node.

1. Browse to https://cdxnode.epa.gov/usr

2. Click the 'Login' link in the left frame and log in using the Node administrator account. If you do not have a node administrator account, contact the CDX helpdesk to have one created.

3. Click the 'Add User' link in the left frame and create a user with the following parameters:

- User ID: node@myagency.gov

- Password: (*an acceptable NAAS password*)

- User Type: operator

- Affiliate: *(choose your agency NodeID)*

Operating System Account

The OpenNode2 applications do not require a separate user account on the host operating system; they typically run under the user account of the Java application server to which they are deployed. This account is typically created or selected by a system administrator or installation script when the application server is installed. Consult your application server's documentation for further details.

Installation Steps

The following sections explain how to obtain and install the binary distribution of the Java OpenNode2.

Download the Installation Files

A single distribution is available named **Java_OpenNode2_v2.08_ecos-generic.zip**. This distribution contains no server-specific modifications. Although it runs without any special configuration on Apache Tomcat, it may require additional configuration on other application servers.

Note that the Java OpenNode2 now uses a Maven 3 build system. A "profile" is allows the customization of a build. The profile ecos-generic is the only one currently distributed as part of OpenNode2, however with the source files in Subversion anyone can create their own profiles to use.
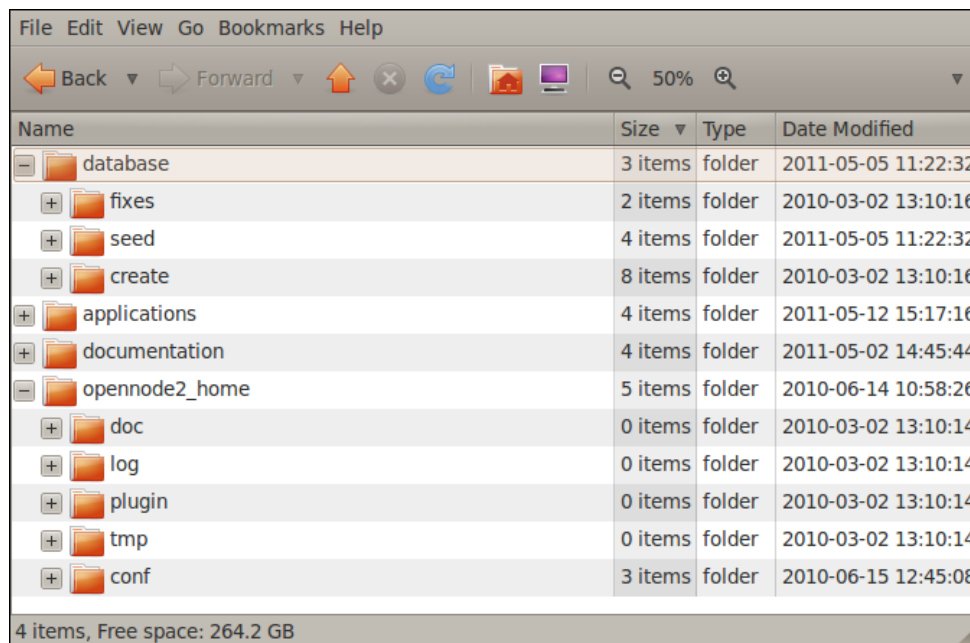
Verify the Installation Files (Optional)

The binary distribution includes a single .zip file and a file SHA1 file signature. Users are encouraged to verify the zip file after downloading by generating their own SHA1 checksum and comparing it to the appropriate checksum accompanying the distribution. If the user-generated checksum matches the one provided, the user can be assured that the zip file has not been modified since it was made available.

A number of utilities for generating checksums are freely available for all major operating systems.

Extract the Files

Expand the **Java_OpenNode2_v2.11_ecos-generic.zip** file to a convenient location on the application server, using WinZip, gzip, or an equivalent utility. This will result in the directory structure with the same name as the .zip file, as shown in Figure 1 and described in the following table.

