MotionWorks Enterprise

2.0

Installation Guide

ZEBRA

P1123394-04EN

Terms of Use

Proprietary Statement

Product Improvements

Liability Disclaimer

Limitation of Liability

Publication Date

January 2024

Contents

Introduction

This document provides instructions for installing MotionWorks Enterprise (MWE) 2.0 software from Zebra Technologies Corporation. It also provides specifications for servers hosting the software.

Zebra Technologies offers world-class real-time asset tracking and management software solutions to optimize the flow of goods in complex logistical operations, increasing productivity, lowering operational costs, and improving safety and security. Zebra Technologies uses a wide range of scalable RTLS (Real Time Locating Systems) technologies to generate accurate, on-demand information about the physical location and status of assets.

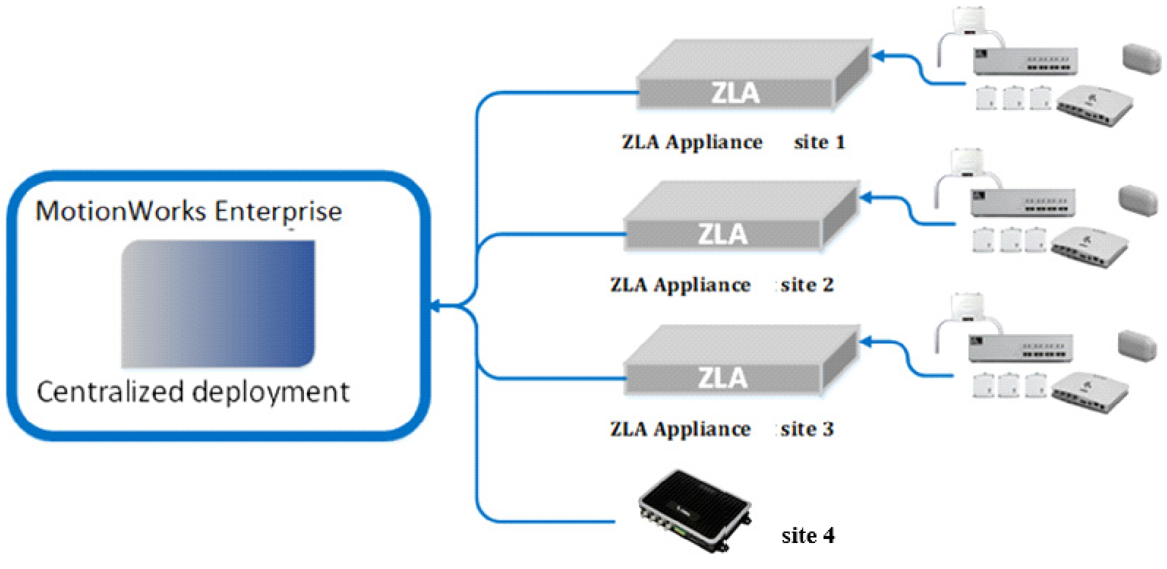
The MWE software suite provides tools for designing, configuring, operating, and troubleshooting RTLS solutions. MWE serves as the central repository for all the real-time location and communication data captured by the RTLS-tracking infrastructure, and offers tools for integrating this data with customer and third-party applications.

Some of the location and telemetry RFID technologies supported by MWE include:

- Passive RFID
- UWB
- Bluetooth Low-Energy
- Wi-Fi
- ISO 24730
- GPS

The following diagram provides an overview of RTLS system components and data flow. Sensors connected to a network, wired or wirelessly, detect over-the-air RF transmissions from tags (RF transmitters), and the generated data flows across the network to a Zebra Location Appliance (ZLA) and then to the MWE server. Some sensors, such as passive RFID readers, send data directly to the MWE server without requiring a ZLA. Multiple sites are supported, each with its own ZLA.

The ZLA can be virtual or physical. A physical ZLA is a 1U rackmount box typically installed on-site. Depending on the RTLS technology deployed, a ZLA can be installed on-premises at a site, or in a remote data center. A ZLA runs location algorithms and feeds tag, blink, and location data to an MWE server. The ZLA software also includes filters for removing redundant data and reducing network traffic. The MWE server stores and retrieves data from a database, and forwards data and events to external systems.



A virtual ZLA can be provided as an OVA file with pre-loaded software ready to deploy, or via a ZLA installer that converts a Linux virtual server into a ZLA appliance. Supported Linux flavors include Red Hat 7.9/8.x, CentOS 7.9, Ubuntu 20.04 LTS, and Ubuntu 22.04 LTS.

MWE 2.0 also supports direct communication from passive RFID readers to MWE running locally or in the cloud. No ZLA appliance is required in this case.

Minimum Server Specs

Location data can flow into the MWE servers from a ZLA device or another source. MWE deployment requires at least one Linux server. A ZLA appliance may or may not be required. Following are the minimum hardware and software requirements for three types of deployments. The user selects one of these deployment types when running the MWE installation script.

Figure 1 Production Deployment - Full Functionality

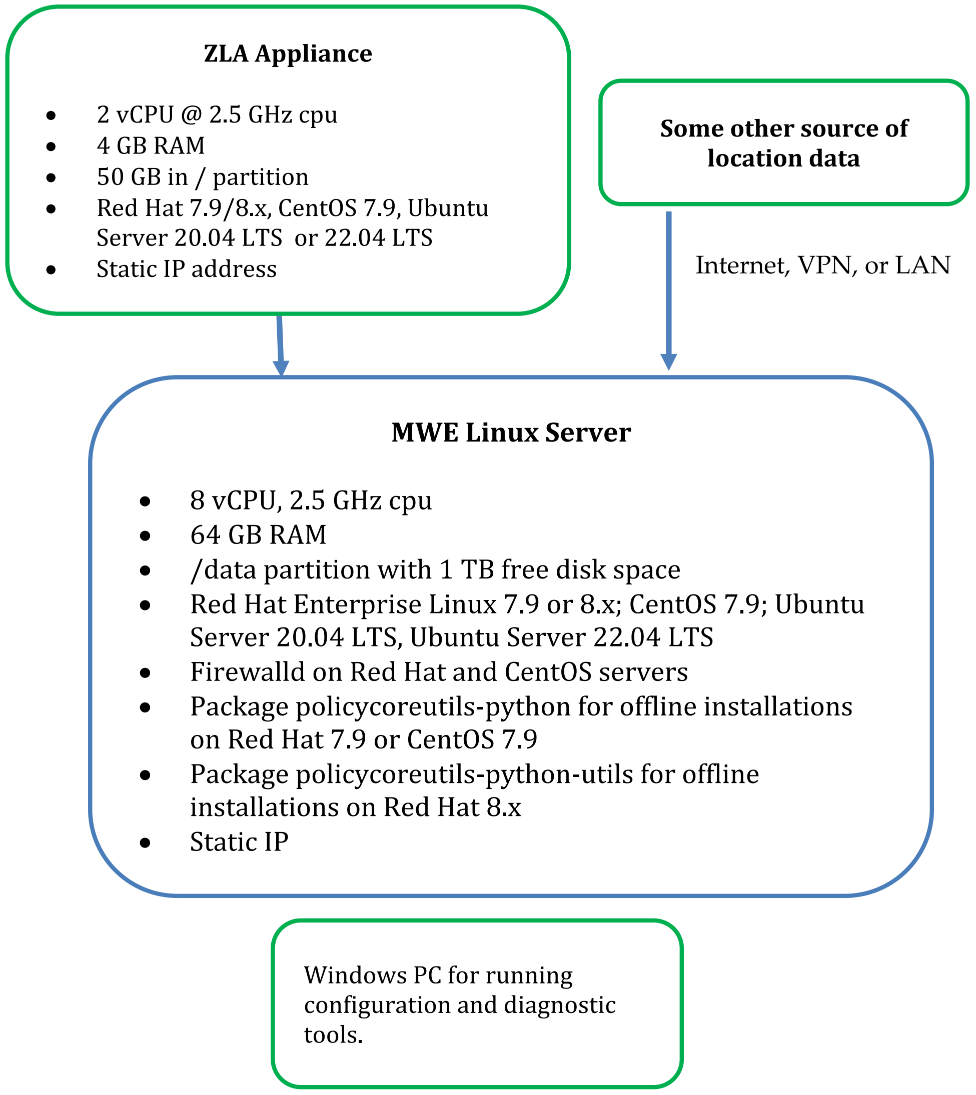
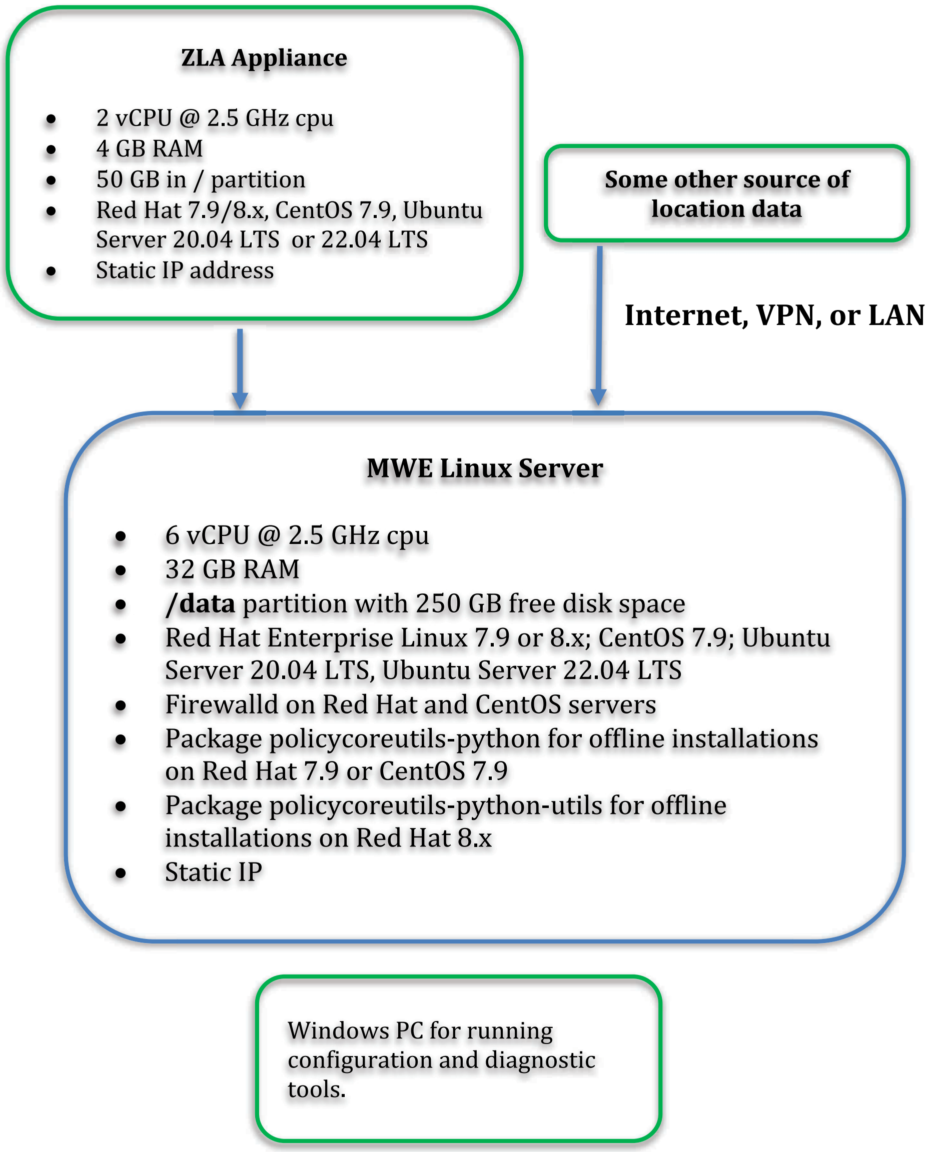
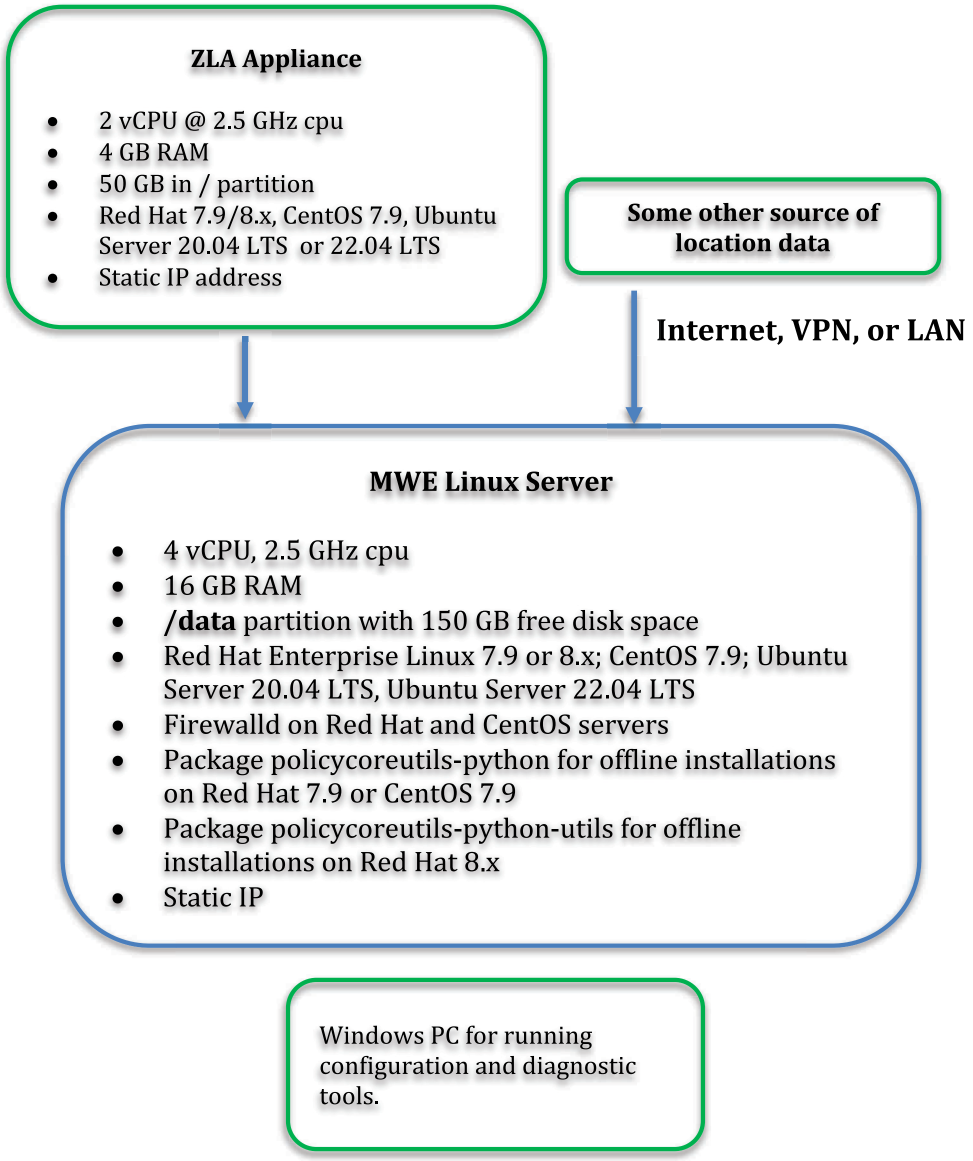

ZLA Appliance
- 2 vCPU @ 2.5 GHz cpu
- 4 GB RAM
- 50 GB in / partition
- Red Hat 7.9/8.x, CentOS 7.9, Ubuntu Server 20.04 LTS or 22.04 LTS
- Static IP address
Some other source of location data
Internet, VPN, or LAN
MWE Linux Server
- 8 vCPU, 2.5 GHz cpu
- 64 GB RAM
- /data partition with 1 TB free disk space
- Red Hat Enterprise Linux 7.9 or 8.x; CentOS 7.9; Ubuntu Server 20.04 LTS, Ubuntu Server 22.04 LTS
- Firewalld on Red Hat and CentOS servers
- Package policycoreutils-python for offline installations on Red Hat 7.9 or CentOS 7.9
- Package policycoreutils-python-utils for offline installations on Red Hat 8.x
- Static IP
Windows PC for running configuration and diagnostic tools.

Figure 2 Demo Deployment - Full Functionality


ZLA Appliance
2 vCPU @ 2.5 GHz cpu
4 GB RAM
50 GB in / partition
Red Hat 7.9/8.x, CentOS 7.9, Ubuntu Server 20.04 LTS or 22.04 LTS
Static IP address
Some other source of location data
Internet, VPN, or LAN
MWE Linux Server
6 vCPU @ 2.5 GHz cpu
32 GB RAM
/data partition with 250 GB free disk space
Red Hat Enterprise Linux 7.9 or 8.x; CentOS 7.9; Ubuntu Server 20.04 LTS, Ubuntu Server 22.04 LTS
Firewalld on Red Hat and CentOS servers
Package policycoreutils-python for offline installations on Red Hat 7.9 or CentOS 7.9
Package policycoreutils-python-utils for offline installations on Red Hat 8.x
Static IP
Windows PC for running configuration and diagnostic tools.

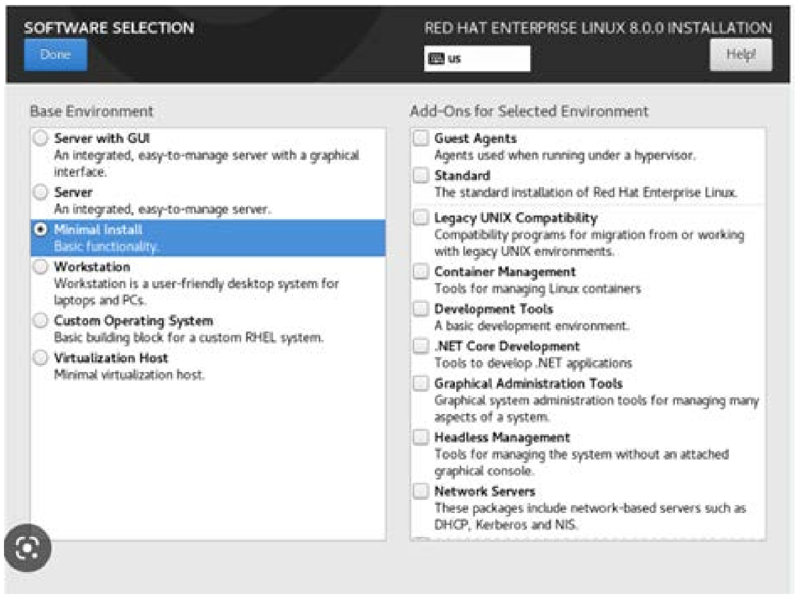
In demo deployments with minimal load the following functionality is turned off: historical service logs, Kibana tool, and MWE monitoring of system health. Historical application data, such as tag blink and event history, is still available.

Figure 3 Minimal Demo Deployment - Most Functionality


ZLA Appliance
2 vCPU @ 2.5 GHz cpu
4 GB RAM
50 GB in / partition
Red Hat 7.9/8.x, CentOS 7.9, Ubuntu Server 20.04 LTS or 22.04 LTS
Static IP address
Some other source of location data
Internet, VPN, or LAN
MWE Linux Server
4 vCPU, 2.5 GHz cpu
16 GB RAM
/data partition with 150 GB free disk space
Red Hat Enterprise Linux 7.9 or 8.x; CentOS 7.9; Ubuntu Server 20.04 LTS, Ubuntu Server 22.04 LTS
Firewalld on Red Hat and CentOS servers
Package policycoreutils-python for offline installations on Red Hat 7.9 or CentOS 7.9
Package policycoreutils-python-utils for offline installations on Red Hat 8.x
Static IP
Windows PC for running configuration and diagnostic tools.

Notes on Linux Server Requirements

- Installation of MWE 2.0 on a Linux server requires a **/data** partition located directly under the root **/** directory.
- For Red Hat or CentOS servers hosting MWE 2.0:
    - The **Minimal Install** option for the operating system is sufficient.



SOFTWARE SELECTION

RED HAT ENTERPRISE LINUX 8.0.0 INSTALLATION

Base Environment

- Server with GUI
  An integrated, easy-to-manage server with a graphical interface.
- Server
  An integrated, easy-to-manage server.
- [✓ Minimal Install]
  Basic functionality.
- Workstation
  Workstation is a user-friendly desktop system for laptops and PCs.
- Custom Operating System
  Basic building block for a custom RHEL system.
- Virtualization Host
  Minimal virtualization host.

Add-Ons for Selected Environment

- Guest Agents
  Agents used when running under a hypervisor.
- Standard
  The standard installation of Red Hat Enterprise Linux.
- Legacy UNIX Compatibility
  Compatibility programs for migration from or working with legacy UNIX environments.
- Container Management
  Tools for managing Linux containers
- Development Tools
  A basic development environment.
- .NET Core Development
  Tools to develop .NET applications
- Graphical Administration Tools
  Graphical system administration tools for managing many aspects of a system.
- Headless Management
  Tools for managing the system without an attached graphical console.
- Network Servers
  These packages include network-based servers such as DHCP, Kerberos and NIS.

    - Install and run the **firewalld** daemon on the server before installing the MWE software.
    - Package **policycoreutils-python** must be installed on Red Hat 7.9 or CentOS 7.9.
      Package **policycoreutils-python-utils** must be installed on Red Hat 8.x.

      To verify this, run the command:
      # rpm -q policycoreutils-python or **# rpm -q policycoreutils-python-utils.**

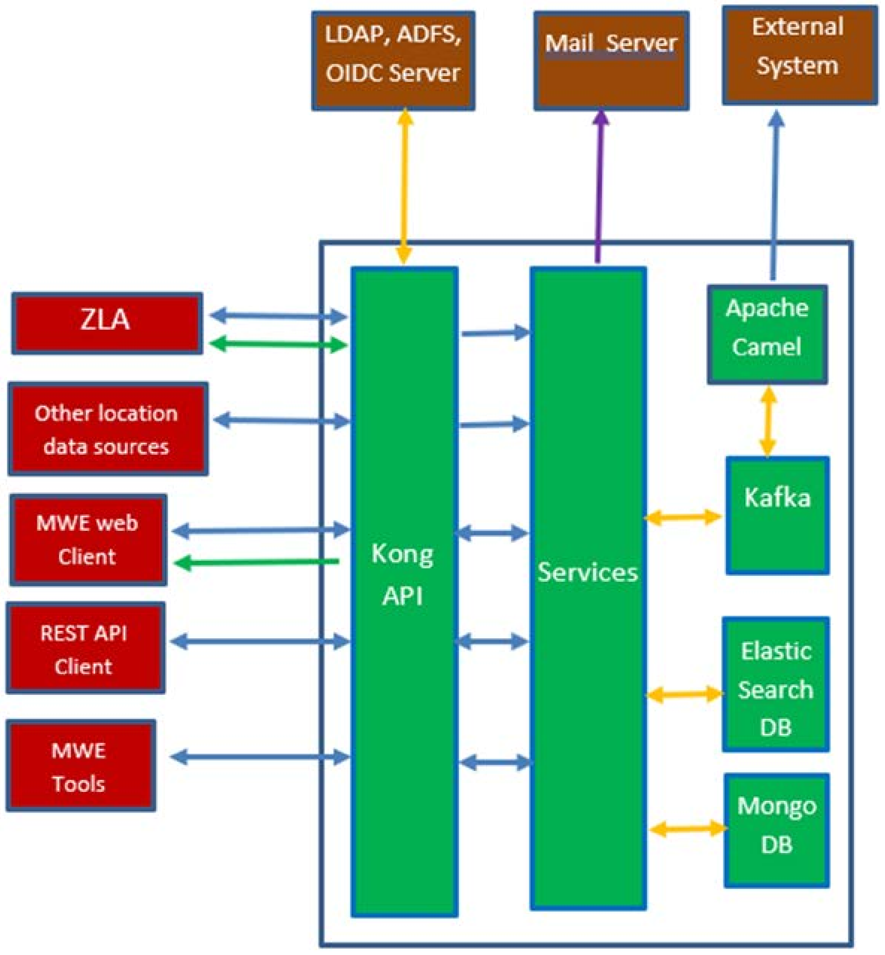
      If the server has connection to a local or public yum repository, to install this package, run the command:
      # yum install policycoreutils-python or **# yum install policycoreutils-python-utils**
- For an Ubuntu server hosting MWE or ZLA software, the Ubuntu Server edition is required.

The core MWE services on the Linux server are installed as Docker containers running on a local Docker subnet created by the installation script. This adds the following two requirements:

- ipv4 forwarding must be enabled on the server.
- If a security agent is installed on the Linux server, it must allow all traffic within the Docker subnet and between the Docker subnet and the local host.

MWE Software Components

The following reference diagram illustrates the various MWE software components hosted on the MWE Linux server and their relationship within the MWE software:



MWE Linux Server

NOTES: The arrows in the figure indicate the following:

Blue arrows	http/https connections
Orange arrows	TCP connections
Green arrow	WebSocket connection
Purple arrow	SMTP connection

- MWE Tools include System Builder and other tools.
- All the core MWE services run on the Linux server as Docker containers.

Installation Files

Obtain a link to download the MWE installation files from Zebra detailed below. The ‘n’ in the filenames denotes the latest 2.0 release version available. When present, the 'm' in the filenames denotes the installer version. A new installer may occasionally be available to consolidate hotfixes and avoid having to apply them manually after installation.

mwe-containers-setup-2.0.n.m-offline.tar.gz	Installs/upgrades MWE core services on a Linux server. The installer is self-contained and no Internet connection is required.
mwe-containers-setup-2.0.n.m-online.tar.gz	Installs/upgrades MWE core services on a Linux server. This installer is much smaller than its offline counterpart but requires the Linux server to have Internet access at installation time.
mwe_tools_2.0.n.exe	Installs/upgrades a set of MWE configuration and diagnostic tools on any Windows PC. These tools can connect to the MWE (Linux) server or to different types of location sensors to perform configuration or diagnostics tasks. See Ports to determine what network ports may need to be opened.
ZLA-2.0.n-m.i386.rpm	Upgrades the software on a ZLA appliance (physical or virtual). A ZLA is provided with pre-loaded ZLA software. The rpm package ZLA-2.0.n-m.i386.rpm upgrades the pre-loaded software to version 2.0.n-m. Note: On request, Zebra can provide an installation script that converts a CentOS or Red Hat server into a ZLA appliance. The script installs a previous version of ZLA software, so the ZLA-2.0.n-m.i386.rpm package must be applied to upgrade the ZLA software to version 2.0.n-m.

Install MWE

This section describes off-line MWE installation for Linux servers with no access to the Internet, and optional on-line installation for servers with Internet access.

Off-Line Installation

Off-line installation does not require the Linux server to have Internet access. Installation involves loading an 8.5 GB tar.gz file to the Linux server, so copying it prior to a scheduled installation is recommended. Before installation, verify the MWE Linux Server requirements in Minimum Server Specs on page 6 are met.