Figure 1: Node Distribution Directory Structure

Path	Description
/applications	Contains the four OpenNode2 applications as four Web application archive (.war) files. These files will be deployed to the application server. • **wna.war:** This is the Node Administration Utility (WNA), the user interface for configuring Data Exchanges and managing OpenNode2. • **wne.war:** The Node Service v1.1 Endpoint (WNE), which implements version 1.1 of the Exchange Network Node specification. • **wne2.war:** The Node Service v2.1 Endpoint (WNE2), which implements version 2.1 of the Exchange Network Node specification. • **wnos.war:** The Node Orchestration Service (WNOS) • **wnrest.war:** The Node REST endpoint (WNREST)
/database/create	SQL Data Definition Language (DDL) scripts for creating the Node Administration database schema.
/database/seed	SQL scripts for inserting key user accounts into the Node Administration database
/database/fixes	SQL scripts to fix data when upgrading any Java version lower than v1.24 to any newer version.
/database/upgrade	SQL scripts to alter the Node Admin database when upgrading from a previous version if any.
/documentation	Contains license and release notes
/opennode2_home	The Node's working directories, which are empty. This folder will be copied to a location to which the application server has full read/write access.
/opennode2_home/doc	May be used for storing documents received by the Node
/opennode2_home/log	Where the four Web applications write their log files
/opennode2_home/plugin	Where Node plugins (which implement data exchange logic) are automatically extracted after installation
/opennode2_home/tmp	Holds temporary files and is automatically cleaned up by the Node's scheduler
/opennode2_home/plugins	Contains .zip files for Node plugins
/opennode2_home/conf	Location for shared configuration files for versions of OpenNode2 2.02 and up.

Copy opennode2_home and Set Permissions

The opennode2_home folder should be copied to */var*, which is outside the application server's directory tree, and should be backed up regularly. On Unix or Linux systems*, /var* is a typical location; on Windows, the root directory of a data drive (i.e., the drive where the application server is installed) should be the location where the */var* folder is created.

After copying *opennode2_home* to its destination, it is necessary to set directory access permissions so that the OpenNode2 applications can write logs, create and delete temporary files, and so on. Instructions for Windows and Unix/Linux systems are provided below, and assume the user is logged into the system with administrative privileges.

> **NOTE:**
>
> *In versions of OpenNode2 before 2.02 it was acceptable to put the opennode2_home folder in any location. In order to avoid configuration to deployed war files, minimize overall configuration, and avoid the need to back up multiple files in multiple locations during small upgrades, it was necessary to standardize the location to /var/opennode2_home (in a Windows system /var will refer to <drive_letter>:\var of the drive where the application server is installed).*

Microsoft Windows

Assume that opennode2_home has been copied to the path *D:\var*.

1. Determine the account under which the Java application server is running.

   a. Start the **Services** applet (Start menu →Administrative Tools→Services).

   b. Scroll down the list to locate the server's name (e.g., "IBM WebSphere Application Server" or "Apache Tomcat").

   c. Make a note of the name in the "**Log On As**" column; this will be either a user account name such as "websphere," or "LocalSystem.".

2. Browse to the **D:\var** directory.

3. Right click on **opennode2_home** and choose the **Sharing and Security**…

4. Click the **Security** tab.

5. If the application server is running under the **LocalSystem** account, verify that the "**SYSTEM**" user account has **Modify**, **Read & Execute**, **List Folder Contents**, **Read**, and **Write** permission to **opennode2_home** and its subdirectories.

6. If the server is running under a name *other* than **LocalSystem**,

   a. Click **Add.**

   b. Type the server's user name in the box labeled "**Enter the object names to select**" box.

   c. Click **Check Names**, then **OK.**

   d. Click the "**Allow**" checkboxes next to **Modify**, **Read & Execute**, **List Folder Contents**, **Read**, and **Write**.

   e. Click **Apply**, then **OK.**

   f. Verify that the permissions on the folders within **opennode2_home** have the same permissions.