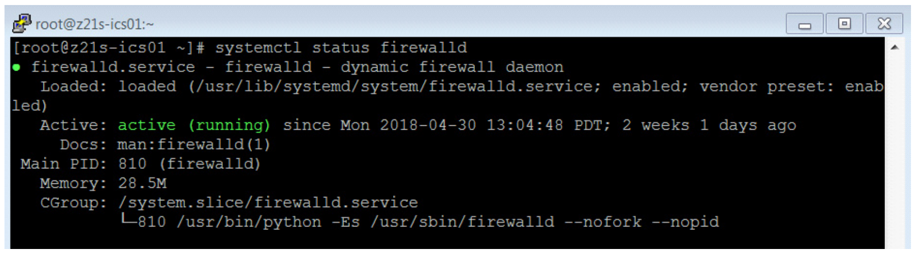
Initial Checks

Verify that the following requirements are fulfilled:

- A **/data** partition exists on the Linux server. See details in Minimum Server Specs on page 6.
- The package **policycoreutils-python** is installed on Red Hat 7.9 or CentOS 7.9. The package **policycoreutils-python-utils** is installed on Red Hat 8.x. See Notes on Linux Server Requirements on page 9 to check for and install this package.
- The **firewalld** daemon is installed and running on a Red Har or CentOS server. Enter the following command:

  ```
  # systemctl status firewalld
  ```

  Verify the status as **active (running)**, as in the following screen.



```
[root@z21s-ics01 ~]# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset: enab
led)
   Active: active (running) since Mon 2018-04-30 13:04:48 PDT; 2 weeks 1 days ago
     Docs: man:firewalld(1)
 Main PID: 810 (firewalld)
   Memory: 28.5M
   CGroup: /system.slice/firewalld.service
           └─810 /usr/bin/python -Es /usr/sbin/firewalld --nofork --nopid
```

Copying and Extracting the Installation Package

1. Using WinSCP, Putty, or a similar SSH client, copy the **mwe-containers-setup-2.0.n.m-offline.tar.gz** installation package to the **/data** directory on the Linux server.
2. Using a Terminal window, Putty, or a similar SSH client, log into the MWE server using the **root** account.
3. Change directory to **/data** and extract the **tar.gz** file:

   ```
   # cd /data
   # tar -xvf  mwe-containers-setup-2.0.n.m-offline.tar.gz
   ```

NOTE: Occasionally characters are misinterpreted when copying a command from a document and pasting it into a Putty or Terminal window. Some commands may require manual entry.

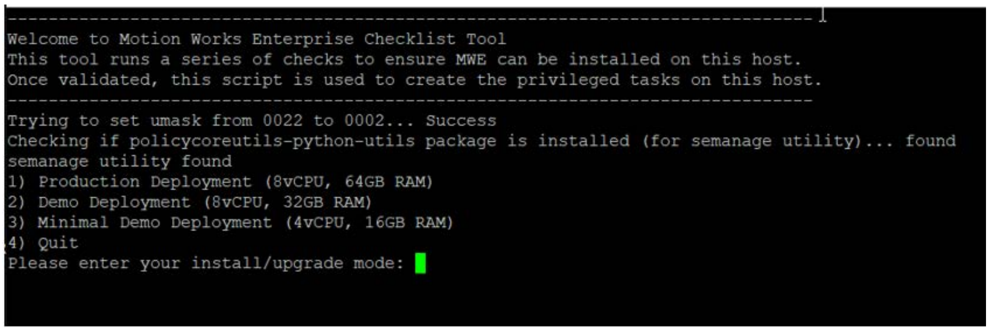
Performing Environment Validation and Initial Setup

The following setup command verifies that the environment meets the requirements, installs the Docker service, and creates an **mwe** user account to be used by the MWE application.

1. Extract and run the following commands.

```
# cd /data/mwe_setup
# ./mwe_setup.sh -offline-setup
```

2. Select one of four installation options.



```
--------------------------------------------------------------------------------
Welcome to Motion Works Enterprise Checklist Tool
This tool runs a series of checks to ensure MWE can be installed on this host.
Once validated, this script is used to create the privileged tasks on this host.
--------------------------------------------------------------------------------
Trying to set umask from 0022 to 0002... Success
Checking if policycoreutils-python-utils package is installed (for semanage utility)... found
semanage utility found
1) Production Deployment (8vCPU, 64GB RAM)
2) Demo Deployment (8vCPU, 32GB RAM)
3) Minimal Demo Deployment (4vCPU, 16GB RAM)
4) Quit
Please enter your install/upgrade mode:
```

- **Production Deployment** installs full functionality and is for production deployments.
  The installation script checks the following:
  - Red Hat 7.9/8.x or CentOS 7.9 or Ubuntu 20.04 or Ubuntu 22.04 is installed
  - There is at least 64 GB of memory installed
  - There is at least 1000 GB (1 TB) of disk space available under **`/data`**
  - firewalld is running on a Red Hat or CentOS server

  The installation script exits if one of the conditions above is not met.
- **Demo Deployment** includes full functionality, but is meant for demonstrations or proof of concepts with a light load. The installation script checks the following:
  - Red Hat 7.9/8.x or CentOS 7.9 or Ubuntu 20.04 or Ubuntu 22.04 is installed
  - There is at least 32 GB of memory installed
  - There is at least 250 GB of disk space available under **`/data`**
  - firewalld is running on a Red Hat or CentOS server

  The installation script exits if one of the conditions above is not met. The display recommends 6 vCPU's, but the setup script does not enforce it. You can use 4 vCPU for a demo deployment with a light load if occasional response time delays are acceptable.
- **Minimal Demo Deployment** includes most functionality and is for deployments with minimal load. The following functionality is turned off: Camel, Resource Alerts, historical service logs, Kibana tool, and WherePort Health alerts. Historical application data, such as tag blink and event history, is still available. It is possible to enable/disable services after installation; contact Product Support.

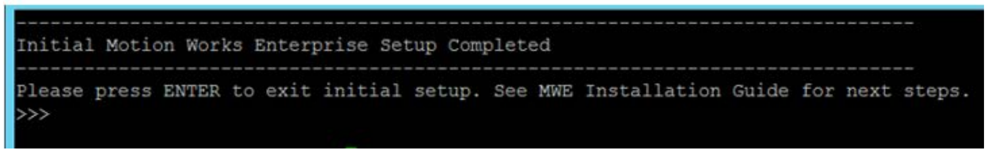
  The installation script checks the following:
  - Red Hat 7.9/8.x or CentOS 7.9 or Ubuntu 20.04 or Ubuntu 22.04 is installed
  - There is at least 16 GB of memory installed
  - There is at least 150 GB of disk space available under **`/data`**

  - firewalld is running on a Red Hat or CentOS server

  The installation script exits if one of the conditions above is not met.

- **Quit**

If all of the checks are successful, the following screen appears:



```
-------------------------------------------------------------------------------
Initial Motion Works Enterprise Setup Completed
-------------------------------------------------------------------------------
Please press ENTER to exit initial setup. See MWE Installation Guide for next steps.
>>>
```

3. Press the Enter key to exit the setup and continue with MWE software installation.

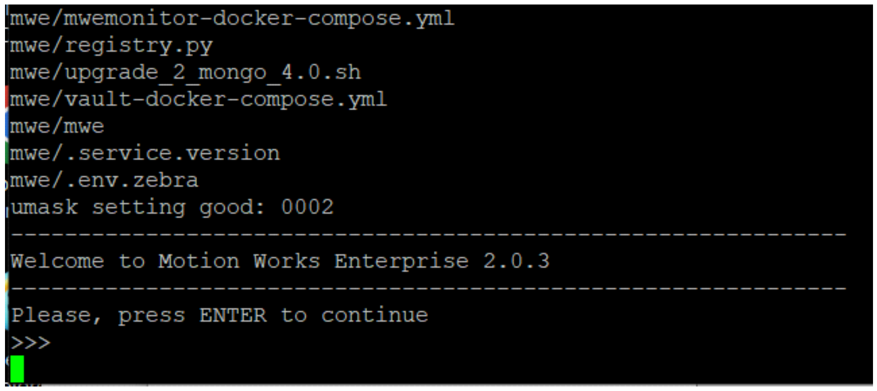
Installing the MWE Software

1. Using a Terminal window, Putty, or another SSH client, log into the MWE server using the root account.
2. Change directory and run the installation script:

```
# cd /data/mwe_setup
# ./mwe_setup.sh --offline-install
```

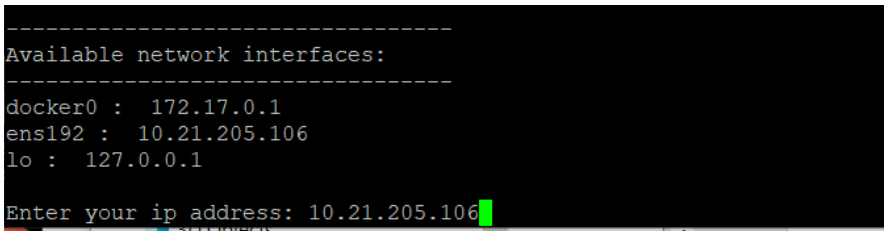
   The installer extract files and docker images from the installation files. This may take several minutes.

3. When prompted, press **ENTER**.



```
mwe/mwemonitor-docker-compose.yml
mwe/registry.py
mwe/upgrade_2_mongo_4.0.sh
mwe/vault-docker-compose.yml
mwe/mwe
mwe/.service.version
mwe/.env.zebra
umask setting good: 0002
-------------------------------------------------------------
Welcome to Motion Works Enterprise 2.0.3
-------------------------------------------------------------
Please, press ENTER to continue
>>>
```

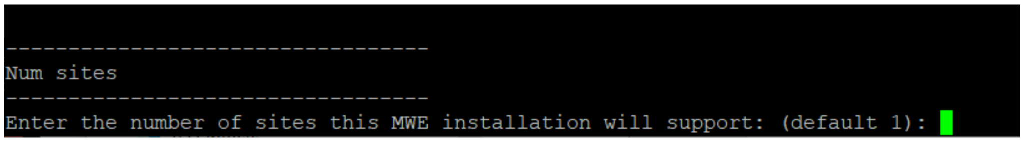
4. Enter the IP address of the Linux server where this installation is being done. It is one of the network interfaces listed by the installation script, as in the following screen.



```
----------------------------------
Available network interfaces:
----------------------------------
docker0 :  172.17.0.1
ens192 :  10.21.205.106
lo :  127.0.0.1

Enter your ip address: 10.21.205.106
```

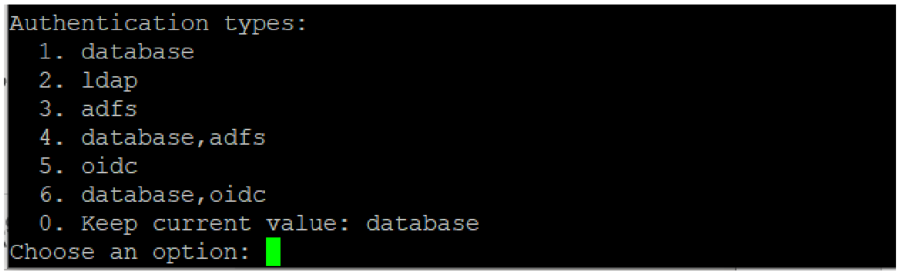
5. Enter the number of sites this MWE installation will support.

   If you are not sure, enter **1**. MWE automatically adjusts this value as needed.



```
---------------------------------
Num sites
---------------------------------
Enter the number of sites this MWE installation will support: (default 1):
```

6. Select an authentication type. At installation, you can select option `1. database`. After installation, you can choose and configure a different authentication type, as explained in the MWE 2.0 Configuration Guide.



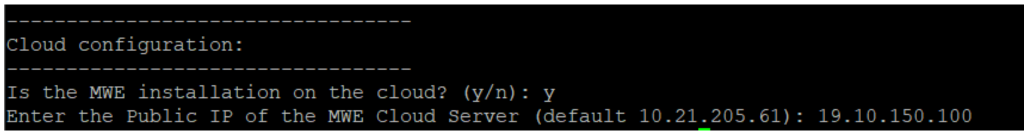
```
Authentication types:
  1. database
  2. ldap
  3. adfs
  4. database,adfs
  5. oidc
  6. database,oidc
  0. Keep current value: database
Choose an option:
```

7. Respond to **`Is the MWE installation on the cloud? (y/n)`** based on the MWE deployment:

   - If the MWE server is hosted in a private network, enter 'n'.
     Specifically, the MWE server and any reader/location sensors sending data to MWE are all on the same network, or on different subnets linked by a router. In this case the MWE server and the readers/sensors can reach each other's IP addresses and traffic between them can flow unimpeded (after opening necessary ports. See Ports).

   - If the MWE server is hosted in a cloud and has a public IP address, enter 'y'.
     Specifically, the MWE server is in a cloud behind the cloud firewall while the readers/sensors are on-premises at a site behind a local firewall.

     If you answer yes **`(y)`** you are prompted to enter the public IP address of the MWE server in the cloud.

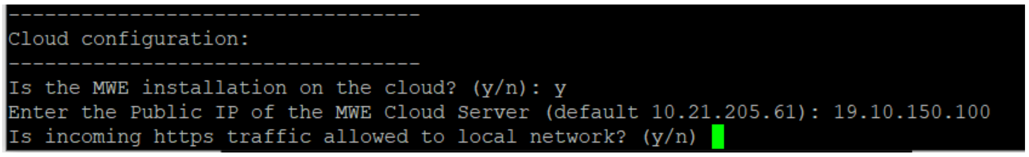


```
---------------------------------
Cloud configuration:
---------------------------------
Is the MWE installation on the cloud? (y/n): y
Enter the Public IP of the MWE Cloud Server (default 10.21.205.61): 19.10.150.100
```

8. When prompted **Is incoming https traffic allowed to local network? (y/n)** the answer is related to connections initiated by the MWE server in the cloud to the readers/sensors in the local network.

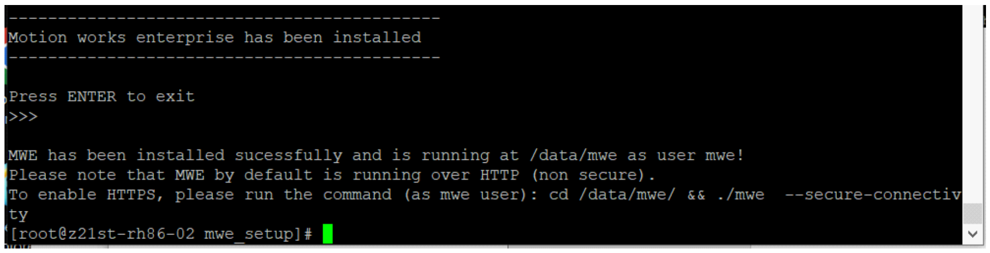
   - The MWE server in the cloud cannot directly reach the RFID readers/sensors IP addresses in the local network because the reader IP is private and sits behind the site’s firewall. There is no public IP for the reader/sensor. Therefore, HTTPS traffic from the MWE server in the cloud to a local reader is not allowed. In this case, answer no **`(n)`**.

- The local RFID readers/sensors have a public Internet address, or otherwise the MWE server is in a private cloud that allows the MWE server to initiate a connection to the reader IP address. In this case answer yes `(y)`.



```
-----------------------------------
Cloud configuration:
-----------------------------------
Is the MWE installation on the cloud? (y/n): y
Enter the Public IP of the MWE Cloud Server (default 10.21.205.61): 19.10.150.100
Is incoming https traffic allowed to local network? (y/n)
```

The installation script installs the Docker images, which typically takes several minutes. When installation completes, press **Enter** to exit the installation.



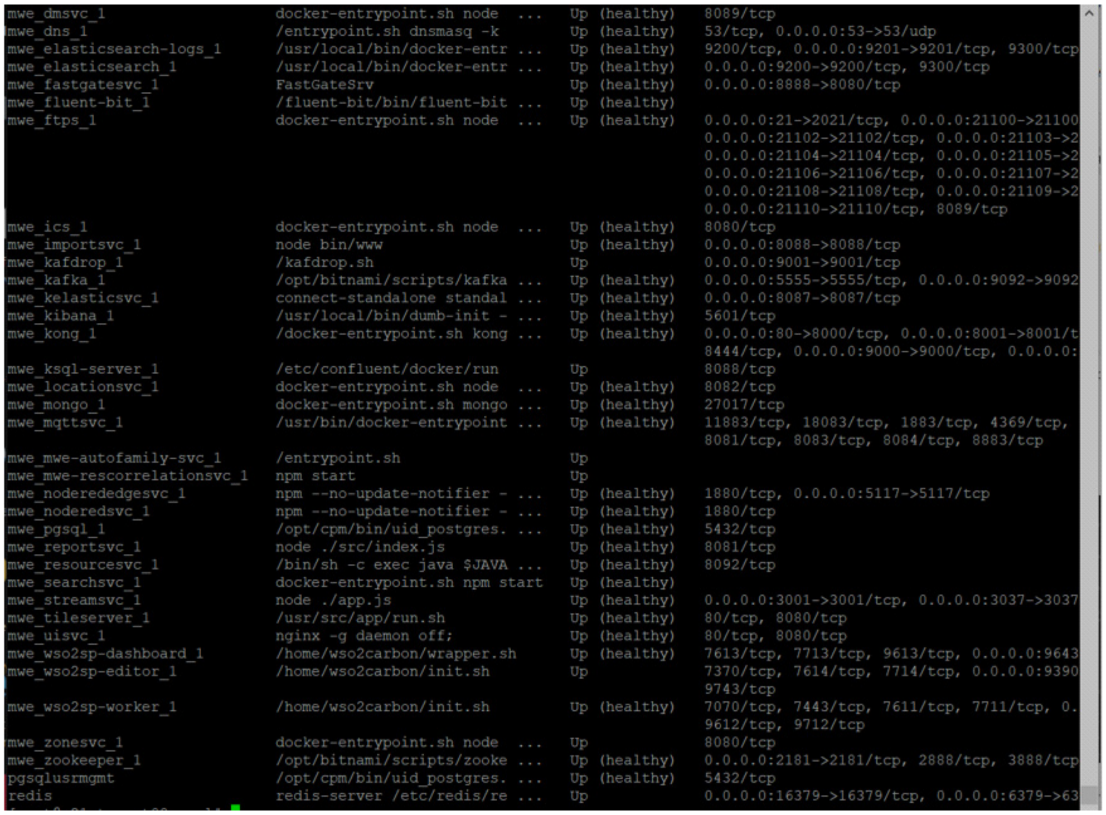
```
--------------------------------------------
Motion works enterprise has been installed
--------------------------------------------

Press ENTER to exit
>>>

MWE has been installed sucessfully and is running at /data/mwe as user mwe!
Please note that MWE by default is running over HTTP (non secure).
To enable HTTPS, please run the command (as mwe user): cd /data/mwe/ && ./mwe  --secure-connectiv
ty
[root@z21st-rh86-02 mwe_setup]#
```

Checking Status

Verify that all MWE services are up and running. Depending on the resources on your server (CPU, memory), this may take a few minutes. MWE services run as Docker containers which can be stopped and started. Use the **docker** and **docker-compose** commands to check the status of the containers.

1. To check the status of the MWE services, run the following commands:

```
# cd /data/mwe
# docker-compose ps
```



```
mwe_dmsvc_1                   docker-entrypoint.sh node   ...   Up (healthy)   8089/tcp
mwe_dns_1                     /entrypoint.sh dnsmasq -k         Up (healthy)   53/tcp, 0.0.0.0:53->53/udp
mwe_elasticsearch-logs_1      /usr/local/bin/docker-entr ...   Up (healthy)   9200/tcp, 0.0.0.0:9201->9201/tcp, 9300/tcp
mwe_elasticsearch_1           /usr/local/bin/docker-entr ...   Up (healthy)   0.0.0.0:9200->9200/tcp, 9300/tcp
mwe_fastgatesvc_1             FastGateSrv                       Up (healthy)   0.0.0.0:8888->8080/tcp
mwe_fluent-bit_1              /fluent-bit/bin/fluent-bit ...   Up (healthy)
mwe_ftps_1                    docker-entrypoint.sh node   ...   Up (healthy)   0.0.0.0:21->2021/tcp, 0.0.0.0:21100->21100
                                                                               0.0.0.0:21102->21102/tcp, 0.0.0.0:21103->2
                                                                               0.0.0.0:21104->21104/tcp, 0.0.0.0:21105->2
                                                                               0.0.0.0:21106->21106/tcp, 0.0.0.0:21107->2
                                                                               0.0.0.0:21108->21108/tcp, 0.0.0.0:21109->2
                                                                               0.0.0.0:21110->21110/tcp, 8089/tcp
mwe_ics_1                     docker-entrypoint.sh node   ...   Up (healthy)   8080/tcp
mwe_importsvc_1               node bin/www                      Up (healthy)   0.0.0.0:8088->8088/tcp
mwe_kafdrop_1                 /kafdrop.sh                       Up             0.0.0.0:9001->9001/tcp
mwe_kafka_1                   /opt/bitnami/scripts/kafka ...   Up (healthy)   0.0.0.0:5555->5555/tcp, 0.0.0.0:9092->9092
mwe_kelasticsvc_1             connect-standalone standal ...   Up (healthy)   0.0.0.0:8087->8087/tcp
mwe_kibana_1                  /usr/local/bin/dumb-init - ...   Up (healthy)   5601/tcp
mwe_kong_1                    /docker-entrypoint.sh kong ...   Up (healthy)   0.0.0.0:80->8000/tcp, 0.0.0.0:8001->8001/t
                                                                               8444/tcp, 0.0.0.0:9000->9000/tcp, 0.0.0.0:
mwe_ksql-server_1             /etc/confluent/docker/run         Up             8088/tcp
mwe_locationsvc_1             docker-entrypoint.sh node   ...   Up (healthy)   8082/tcp
mwe_mongo_1                   docker-entrypoint.sh mongo ...   Up (healthy)   27017/tcp
mwe_mqttsvc_1                 /usr/bin/docker-entrypoint ...   Up (healthy)   11883/tcp, 18083/tcp, 1883/tcp, 4369/tcp,
                                                                               8081/tcp, 8083/tcp, 8084/tcp, 8883/tcp
mwe_mwe-autofamily-svc_1      /entrypoint.sh                    Up
mwe_mwe-rescorrelationsvc_1   npm start                         Up
mwe_noderededgesvc_1          npm --no-update-notifier - ...   Up (healthy)   1880/tcp, 0.0.0.0:5117->5117/tcp
mwe_noderedsvc_1              npm --no-update-notifier - ...   Up (healthy)   1880/tcp
mwe_pgsql_1                   /opt/cpm/bin/uid_postgres. ...   Up (healthy)   5432/tcp
mwe_reportsvc_1               node ./src/index.js               Up (healthy)   8081/tcp
mwe_resourcesvc_1             /bin/sh -c exec java $JAVA ...   Up (healthy)   8092/tcp
mwe_searchsvc_1               docker-entrypoint.sh npm start    Up (healthy)
mwe_streamsvc_1               node ./app.js                     Up (healthy)   0.0.0.0:3001->3001/tcp, 0.0.0.0:3037->3037
mwe_tileserver_1              /usr/src/app/run.sh               Up (healthy)   80/tcp, 8080/tcp
mwe_uisvc_1                   nginx -g daemon off;              Up (healthy)   80/tcp, 8080/tcp
mwe_wso2sp-dashboard_1        /home/wso2carbon/wrapper.sh       Up (healthy)   7613/tcp, 7713/tcp, 9613/tcp, 0.0.0.0:9643
mwe_wso2sp-editor_1           /home/wso2carbon/init.sh          Up             7370/tcp, 7614/tcp, 7714/tcp, 0.0.0.0:9390
                                                                               9743/tcp
mwe_wso2sp-worker_1           /home/wso2carbon/init.sh          Up (healthy)   7070/tcp, 7443/tcp, 7611/tcp, 7711/tcp, 0.
                                                                               9612/tcp, 9712/tcp
mwe_zonesvc_1                 docker-entrypoint.sh node   ...   Up             8080/tcp
mwe_zookeeper_1               /opt/bitnami/scripts/zooke ...   Up (healthy)   0.0.0.0:2181->2181/tcp, 2888/tcp, 3888/tcp
pgsqlusrmgmt                  /opt/cpm/bin/uid_postgres. ...   Up (healthy)   5432/tcp
redis                         redis-server /etc/redis/re ...   Up             0.0.0.0:16379->16379/tcp, 0.0.0.0:6379->63
```

2. Verify that the column **State** shows **Up** for all containers. Some services also report a health condition and should show **`healthy`**.

NOTE: You may see several warnings if you run the docker-compose command under the **root** account. Ignore these warnings. If you switch to the **mwe** account, the warnings do not appear.

`# su - mwe` (if prompted, the default password is Zebra123)

```
# cd /data/mwe
# docker-compose ps
```

To switch back to the root account use:

```
# su -
```

To restart all MWE services for any reason, either reboot the server or use these commands:

`# su - mwe` (if prompted, the default password is Zebra123)

```
# cd /data/mwe
# ./mwe --restart
```

MWE Files Location

The MWE files are copied to the following directories on the Linux server:

Installation package:	`/data/`
Setup files:	`/data/mwe_setup/`
MWE commands and configuration (.env):	`/data/mwe/`
MWE Services configuration:	`/data/mwe-conf/`
Docker images and containers:	`/data/docker/`
3rd Party services and database backups:	`/data/zebra`
Log files:	`/data/mwe-logs`

On-Line Installation

The following installation procedure assumes that, during installation, the MWE Linux server has access to the websites listed below. If this is not the case, see Off-Line Installation. Before proceeding with the installation, verify that the requirements listed in Minimum Server Specs for the MWE Linux Server are met.

Performing Initial Checks

Before installing, verify the following requirements are met:

- A **/data** partition exists on the Linux server. See details in Minimum Server Specs.
- The MWE Linux server has access to the following websites:

  https://registry.zebramwe.com

  https://yumrepo.zebramwe.com

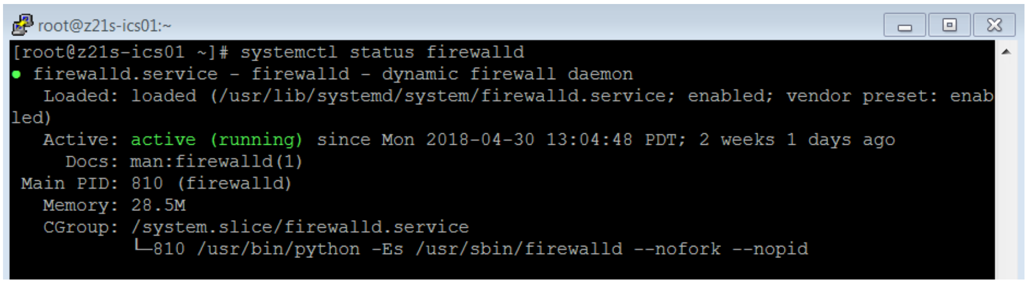
  To check access, use the curl command, for example:

  ```
  # curl -k https://registry.zebramwe.com
  ```

  returns quickly with no error if the server has access to https://registry.zebramwe.com. Otherwise a connection timeout or similar error message occurs.
- The firewalld daemon is installed and running on a Red Hat or CentOS server. Use the following command to verify:

  ```
  # systemctl status firewalld
  ```

  The result shows the status as 'active (running)'.



```
[root@z21s-ics01 ~]# systemctl status firewalld
● firewalld.service - firewalld - dynamic firewall daemon
   Loaded: loaded (/usr/lib/systemd/system/firewalld.service; enabled; vendor preset: enab
led)
   Active: active (running) since Mon 2018-04-30 13:04:48 PDT; 2 weeks 1 days ago
     Docs: man:firewalld(1)
 Main PID: 810 (firewalld)
   Memory: 28.5M
   CGroup: /system.slice/firewalld.service
           └─810 /usr/bin/python -Es /usr/sbin/firewalld --nofork --nopid
```

Copying and Extracting Installation Package

1. Using WinSCP, Putty, or a similar SSH client, copy the **mwe-containers-setup-2.0.n.m-online.tar.gz** installation package to the **/data** directory on the MWE Linux server.
2. Using a Terminal window, Putty, or a similar SSH client, log into the MWE server using the root account.
3. Change the directory to **/data** and extract the **tar.gz** file:

   ```
   # cd /data
   # tar -xvf mwe-containers-setup-2.0.n.m-online.tar.gz
   ```

NOTE: Occasionally characters are misinterpreted when copying a command from a document and pasting it into a Putty or Terminal window. Some commands may require manual entry.