Unix/Linux

Assume that opennode2_home has been copied to */var*.

1. Determine the account under which the Java application server is running, or which owns the application server's installation directory. The easiest way to do this is by listing the installation directory. From the command line, type

   ```
   $ ls -ld /opt/apache-tomcat-6.0.24/
   ```

   the console output will show the user and group owning the directory – in this case, both are named "tomcat"

   ```
   drwxr-xr-x 12 tomcat tomcat 4096 2008-07-29 12:41 /opt /apache-
   tomcat-6.0.24/
   ```

2. Change the owner and group for *opennode2_home* and its subdirectories:

   ```
   $ chown –R tomcat:tomcat /var/opennode2_home
   ```

3. Grant the application server user full access to *opennode2_home* and its subdirectories:

   ```
   $ chmod –R u+rwx /var/opennode2_home
   ```

4. Verify the permissions *opennode2_home* and its subdirectories with the **ls -l** command.

Install the Node Administration Database

The **database/create** directory within **Java_OpenNode2_v2.11_ecos-generic** contains setup scripts for creating tables, keys, and indexes for the Node Administration database. Separate scripts are provided for Oracle, Microsoft SQL Server, and MySQL. The file names indicate which DBMS they target. The following steps assume that a database administrator is connected to the database server with an appropriate management tool (e.g., Oracle SQL Developer, SQL Server Management Studio, etc.) and with privileges sufficient to create and delete databases and schemas.

1. Create a database or schema named "OpenNode2" (or a similar name adhering to local conventions) on the target database server.

2. Create a database user account that will have permission to create, read, update, and delete data in this database. This is the account that the Node will use to connect to the database; the name and password will be needed when configuring the WNOS application, as described in the "Configure the Node" section.

3. Load and run the appropriate script (e.g., **Oracle_CREATE.sql**).

4. Load the corresponding script from the **Java_OpenNode2_v<version-number>_ecos-generic/database/seed** directory (e.g., **seed_NAccount_Oracle.sql**). The script will resemble the following example:

```
INSERT
INTO
      NAccount
      (      Id ,
             NAASAccount ,
             IsActive ,
             SystemRole ,
             Affiliation,
             ModifiedBy ,
             ModifiedOn)
VALUES
      (      '00000000-0000-0000-0000-000000000000',
             'node@myagency.gov',
             'Y',
             'Admin',
             'MY_AGENCY',
             '00000000-0000-0000-0000-000000000000',
             sysdate)
INSERT
INTO
      NAccount
      (      Id ,
             NAASAccount ,
             IsActive ,
             SystemRole ,
             Affiliation,
             ModifiedBy ,
             ModifiedOn)
VALUES
      (      '00000000-0000-0000-0000-000000000001',
             'Anonymous',
             'Y',
             'Anonymous',
             'MY_AGENCY',
             '00000000-0000-0000-0000-000000000000',
             sysdate)
```

5. Substitute your agency's node administrator NAAS account name for
   "node@myagency.gov," and your affiliation code for "MY_AGENCY", then change the
   Anonymous account's affiliation code to the same, and finally run the script.
   This user account can be used to log in to the Node Administration application after
   deploying the Node applications as described in the following section. The Anonymous
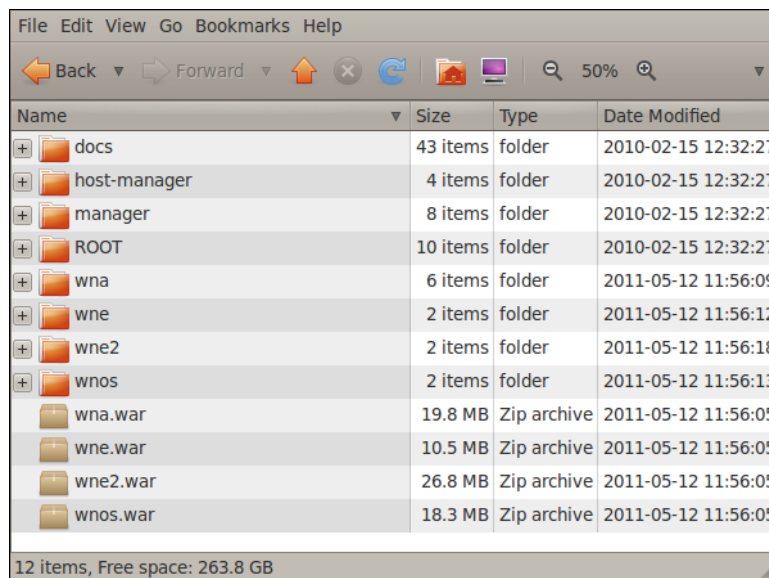   account is used for optional, Anonymous Service access.

NOTE:

The distribution contains 2 folders, one named "fixes" and the other "upgrade". These folders contain scripts to correct data in existing Java OpenNode2 deployments. These scripts do not need to be run on new installations.

Deploy and Configure the Web Applications

The procedure for deploying a Java Web application varies by application server. Commercial platforms such as WebSphere include sophisticated Web-based management consoles as well as specialized scripting tools for deploying and configuring applications. Apache Tomcat, in contrast, has a relatively simple Web-based management tool, and allows an administrator to install an application simply by copying a .war file into the appropriate directory.

In any case, when a .war file is deployed, it is expanded on the file system. In many cases the directory has the same base name as the .war file; some deployment tools allow specifying a name for the directory, and others append to the base name (WebSphere adds a ".war" extension to the administrator-supplied name). The exact location of the resulting directory tree also varies by application server, but contains, among other items, configuration files that need to be customized for each installation.

For illustration purposes, Figure 2 shows a Node installation on Apache Tomcat 6.0.

File Edit View Go Bookmarks Help			
← Back ▼ → Forward ▼ ⬆ ⊗ ↻ | 🏠 🖥 | 🔍 50% 🔍 ▼			
Name ▼	Size	Type	Date Modified
⊞ 📁 docs	43 items	folder	2010-02-15 12:32:27
⊞ 📁 host-manager	4 items	folder	2010-02-15 12:32:27
⊞ 📁 manager	8 items	folder	2010-02-15 12:32:27
⊞ 📁 ROOT	10 items	folder	2010-02-15 12:32:27
⊞ 📁 wna	6 items	folder	2011-05-12 11:56:09
⊞ 📁 wne	2 items	folder	2011-05-12 11:56:12
⊞ 📁 wne2	2 items	folder	2011-05-12 11:56:18
⊞ 📁 wnos	2 items	folder	2011-05-12 11:56:13
📄 wna.war	19.8 MB	Zip archive	2011-05-12 11:56:05
📄 wne.war	10.5 MB	Zip archive	2011-05-12 11:56:05
📄 wne2.war	26.8 MB	Zip archive	2011-05-12 11:56:05
📄 wnos.war	18.3 MB	Zip archive	2011-05-12 11:56:05
12 items, Free space: 263.8 GB			

Figure 2: Node applications deployed to Apache Tomcat

The following sections assume:

- The five OpenNode2 applications (wnerest.war not pictured above) have been deployed to the application server using the appropriate tools.

- The label **<application-home>** indicates the directory into which the .war files have been expanded (i.e., "tomcat-7.0.37/webapps" in Figure 2).

- The name of the directory resulting from expanding each .war file is the .war file's base name (e.g., wnos.war is expanded to a directory named "wnos").

- The applications are not running.

- The administrator is familiar with Java .properties file syntax.

In most files, only a few parameters need modification. These are indicated by a description surrounded by angle brackets (e.g., "<port>").

Configure the Node

In Node versions 2.02 and above the configuration files are in **/var/opennode2_home/conf**. If you are upgrading an old installation to 2.02 or above please consult the "Upgrade Steps" section below.