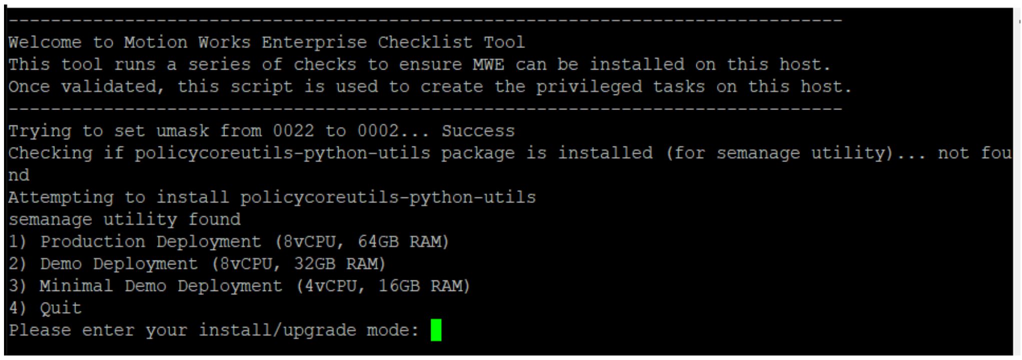
Performing Environment Validation and Initial Setup

1. Run the following command to verify the environment meets the requirements.

   If the requirements are met, the command installs the Docker service and created an **mwe** account to be used by the MWE application.

   ```
   # cd /data/mwe_setup
   # ./mwe_setup.sh --setup
   ```

2. Select one of four installation options.



```
---------------------------------------------------------------------------
Welcome to Motion Works Enterprise Checklist Tool
This tool runs a series of checks to ensure MWE can be installed on this host.
Once validated, this script is used to create the privileged tasks on this host.
---------------------------------------------------------------------------
Trying to set umask from 0022 to 0002... Success
Checking if policycoreutils-python-utils package is installed (for semanage utility)... not fou
nd
Attempting to install policycoreutils-python-utils
semanage utility found
1) Production Deployment (8vCPU, 64GB RAM)
2) Demo Deployment (8vCPU, 32GB RAM)
3) Minimal Demo Deployment (4vCPU, 16GB RAM)
4) Quit
Please enter your install/upgrade mode:
```

- **Production Deployment** installs full functionality and is for production deployments.
  The installation script checks the following:
  - Red Hat 7.9/8.x or CentOS 7.9 or Ubuntu 20.04 or Ubuntu 22.04 is installed
  - There is at least 64 GB of memory installed
  - There is at least 1000 GB (1 TB) of disk space available under **`/data`**
  - firewalld is running on a Red Hat or CentOS server
  - Internet connectivity
  - Connectivity to MWE Yum repo (https://yumrepo.zebramwe.com)
  - Connectivity to MWE registry (https://registry.zebramwe.com)

  The installation script exits if one of the conditions above is not met.
- **Demo Deployment** includes full functionality, but is meant for demonstrations or proof of concepts with a light load. The installation script checks the following:
  - Red Hat 7.9/8.x or CentOS 7.9 or Ubuntu 20.04 or Ubuntu 22.04 is installed
  - There is at least 32 GB of memory installed
  - There is at least 250 GB of disk space available under **`/data`**
  - firewalld is running on a Red Hat or CentOS server
  - Internet connectivity
  - Connectivity to MWE Yum repo (https://yumrepo.zebramwe.com)
  - Connectivity to MWE registry (https://registry.zebramwe.com)

  The installation script exits if one of the conditions above is not met. The display recommends 6 vCPU's, but the setup script does not enforce it. You can use 4 vCPU for a demo deployment with a light load if occasional response time delays are acceptable.

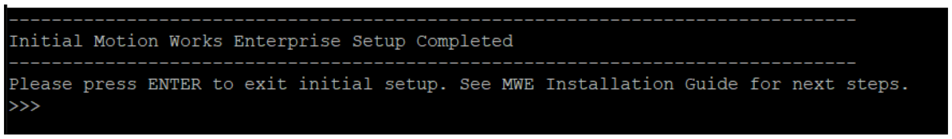
- **Minimal Demo Deployment** includes most functionality and is for deployments with minimal load. The following functionality is turned off: Camel, Resource Alerts, historical service logs, Kibana tool, and WherePort Health alerts. Historical application data, such as tag blink and event history, is still available. It is possible to enable/disable services after installation; contact Product Support.

  The installation script checks the following:

  - Red Hat 7.9/8.x or CentOS 7.9 or Ubuntu 20.04 or Ubuntu 22.04 is installed
  - There is at least 16 GB of memory installed
  - There is at least 150 GB of disk space available under **`/data`**
  - firewalld is running on a Red Hat or CentOS server
  - Internet connectivity
  - Connectivity to MWE Yum repo (https://yumrepo.zebramwe.com)
  - Connectivity to MWE registry (https://registry.zebramwe.com)

  The installation script exits if one of the conditions above is not met.
- **Quit**

If all of the checks are successful, the following screen appears:



```
-------------------------------------------------------------------------------
Initial Motion Works Enterprise Setup Completed
-------------------------------------------------------------------------------
Please press ENTER to exit initial setup. See MWE Installation Guide for next steps.
>>>
```

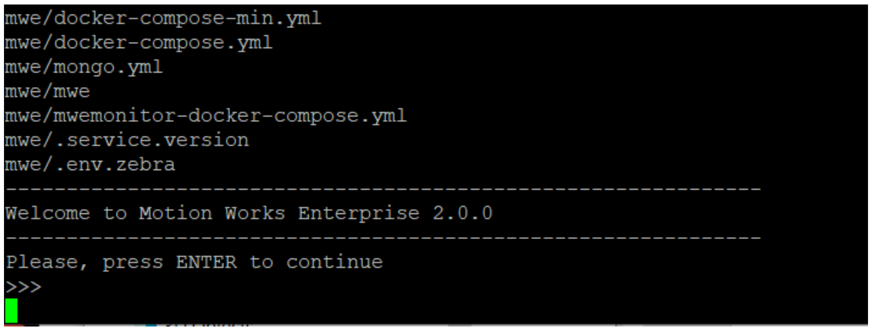
3. Press the Enter key to exit the setup and continue with MWE software installation.

Installing the MWE Software

1. Using a Terminal window, Putty, or another SSH client, log into the MWE server using the root account.
2. Change directory and run the installation script:

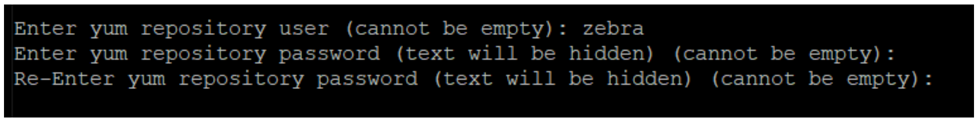
```
# cd /data/mwe_setup
# ./mwe_setup.sh --install
```

The installer extracts and copies a set of files and prompts you to press Enter to continue.



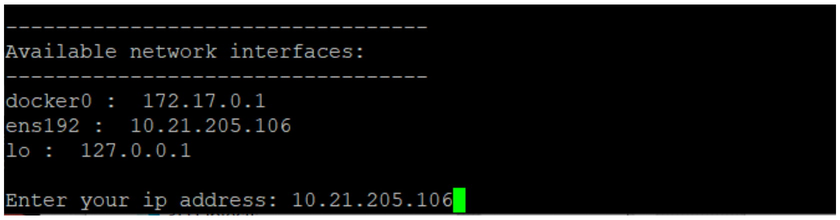
```
mwe/docker-compose-min.yml
mwe/docker-compose.yml
mwe/mongo.yml
mwe/mwe
mwe/mwemonitor-docker-compose.yml
mwe/.service.version
mwe/.env.zebra
-------------------------------------------------------------
Welcome to Motion Works Enterprise 2.0.0
-------------------------------------------------------------
Please, press ENTER to continue
>>>
```

3. Enter login credentials to download the docker images from the Zebra website hosting the MWE yum repository that contains the MWE Docker images. The username is **zebra**; obtain the password from Zebra. Re-enter the password.



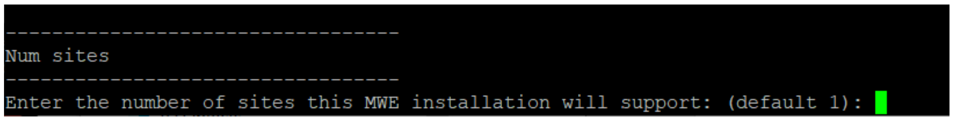
```
Enter yum repository user (cannot be empty): zebra
Enter yum repository password (text will be hidden) (cannot be empty):
Re-Enter yum repository password (text will be hidden) (cannot be empty):
```

4. Enter the IP address of the Linux server where this installation is being done. It is one of the network interfaces listed by the installation script.



```
----------------------------------
Available network interfaces:
----------------------------------
docker0 :  172.17.0.1
ens192 :  10.21.205.106
lo :  127.0.0.1

Enter your ip address: 10.21.205.106
```

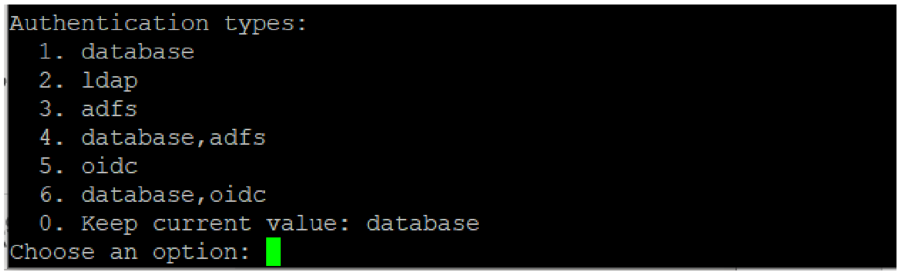
5. Enter the number of sites the MWE installation will support. If you are not sure, enter `1`. MWE automatically adjusts this value as needed.



```
----------------------------------
Num sites
----------------------------------
Enter the number of sites this MWE installation will support: (default 1):
```

6. Select an authentication type.

   At installation, select option **`1. database`**. After installation, you can choose and configure a different authentication type, as explained in the MWE 2.0 Configuration Guide.

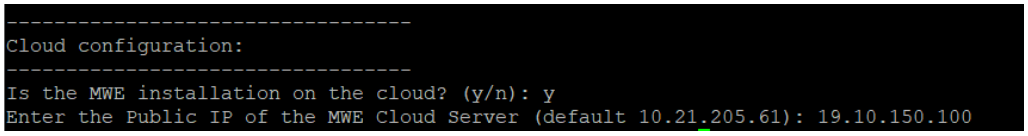


```
Authentication types:
  1. database
  2. ldap
  3. adfs
  4. database,adfs
  5. oidc
  6. database,oidc
  0. Keep current value: database
Choose an option:
```

7. Respond to **`Is the MWE installation on the cloud? (y/n)`** based on the MWE deployment:

   - If the MWE server is hosted in a private network, enter 'n'.
     Specifically, the MWE server and any reader/location sensors sending data to MWE are all on the same network, or on different subnets linked by a router. In this case the MWE server and the readers/sensors can reach each other's IP addresses and traffic between them can flow unimpeded (after opening necessary ports. See Ports).
   - If the MWE server is hosted in a cloud and has a public IP address, enter 'y'.
     Specifically, the MWE server is in a cloud behind the cloud firewall while the readers/sensors are on-premises at a site behind a local firewall.

     If you answer yes **`(y)`** you are prompted to enter the public IP address of the MWE server in the cloud.

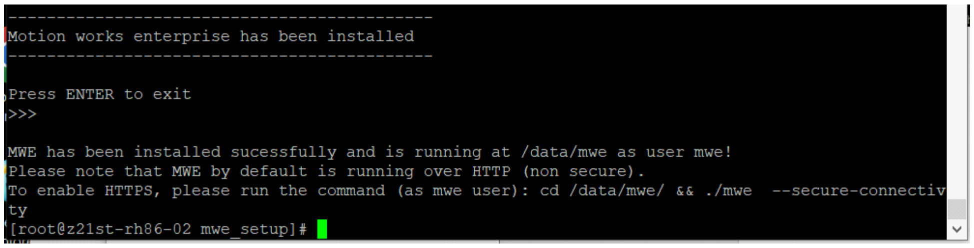


```
----------------------------------
Cloud configuration:
----------------------------------
Is the MWE installation on the cloud? (y/n): y
Enter the Public IP of the MWE Cloud Server (default 10.21.205.61): 19.10.150.100
```

8. When prompted **Is incoming https traffic allowed to local network? (y/n)** the answer is related to connections initiated by the MWE server in the cloud to the readers/sensors in the local network.

   - The MWE server in the cloud cannot directly reach the readers/sensors IP addresses in the local network because the reader IP is private and sits behind the site's firewall. There is no public IP for the readers/sensors. Therefore, HTTPS traffic from the MWE server in the cloud to a local reader is not allowed. In this case, answer no `(n)`.
   - The local RFID readers/sensors has a public Internet address, or otherwise the MWE server is in a private cloud that allows the MWE server to initiate a connection to the reader IP address. In this case answer yes `(y)`.

   The installation script installs the Docker images, which can take several minutes. When installation completes, press **Enter** to exit the installation mode.