> **NOTE:**
>
> *Node versions prior to 2.02 used Log4J. Starting with v2.02, logging has been upgraded to use SFL4J. As such old log4j.properties files may all be discarded as the application now uses SFL4J logging with a Logback implementation behind it (logging settings may be modified in the logback.xml file – see* http://logback.qos.ch/ *for more information).*
>
> *Restart the application server services after making any configuration changes. Otherwise, changes will not be picked up by the affected application.*

Configuration file	Description, parameters, and values
jdbc.properties	Contains parameters for connecting to the Node Administration Database. This file includes complete sections for each database supported by the Node, beginning with the parameter named "jdbc.driverClassName" and ending with "jdbc.password". Locate the section for the appropriate database, and un-comment each line by deleting the initial "#" character. Modify the following parameters with the appropriate values for your environment. **jdbc.url:** - Replace **\<hostname\>** with the host name or IP address of the database server. - Replace **\<port\>** with the port number on which the database service listens. (DB2 and SQL Server) Replace **\<database name\>** with the name of the Node Administration database. (MySQL) Replace **\<schema name\>** with the name of the Node Administration schema. (Oracle) Replace **\<SID\>** with the SID of the Oracle instance on the database server. **jdbc.username:** Replace **\<database user name\>** with the name of the user (in Oracle, this is usually same as the schema name). This is the account created in 0the "Install the Node Administration Database" section of this document. **jdbc.password:** Replace **\<database user password\>** with the password for the account created in the "Install the Node Administration Database" section.
naas.properties	Contains parameters for connecting to NAAS. This file includes complete sections for connecting to both the test and production instances of NAAS. First, un-comment the lines for the instance of NAAS to which this Node will connect. Next, for the active NAAS instance, modify the following parameters with the appropriate values for your environment. **naas.runtime.username:** Replace \<someuser@some.domain\> with the NAAS ID that the Node will use for connecting to other Nodes. **naas.runtime.password:** Replace \<password\> with the password for the NAAS ID that the Node will use for connecting to other Nodes.

Configuration file	Description, parameters, and values
	• **naas.runtime.authmethod**: Leave set to "password". • **naas.admin.username:** Replace <someadmin@some.domain> with the NAAS ID of a local Node administrator (typically the same ID added to the Node Administration database in the section entitled, "Install the Node Administration Database." **Important:** If a non-Admin NAAS account is entered, Node Admin will produce errors when security-related actions are attempted. NAAS Admin accounts are different than NAAS User or Operator accounts. • **naas.admin.authmethod**: Leave set to "password". • **naas.admin.password:** Replace <password> with the password associated with the NAAS Administrator username.

nos.properties	Contains additional settings for the Node Orchestration Service.
	
	 HOME Section
	 • **node.version:** No longer used as of OpenNode2 2.08 and may be removed from existing nos.property files during an upgrade if desired.
	 • **path.node.home:** Replace <parent of OpenNode2_home> with the fully-qualified name of the directory containing the *opennode2_home* directory. This is the location selected in the "Copy opennode2_home and Set Permissions" section.
	 • **url.node.base:** This is the base URL that the Node applications use for communicating with each other.
	     o Replace **<port>** with port number assigned by the application server.
	     o In the extremely rare circumstance that the machine on which the Node runs is not configured to respond to internal requests for "localhost", replace "localhost" with the server's hostname or IP address.
	 • **url.external.admin:**
	     o Replace **<hostname>** with the hostname that other machines will use to reach the Node Admin application.
	     o Replace **<port>** with port number that other machines will use to reach the Node Admin application. If a port number is not required, delete "**<port>**" and the colon that precedes it.
	
	 PATHS and NOS Sections
	 • No changes should be made to these sections
	
	 SMTP Section
	 • **smtp.gateway:** Replace **<mail host>** with the host name or IP address of an SMTP server that the Node can use to send email.
	 • **smtp.from.email:** Replace **<from-address>** with the email address that will identify mail sent by the Node.
	     o If your mail server requires authentication, supply the SMTP username and password and change the **smtp.auth** value to *true*.
	