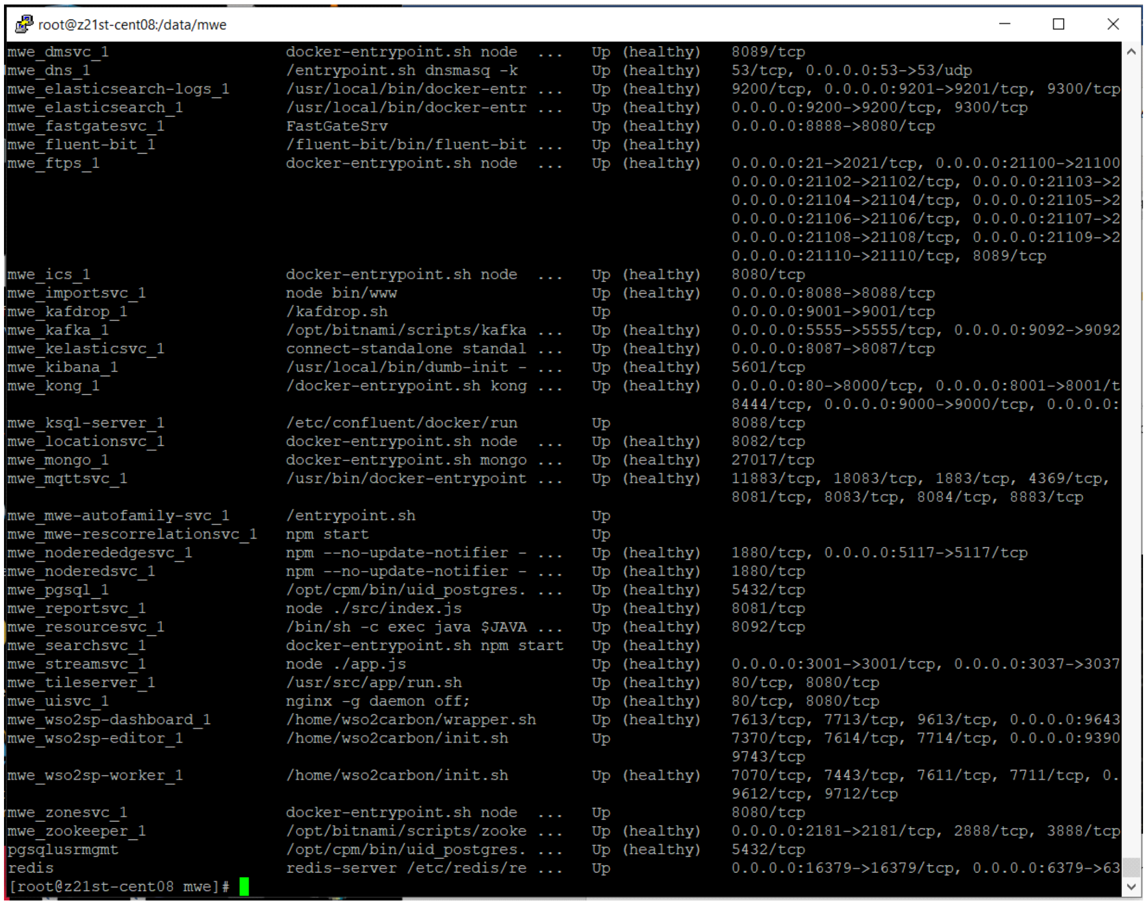
```
--------------------------------------------
Motion works enterprise has been installed
--------------------------------------------

Press ENTER to exit
>>>

MWE has been installed sucessfully and is running at /data/mwe as user mwe!
Please note that MWE by default is running over HTTP (non secure).
To enable HTTPS, please run the command (as mwe user): cd /data/mwe/ && ./mwe  --secure-connectiv
ty
[root@z21st-rh86-02 mwe_setup]#
```

Checking Status

Verify that all MWE services are up and running. Depending on the resources on your server (cpu, memory), this may take a few minutes. MWE services run as Docker containers which can be stopped and started. Use the **docker** and **docker-compose** commands to check the status of the containers.

1. To check the status of MWE services (Docker containers), run the following commands:

```
# cd /data/mwe
# docker-compose ps
```



root@z21st-cent08:/data/mwe

```
mwe_dmsvc_1                   docker-entrypoint.sh node  ...   Up (healthy)   8089/tcp
mwe_dns_1                     /entrypoint.sh dnsmasq -k        Up (healthy)   53/tcp, 0.0.0.0:53->53/udp
mwe_elasticsearch-logs_1      /usr/local/bin/docker-entr ...   Up (healthy)   9200/tcp, 0.0.0.0:9201->9201/tcp, 9300/tcp
mwe_elasticsearch_1           /usr/local/bin/docker-entr ...   Up (healthy)   0.0.0.0:9200->9200/tcp, 9300/tcp
mwe_fastgatesvc_1             FastGateSrv                      Up (healthy)   0.0.0.0:8888->8080/tcp
mwe_fluent-bit_1              /fluent-bit/bin/fluent-bit ...   Up (healthy)
mwe_ftps_1                    docker-entrypoint.sh node  ...   Up (healthy)   0.0.0.0:21->2021/tcp, 0.0.0.0:21100->21100
                                                                              0.0.0.0:21102->21102/tcp, 0.0.0.0:21103->2
                                                                              0.0.0.0:21104->21104/tcp, 0.0.0.0:21105->2
                                                                              0.0.0.0:21106->21106/tcp, 0.0.0.0:21107->2
                                                                              0.0.0.0:21108->21108/tcp, 0.0.0.0:21109->2
                                                                              0.0.0.0:21110->21110/tcp, 8089/tcp
mwe_ics_1                     docker-entrypoint.sh node  ...   Up (healthy)   8080/tcp
mwe_importsvc_1               node bin/www                     Up (healthy)   0.0.0.0:8088->8088/tcp
mwe_kafdrop_1                 /kafdrop.sh                      Up             0.0.0.0:9001->9001/tcp
mwe_kafka_1                   /opt/bitnami/scripts/kafka ...   Up (healthy)   0.0.0.0:5555->5555/tcp, 0.0.0.0:9092->9092
mwe_kelasticsvc_1             connect-standalone standal ...   Up (healthy)   0.0.0.0:8087->8087/tcp
mwe_kibana_1                  /usr/local/bin/dumb-init - ...   Up (healthy)   5601/tcp
mwe_kong_1                    /docker-entrypoint.sh kong ...   Up (healthy)   0.0.0.0:80->8000/tcp, 0.0.0.0:8001->8001/t
                                                                              8444/tcp, 0.0.0.0:9000->9000/tcp, 0.0.0.0:
mwe_ksql-server_1             /etc/confluent/docker/run        Up             8088/tcp
mwe_locationsvc_1             docker-entrypoint.sh node  ...   Up (healthy)   8082/tcp
mwe_mongo_1                   docker-entrypoint.sh mongo ...   Up (healthy)   27017/tcp
mwe_mqttsvc_1                 /usr/bin/docker-entrypoint ...   Up (healthy)   11883/tcp, 18083/tcp, 1883/tcp, 4369/tcp,
                                                                              8081/tcp, 8083/tcp, 8084/tcp, 8883/tcp
mwe_mwe-autofamily-svc_1      /entrypoint.sh                   Up
mwe_mwe-rescorrelationsvc_1   npm start                        Up
mwe_noderededgesvc_1          npm --no-update-notifier - ...   Up (healthy)   1880/tcp, 0.0.0.0:5117->5117/tcp
mwe_noderedsvc_1              npm --no-update-notifier - ...   Up (healthy)   1880/tcp
mwe_pgsql_1                   /opt/cpm/bin/uid_postgres. ...   Up (healthy)   5432/tcp
mwe_reportsvc_1               node ./src/index.js              Up (healthy)   8081/tcp
mwe_resourcesvc_1             /bin/sh -c exec java $JAVA ...   Up (healthy)   8092/tcp
mwe_searchsvc_1               docker-entrypoint.sh npm start   Up (healthy)
mwe_streamsvc_1               node ./app.js                    Up (healthy)   0.0.0.0:3001->3001/tcp, 0.0.0.0:3037->3037
mwe_tileserver_1              /usr/src/app/run.sh              Up (healthy)   80/tcp, 8080/tcp
mwe_uisvc_1                   nginx -g daemon off;             Up (healthy)   80/tcp, 8080/tcp
mwe_wso2sp-dashboard_1        /home/wso2carbon/wrapper.sh      Up (healthy)   7613/tcp, 7713/tcp, 9613/tcp, 0.0.0.0:9643
mwe_wso2sp-editor_1           /home/wso2carbon/init.sh         Up             7370/tcp, 7614/tcp, 7714/tcp, 0.0.0.0:9390
                                                                              9743/tcp
mwe_wso2sp-worker_1           /home/wso2carbon/init.sh         Up (healthy)   7070/tcp, 7443/tcp, 7611/tcp, 7711/tcp, 0.
                                                                              9612/tcp, 9712/tcp
mwe_zonesvc_1                 docker-entrypoint.sh node  ...   Up             8080/tcp
mwe_zookeeper_1               /opt/bitnami/scripts/zooke ...   Up (healthy)   0.0.0.0:2181->2181/tcp, 2888/tcp, 3888/tcp
pgsqlusrmgmt                  /opt/cpm/bin/uid_postgres. ...   Up (healthy)   5432/tcp
redis                         redis-server /etc/redis/re ...   Up             0.0.0.0:16379->16379/tcp, 0.0.0.0:6379->63
[root@z21st-cent08 mwe]#
```

2. Verify that the column **State** shows **Up** for all containers. Some services also report a health condition and should show **`healthy`**.

NOTE: You may see several warnings if you run the docker-compose command under the **root** account. Ignore these warnings. If you switch to the **mwe** account, the warnings do not appear.

```
# su - mwe
```
(if prompted, the default password is Zebra123)

```
# cd /data/mwe
# docker-compose ps
```

To switch back to the root account use:

```
# su -
```

To restart all MWE services for any reason, either reboot the server or use these commands:

```
# su - mwe
```
(if prompted, the default password is Zebra123)

```
# cd /data/mwe
# ./mwe --restart
```

MWE Files Location

The MWE files are copied to the following directories on the Linux server:

Installation package:	`/data/`
Setup files:	`/data/mwe_setup/`
MWE commands and configuration (.env):	`/data/mwe/`
MWE Services configuration:	`/data/mwe-conf/`
Docker images and containers:	`/data/docker/`
3rd Party services and database backups:	`/data/zebra`
Log files:	`/data/mwe-logs`

Ports

Under normal operation, sensors communicate directly with the MWE server, or with a ZLA which then communicates with the MWE server. This requires that some ports be open on the sensors, on the ZLA, on the MWE server, and in firewalls on the network. Additional ports are required for monitoring and troubleshooting tools. The following tables list these ports. See also the MWE Network Block Diagram.

Table 1 MWE Server Ports

Source	Destination	Protocol & Port Number	Description
Passive RFID reader running firmware 3.9.16	MWE server	TCP 21, TCP 21100-21110	Required by passive RFID readers for downloading an SSL certificate when: • Reader is running firmware version 3.9.16 • Reader is added to MWE via Device Manager • MWE is configured to use secure connection (https) to reader Note: RFID reader firmware 3.10.30 and higher uses https and port 443 to download the certificate.
Passive RFID readers	MWE server	TCP 80 or 443	Used by passive RFID readers added to MWE via Device Manager to post tag blink data. If MWE is configured to use a secure connection (https) with the readers, 443 is used. For non-secure connection, port 80 is used.
Passive RFID readers	MWE server	UDP 53	Required by passive RFID readers added to MWE via Device Manager when MWE is configured to use a secure connection using a self-signed certificate with domain name **zebramwe**.
Passive RFID readers	MWE server	TCP 9000 or 9443	Required by passive RFID readers added to MWE via Device Manager, and used for management commands and health and status reporting. If MWE is configured to use a secure connection (https) with the readers, port 9443 is used. For non-secure connection, port 9000 is used.
Browser – MWE web client	MWE server	TCP 80 or 443	Between web client and MWE server. Used for http/s client connections.
ZLA	MWE server	TCP 80 or 443	Used by ZLA for http/s and web-socket connection to MWE server.
Kafka Tool, Offset Explorer	MWE server	TCP 2181	For connecting Kafka tool (Offset Explorer) to the MWE server for troubleshooting purposes.
Tool on PC	MWE server	TCP 9092, 9093	For connecting debugging tool to Kafka for troubleshooting purposes.
SSH Client	MWE server	TCP 22	Used by SSH client to connect to MWE server.
MWE server	Zebra yum repo web site	TCP 5000	Between MWE server and Zebra yum repo web site. Required only during on-line installation.

Table 2 ZLA Ports

Source	Destination	Protocol & Port Number	Description
SSH client	ZLA	TCP 22	Used by SSH clients to remotely connect to the ZLA.
WhereLAN III and DVR sensors	ZLA	UDP 2496, UDP 12273, TCP 12285	Communication between sensors and ZLA
Telnet session or 3rd party app	ZLA	TCP 9003	Locate data stream for third party applications. Also used for diagnostics and troubleshooting.
BLE Receivers	ZLA	TCP 8005	Default port used by MPACT BLE receivers to send data to a ZLA. This is configurable.
MWE tools on Windows PC	ZLA	TCP 13287	Used by ZLA diagnostic/troubleshooting tools installed on Windows PC.
GPS tag	ZLA	TCP 12281	Used by GPS tags sending data to a ZLA via a WiFi access point.

Table 3 Sensors, Readers, and Dart Hubs Ports

Source	Destination	Protocol & Port Number	Description
ZLA	WhereLAN III and DVR sensors	UDP 12273, UDP 12282, TCP 12283	Communication between sensors and ZLA
MWE Tools on Windows PC	WhereLAN III and DVR	TCP 12277	Used by diagnostic/troubleshooting tool installed on Windows PC.
ZLA	Passive RFID reader	TCP 5084	ZLA connects to this port on a passive RFID reader using LLRP protocol.
MWE server	Passive RFID reader	TCP 80 or 443	Used by MWE server to connect to RFID reader when running R2C application.
ZLA		TCP 22	Used by ZLA to subscribe to Dart hub using SSH connection.
Telnet session on Windows PC	Dart Hub	TCP 5117	For monitoring data output on Dart hub via Telnet (port must be enabled/open on Dart hub).
Locate Analyzer tools on Windows PC	Dart Hub	TCP 5111, 5115, 5116, 5117	Used by Locate Analyzer tool to collect raw data from a Dart hub.
MWE server, web browser	Dart Hub	TCP 80 or 443	Used by web client and MWE server.

Table 4 Ports on Windows PC Running MWE Tools

Source	Destination	Protocol & Port Number	Description
WhereLAN III and DVR sensors	Windows PC running MWE Tools	TCP 12289, TCP 12293, UDP 69	Communication between sensors and Sensor Analyzer tool

Installing a Virtual ZLA

The virtual ZLA appliance is available as an OVA file with pre-loaded software ready to deploy in a hypervisor environment. A link to download this OVA file is provided upon request. Alternatively, run an installer to convert a Red Hat 7.9/8.x or CentOS 7.9 server into a ZLA appliance. The specs for the ZLA are:

- Red Hat 7.9/8.x or CentOS 7.9
- 2 vCPU's @ 2.5 GHz, 4 GB RAM
- 50 GB under / (root) partition
- Static IP

Converting a Server

To convert a Red Hat 7.9/8.x or a CentOS 7.9 server into a ZLA, request the most recent installation package from Zebra (**ZLA_installer_v1.n.tgz**, where n denotes a release/version number of the installer). See the **ReadMe.txt** provided with the installation package for more details.

1. Log into the Red Hat or CentOS server as root.
2. Copy the installer file (**ZLA_installer_v1.n.tgz**) to the **`/home`** directory on the server.
3. Run the following commands to create the ZLA appliance.

```
# cd /home
# tar -xvf ZLA-installer_v1.n.tgz
# cd ZLA
# ./install_zla.sh online  (if Red Hat/CentOS host has access to yum repository)
or
# ./install_zla.sh offline (if CentOS host does not have access to yum repository)
```

4. Verify the version of the ZLA software using the following commands.

```
# rpm -qa | grep ZLA
ZLA-2.0.3-1.i386
```

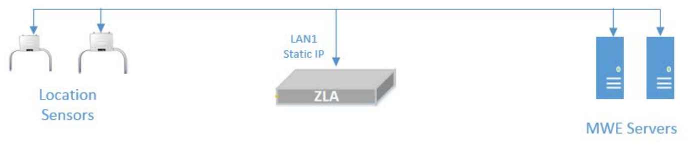
5. Upgrade the ZLA software to the latest 2.0.n-m version using the latest **ZLA-2.0.n-m.i386.rpm** package. See Upgrading ZLA Software.

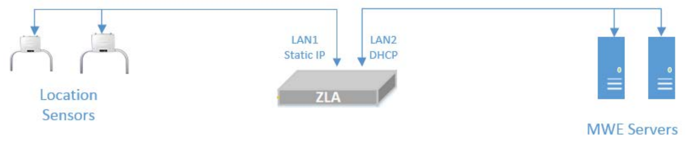
NOTE: Two services are installed: icsagent and zls. Near the end of the installation log these services display as failing to start. Disregard this, as they start when the configuration file **`site.json`** is published to the ZLA (see the MWE Configuration Guide).

ZLA Network Connections

A ZLA (Zebra Location Appliance) can be a virtual appliance (a virtual server with specialized software) or a physical appliance. A virtual ZLA has by default a single network interface, but more can be easily added as needed. A physical ZLA has 4 Ethernet ports labeled MGMT, WAN, LAN1, and LAN2, intended for optional remote management (MGMT), connecting to a Wide Area Network (WAN), and connecting to two different Local Area Networks (LAN1 and LAN2). See Connecting to a Physical ZLA for more details.

The following diagram illustrates two way to connect a ZLA to the MWE servers and location sensors in a deployment, using one or two network interfaces on the ZLA.





Note that the ZLA network interface connecting to the sensors (LAN1 in these examples) must be assigned a static IP address.

The customer IT department typically configures the network interfaces on a ZLA, using a GUI tool or directly modifying the network interface files in the **`/etc/sysconfig/network-scripts`** directory for example:

```
TYPE=Ethernet
BOOTPROTO=static
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
NAME=ens160
UUID=b43ddc0a-1a8b-492b-b10e-189ea6ee2160
ONBOOT=yes
PREFIX=24
IPADDR=192.168.1.77
DNS1=192.168.1.254
GATEWAY=192.168.1.254
```

Reboot the ZLA after making changes to the network interface configuration files.

Upgrading ZLA Software

To upgrade the software on a physical or virtual ZLA:

1. Using WinSCP or a similar tool, copy the **`ZLA-2.0.n-m.i386.rpm`** upgrade package to the **`/home`** directory on the ZLA.

   The digits **`n`** and **`m`** may change over time as new builds with minor changes become available.

2. Using Putty or a similar SSH client, log into the ZLA using the root account (obtain login credentials from Zebra).

3. Verify that the **firewalld** daemon is running.

   ```
   # systemctl start firewalld
   # systemctl status firewalld
   ```

4. Verify that the status is reported as **`active (running)`**.

5. Check the current version of the ZLA software.

   ```
   # rpm -qa | grep ZLA
   ```

6. Perform the software upgrade using the following commands:

   ```
   # cd /home
   # rpm -Uvh ZLA-2.0.n-m.i386.rpm
   ```

   Replace **n** and **m** with the values of your ZLA sof t war e

7. Restart the services:

   ```
   # systemctl restart icsagent
   # systemctl restart zls
   ```

8. Check services status:

   ```
   # systemctl status icsagent
   # systemctl status zls
   ```

   For icsagent, the status should be **`active (running)`** if the ZLA was previously configured via the **./configure.sh** command (see Registering a ZLA). Otherwise the status is **`activating`** or **`failed`**. This is expected.

   For the zls service, if no **`site.json`** configuration file has been published to the ZLA the status may show as **`activating`** or **`failed`**. If a default **`site.json`** file exists on the ZLA, it may show **`active (running)`**.

9. Check the version of the ZLA software just installed.

   ```
   # rpm -qa | grep ZLA
   ```

Configuring a ZLA

In the current release of the ZLA software, the environmental variable WHERENE_HOST_IP may need to be set, and the time zone, system time, and hostname may require updating.

Setting WHERENET_HOST_IP

If the ZLA has more than one network interface connected to the network (such as LAN1 and LAN2 in Upgrading ZLA Software), specify to the location engine running on the ZLA which interface is receiving data from the Location Sensors. To do this, set the WHERENET_HOST_IP environment variable to the static IP address of that network card, that is, the network card on the same subnet as the Location Sensors.

1. Log into the ZLA using the root account (obtain login credentials from Zebra), and open a terminal window. You can also use Putty, WinSCP, or similar SSH client to remotely access the ZLA. Putty provides a command line interface, while WinSCP provides a graphical user interface to view and edit files and directories.

NOTE: If you have not already configured the network interfaces on the ZLA, see Upgrading ZLA Software and Connecting to a Physical ZLA. You should then have a known IP address to connect to the ZLA using Putty or WinSCP.

2. If using a terminal window, change directory to **/etc/systemd/system** with the command: **`# cd /etc/systemd/system`**. If using WinScp, browse to this folder.

3. Open the **zls.service** file for editing. Use a text editing command such as **vi** or, if using WinSCP, double-click on the file. The content of this file is:

```
[Unit]
Description=Zebra Location Service
After=network.target

[Service]
Type=forking
Environment=ZEBRA_HOME=/opt/zebra
Environment=LD_LIBRARY_PATH=/opt/zebra/zla/zlpcore/bin
Environment=WHERENET_HOST_IP=192.168.30.120
ExecStart=/opt/zebra/zla/zlpcore/bin/start-zls.sh
ExecStop=/opt/zebra/zla/zlpcore/bin/stop-zls.sh
KillMode=none
Restart=always
RestartSec=15
StartLimitInterval=60
StartLimitBurst=3
[Install]
WantedBy=multi-user.target
```

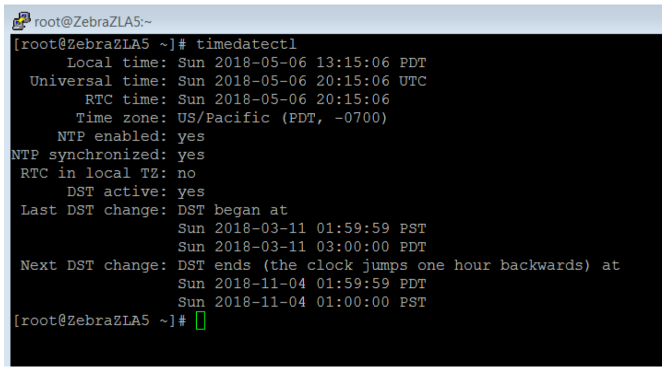
4. Add or edit the line highlighted in red above to set the IP address to the (static) IP of the network interface on the same subnet as the Location Sensors. Save and close the file.

5. To implement the changes, run the following command in a terminal window:

```
# systemctl daemon-reload
```

ZLA Time and Date

1. Log into the ZLA using the root account (obtain login credentials from Zebra) and open a Terminal window. You can use Putty or similar SSH client to remotely access the ZLA.
2. To check the current time and date on your ZLA, use the **`timedatectl`** command.



root@ZebraZLA5:~

```
[root@ZebraZLA5 ~]# timedatectl
      Local time: Sun 2018-05-06 13:15:06 PDT
  Universal time: Sun 2018-05-06 20:15:06 UTC
        RTC time: Sun 2018-05-06 20:15:06
       Time zone: US/Pacific (PDT, -0700)
     NTP enabled: yes
NTP synchronized: yes
 RTC in local TZ: no
      DST active: yes
 Last DST change: DST began at
                  Sun 2018-03-11 01:59:59 PST
                  Sun 2018-03-11 03:00:00 PDT
 Next DST change: DST ends (the clock jumps one hour backwards) at
                  Sun 2018-11-04 01:59:59 PDT
                  Sun 2018-11-04 01:00:00 PST
[root@ZebraZLA5 ~]#
```

3. Set the Time Zone.

   In this example the Time Zone is set to US/Pacific (PDT). To change it use one of these commands:

```
# timedatectl set-timezone US/Pacific
# timedatectl set-timezone US/Mountain
# timedatectl set-timezone US/Central
# timedatectl set-timezone US/Eastern
```

   To see a list of worldwide Time Zones, use the command:

```
# timedatectl list-timezones
```

4. Set the correct time.

   The ZLA includes an NTP client (chrony) that automatically syncs the system clock with NTP servers on the Internet. If the ZLA does not have access to the Internet, to use an NTP server on a local network, in the **/etc/chrony.conf** file, add the line highlighted in red font below, replacing the IP address with the IP address or name of your local NTP server.

```
# Use public servers from the pool.ntp.org project.
# Please consider joining the pool (http://www.pool.ntp.org/join.html).
server 0.centos.pool.ntp.org iburst
server 1.centos.pool.ntp.org iburst
server 2.centos.pool.ntp.org iburst
server 3.centos.pool.ntp.org iburst
server 192.168.1.50 prefer iburst
```

5. Restart the NTP client to implement the change:

```
# systemctl restart chronyd
```

6. If required, use the following commands to manually set the date and time:

```
# timedatectl set-ntp 0(this command will disable the NTP client)
# timedatectl set-time YYYY-MM-DD
```

   (as in timedatectl set-time 2018-05-07)

```
# timedatectl set-time HH:MM:SS
```

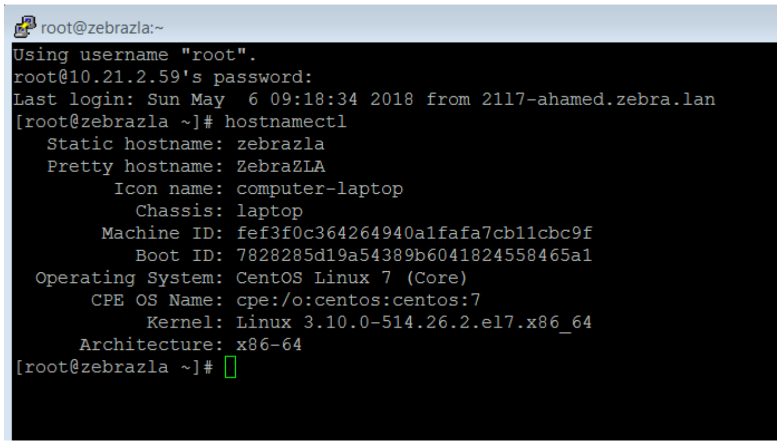
   (as in timedatectl set-time 14:45:05)

NOTE: ZLA supports only one time zone, so all site maps associated with a ZLA in MWE have the same time zone. Use separate ZLA's for sites in different time zones.

Changing the Hostname

1. Log into the ZLA using the root account (obtain login credentials from Zebra) and open a Terminal window. You can use Putty or similar SSH client to remotely access the ZLA.

   A ZLA ships with the default hostname **ZebraZLA**, located in the command line prompt shown in the following screen. Changing this default hostname is recommended. The **`hostnamectl`** command provides more details.



```
Using username "root".
root@10.21.2.59's password:
Last login: Sun May  6 09:18:34 2018 from 2117-ahamed.zebra.lan
[root@zebrazla ~]# hostnamectl
   Static hostname: zebrazla
   Pretty hostname: ZebraZLA
         Icon name: computer-laptop
           Chassis: laptop
        Machine ID: fef3f0c364264940a1fafa7cb11cbc9f
           Boot ID: 7828285d19a54389b6041824558465a1
  Operating System: CentOS Linux 7 (Core)
       CPE OS Name: cpe:/o:centos:centos:7
            Kernel: Linux 3.10.0-514.26.2.el7.x86_64
      Architecture: x86-64
[root@zebrazla ~]#
```

2. To change the hostname, use the command:

   `# hostnamectl set-hostname` *`NewZLAName`*

   This command changes the hostname at the Kernel level (static name). The name displayed at the command prompt (transient name) is updated after a reboot of the ZLA via the **`reboot`** command.