	 WHITELIST SUBNET Section
	 • **ip.whitelist.subnet:** This controls the IP address range from which Node Administrators can log in to the Node Admin application. Replace **<subnet-pattern>** with a valid subnet mask; for example, "192.168.*.*" will allow an

Configuration file	Description, parameters, and values
	administrator to log in from any address in any sub network of 192.168. Simply set to * for universal access. **ENDS Section** This section sets how the node describes itself to the Exchange Network Discovery Service (ENDS). These values are exposed through the ENDS_V20 plugin and associated services, if installed. • **node.name**: a short name for the node. • **organization.identifier**: a short name for the organization or agency hosting the node. • **node.deployment.type**: set to either "Development", "Test", or "Production". • **public.v2.endpoint.url**: the URL other Nodes use to reach this Node's v2 endpoint. • **bounding.coordinate**.*: sets the latitude/longitude bounding box. Describes the geographic area covered by the node's data.

Configure Node Administration Application, Web Service Endpoint 1.1, and Web Service Endpoint 2.1

The remaining OpenNode2 applications for Node versions 2.02 and above now should require no configuration for a standard install. For those upgrading their Node installs, wne.properties, wne2.properties, and wna.properties may be discarded. See "Configure the Node Orchestration Service" above for information on Logback, wne, wne2, and wna applications each contain a logback.xml file, out of the box this file should be adequately configured.

NOTE:

Restart the application server services after making any configuration changes. Otherwise, changes will not be picked up by the affected application.

Special Configuration for IBM WebSphere

Some versions of WebSphere require changing a default value for classloader order to "parent last" for each of the five OpenNode2 applications. This step can be performed using the WebSphere Administration application, either while deploying the applications or after deploying and prior to starting them. Please consult the documentation for your version of WebSphere for details.

Network and Firewall Configuration

OpenNode2's purpose is to exchange data with other Nodes via SOAP messages; thus the machines on which OpenNode2is installed must be able to connect to other Nodes. This is true for both test and production deployments. In the layered and secured networks common to most large organizations, this often requires enabling point-to-point HTTP and HTTPS traffic in the various routers, gateways, and/or firewalls that manage network traffic. At a minimum, test Nodes must be able to send and receive messages with these external addresses at the Environmental Protection Agency:

- https://tools.epacdxnode.net/xml/validator.wsdl

- https://naas.epacdxnode.net/xml/auth.wsdl

- https://naas.epacdxnode.net/xml/usermgr.wsdl

Production Nodes must be able to reach these addresses:

- https://cdxtools.epa.gov/xml/validator.wsdl

- https://cdxnodenaas.epa.gov/xml/auth.wsdl

- https://cdxnodenaas.epa.gov/xml/usermgr.wsdl

In addition, a Node may need to exchange messages with another government agency to develop, test, and operate specific Data Exchanges (aka "flows").

Finally, each machine hosting a Node must be able to reach internal resources, such as the Node Administration Database and the SMTP server. In some networks this may also require explicitly enabling traffic between two hosts on a specific port.

Update the OpenNode2 Plugins

All plugins must be updated to the latest version. You may use the Admin application to upload and configure new plugin versions for all exchanges. Please follow the instructions in the relevant Plugin Implementation Guide for instructions on installing and configuring the plugins correctly.

Finish by following the instructions in section "*Test the Installation*" to ensure the application is working properly.

Test the Installation

This section contains information on ensuring that the installation was successful.
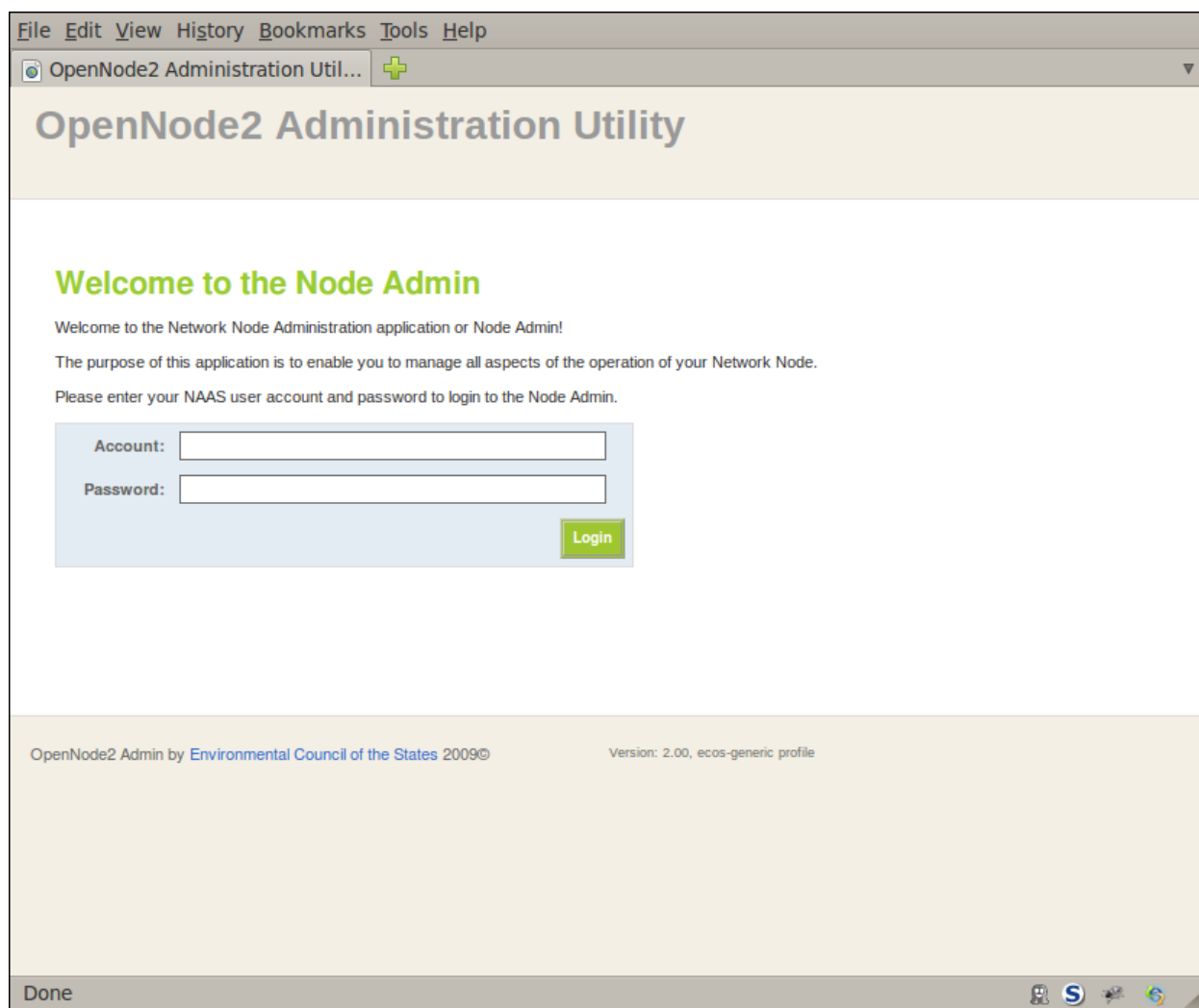
Verifying Startup

Follow these steps to verify that OpenNode2 is installed and functioning properly:

1. Use the application server's management console or command-line utility to start the four Node applications. The startup commands should either confirm success or display errors.

2. Optionally check the application server's log.

3. Examine the Node's log files, in */var/OpenNode2_home/log* directory. With the default logging configurations, you should see these results, with minor differences for file locations and timestamps:

   - **wna.log** should end with this line:
     ```
     2013-09-20 11:50:21,295 INFO [main] o.s.w.s.DispatcherServlet
     [FrameworkServlet.java:296] FrameworkServlet 'winnodeadmin':
     initialization completed in 1149 ms
     ```

   - **wne.log** should end with this line:
     ```
     2013-09-20 11:50:21,295 INFO [main] o.s.w.s.DispatcherServlet
     [HttpServletBean.java:118] Servlet 'remoting' configured successfully
     ```

   - **wne2.log** should end with this line:
     ```
     2013-09-20 11:50:21,295 INFO [main] o.a.a.d.ServiceDeployer
     [ServiceDeployer.java:91] Deploying Web service: wne2 -
     file:webapps/wne2/WEB-INF/services/wne2/
     ```

   - **wnos.log** should end with this line:
     ```
     2013-09-20 11:50:21,295 INFO [main] o.s.w.s.DispatcherServlet
     [FrameworkServlet.java:296] FrameworkServlet 'remoting': initialization
     completed in 2486 ms
     ```

   - **wnrest.log** should end with this line:
     ```
     2013-09-20 11:50:21,295 INFO [localhost-startStop-1]
     o.s.w.s.DispatcherServlet [FrameworkServlet.java:473] FrameworkServlet
     'services': initialization completed in 672 ms
     ```

Testing the Node Administration Application

Use a Web browser to navigate to the URL for the Node Administration application. First, if possible, do this from the machine where the application is installed, to eliminate network connectivity issues from the test. The URL should be **http://localhost:<port>/wna** (where <port> is the port number assigned by the Java application server).

The screen should appear as in Figure 3:

Figure 3: Node Admin Login Page

Try logging in using the admin account that was inserted into the database as specified in the section entitled, "NAAS Node Account." Use the NAAS password for the account.

If an error message appears upon attempting to log in, this may indicate one of the following:

- The WNA application may not be able to connect to WNOS.

- WNOS may not be able to access the Internet to validate the credentials against NAAS. Ensure that the Node server has access to the URLs listed in the "Network and Firewall Configuration" section.

Should errors occur, check the log files for the application server and the OpenNode2 applications.

Testing the Node Service Endpoints

Use a Web browser to navigate to the URLs for the Node Service Endpoints. Again, do this from the machine where the application is installed, to eliminate network connectivity issues from the test.

> **NOTE:**
>
> *All endpoints and their configured URLs now show up on the Dashboard Heartbeat monitor, this is the most reliable to way to test your endpoints.*

The URL for the version 1.1 endpoint is http://localhost:<port>/wne/services/v11. It should display the page shown in Figure 4.

v11

Hi there, this is an AXIS service!

Perhaps there will be a form for invoking the service here...

Figure 4: WNE confirmation

The URL for the version 2.1 endpoint is http://localhost:<port>/wne2/services/v21. The page will display differently depending on the browser used. Using the browser's "View Source" command, the following text (or similar) should be displayed:

```
--MIMEBoundaryurn_uuid_E244193BA7B96D61AA1305672192822

Content-Type: application/xop+xml; charset=UTF-8; type="application/soap+xml"

Content-Transfer-Encoding: binary

Content-ID: <0.urn:uuid:E244193BA7B96D61AA1305672192823@apache.org>


<soapenv:Reason xmlns:soapenv="http://www.w3.org/2003/05/soap-envelope"><soapenv:Text
xml:lang="en-US">The endpoint reference (EPR) for the Operation not found is /wne2/services/v21 and
the WSA Action = null</soapenv:Text></soapenv:Reason>

--MIMEBoundaryurn_uuid_E244193BA7B96D61AA1305672192822--
```

Figure 5: WNE2 confirmation

The URL for the REST endpoint is http://localhost:<port>/wnrest/services/{Query|NodePing}. By selecting the NodePing version (e.g. http://localhost:<port>/wnrest/services/NodePing) You will see a browser message like follows:

OpenNode2 Java v2.10