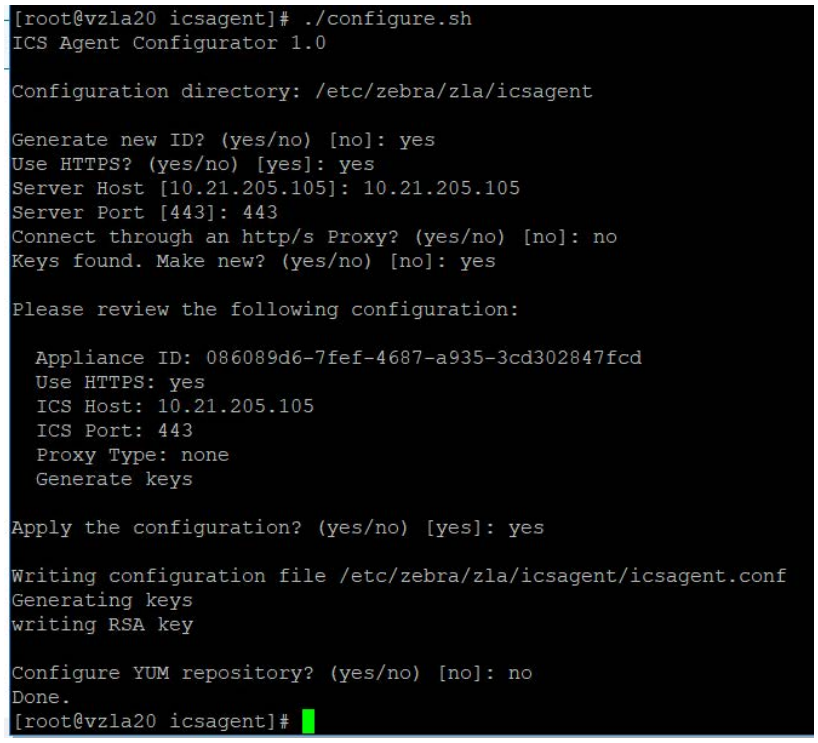
Registering a ZLA

To register the ZLA appliance with the MWE server, which enables the ZLA to forward data to the server, and the MWE web client to monitor, configure, and update the ZLA:

1. Using Putty or a similar SSH client, log into the ZLA using the root account (obtain login credentials from Zebra).
2. Change directory to **/opt/zebra/zla/icsagent** and run the configure script:

```
# cd /opt/zebra/zla/icsagent
# ./configure.sh
```

3. Respond to the prompts as shown in the following figure. For Server Host, enter the IP address or fully qualified domain server name of the MWE Linux server.

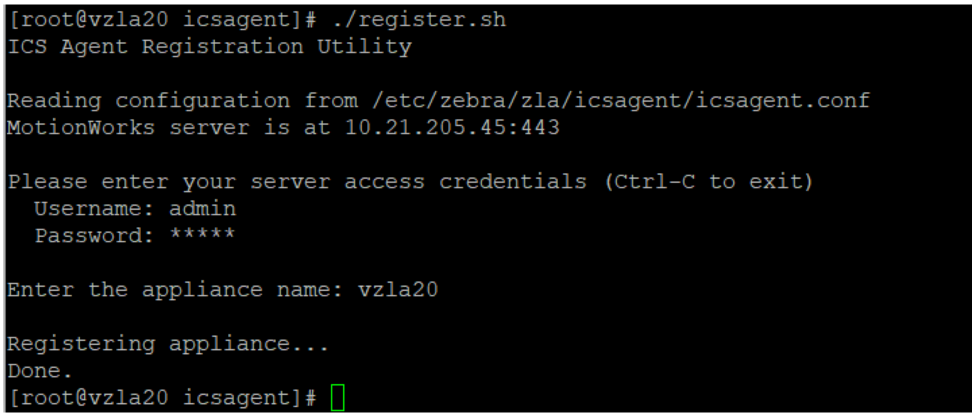


```
[root@vzla20 icsagent]# ./configure.sh
ICS Agent Configurator 1.0

Configuration directory: /etc/zebra/zla/icsagent

Generate new ID? (yes/no) [no]: yes
Use HTTPS? (yes/no) [yes]: yes
Server Host [10.21.205.105]: 10.21.205.105
Server Port [443]: 443
Connect through an http/s Proxy? (yes/no) [no]: no
Keys found. Make new? (yes/no) [no]: yes

Please review the following configuration:

  Appliance ID: 086089d6-7fef-4687-a935-3cd302847fcd
  Use HTTPS: yes
  ICS Host: 10.21.205.105
  ICS Port: 443
  Proxy Type: none
  Generate keys

Apply the configuration? (yes/no) [yes]: yes

Writing configuration file /etc/zebra/zla/icsagent/icsagent.conf
Generating keys
writing RSA key

Configure YUM repository? (yes/no) [no]: no
Done.
[root@vzla20 icsagent]#
```

4. The current or default value is shown in the square brackets [ ]. To accept it, press the **Enter** key.
5. If there is a proxy server between the ZLA and the MWE server, answer **yes** to the server proxy question and enter the URL for the proxy server.
6. Run the register script:

```
#  ./register.sh
```

7. When prompted, enter the **Username** and **Password** (the default is **admin** / **admin**), and enter a name for the ZLA to appear in the MWE web client. Using the same name configured as **hostname** in the previous step is recommended.



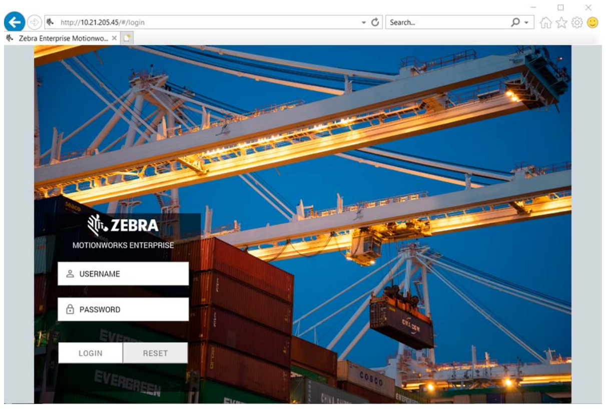
```
[root@vzla20 icsagent]# ./register.sh
ICS Agent Registration Utility

Reading configuration from /etc/zebra/zla/icsagent/icsagent.conf
MotionWorks server is at 10.21.205.45:443

Please enter your server access credentials (Ctrl-C to exit)
  Username: admin
  Password: *****

Enter the appliance name: vzla20

Registering appliance...
Done.
[root@vzla20 icsagent]#
```

8. Restart the **icsagent** daemon:

```
# systemctl restart icsagent
```

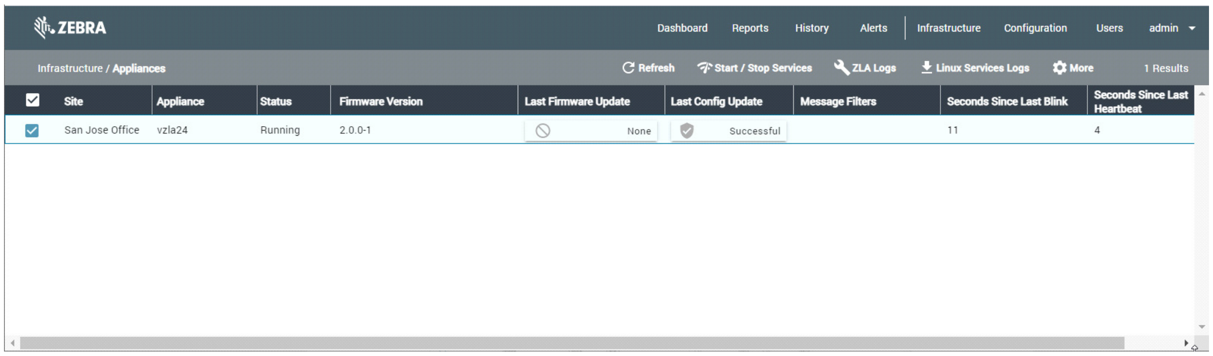
Launching the Web Client

1. Open a web browser (Chrome, Edge, or Firefox) on a client machine or server on the network, and enter http://*MWE_Server_Name* where *MWE_Server_Name* is the MWE Linux server name or IP address.
2. On the login page enter the **Username** and **Password** (the default is **admin** / **admin**).



NOTE: https://MWE_Server_Name is also supported. See the MWE Configuration Guide for information on adding a certificate on the MWE server.

3. Open the **Infrastructure** > **Appliances** page to view the previously registered ZLA.



✓	Site	Appliance	Status	Firmware Version	Last Firmware Update	Last Config Update	Message Filters	Seconds Since Last Blink	Seconds Since Last Heartbeat
✓	San Jose Office	vzla24	Running	2.0.0-1	None	Successful		11	4

The **Status** column displays **Failed** (or **Activating**) until a **site.json** configuration file is published using the System Builder tool (see the MWE Configuration Guide for details). If your ZLA includes a default **site.json** file, the **Status** column may show **Running**, or alternate between **Running** and **Offline** if the ZLA software has not been upgraded to version 2.0.0.

Installing a World Map

After launching the web client, open the **Infrastructure** > **Site Manager** page for a default background world map. To optionally load a more detailed background map, for example a map with street-level information for the U.S. or other countries:

1. Locate the map in mbtiles format (extension .mbtiles). https://openmaptiles.com/downloads/planet/ offers maps with street-level information for different regions of the world.

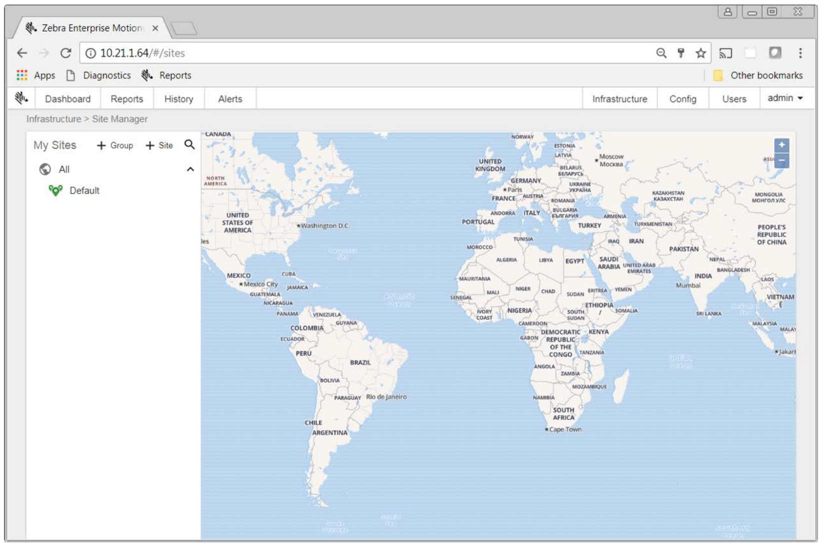
   For example, the file `north-america_us.mbtiles` contains U.S. street-level information. Note that downloading this large file (7 GB) may take some time.

2. Assuming you have installed MWE under `/data`, copy the map file to `/data/zebra/mwe/3rdParty/tileserver-data` on the Linux server, then delete or move the previous world map from this folder. The tileserver service, which processes the file, reads only one file from this folder.

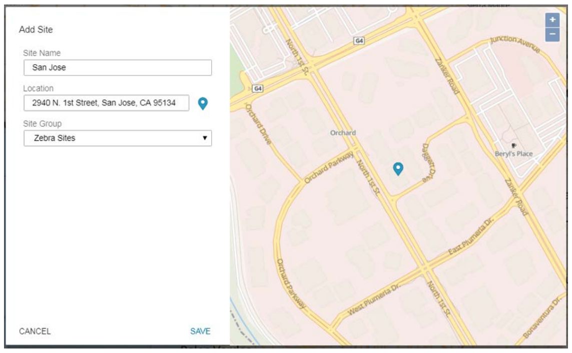
3. To restart the tileserver service on the MWE server, run the commands:

   # cd /data/mwe (assuming you have installed MWE under **/data**)
   ```
   # docker-compose stop tileserver
   # docker-compose start tileserver
   ```

4. Relaunch the web client. Initially, the **Infrastructure** > **Site Manager** page may take 30- 60 seconds to display the more detailed **north-america_us.mbtiles** world map.

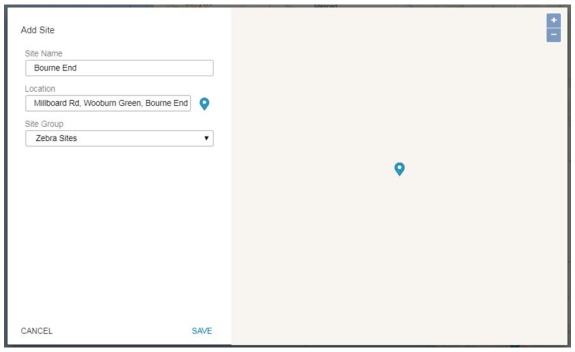


When adding a site, the **Add Site** window shows street level details of the US address you entered as shown below.



Site Name: San Jose
Location: 2940 N. 1st Street, San Jose, CA 95134
Site Group: Zebra Sites

For other regions of the world, no map appears in the **Add Site** window.



Site Name: Bourne End
Location: Millboard Rd, Wooburn Green, Bourne End
Site Group: Zebra Sites

If you keep the default installation world map, no map appears in the **Add Site** window for US addresses. This has no impact on MWE functionality.

Upgrading Sensor Firmware

MWE supports a variety locating sensors and technologies:

- BLE beacons and receivers
- Passive RFID readers
- Dart UWB sensors (ISO 24730-61)
- DVR UWB sensors
- GPS tags
- WhereLAN sensors (ISO 24730)
- Magnetic beacons

Consult Zebra for an updated list.

When deploying sensors in an MWE solution, obtain from Zebra the sensor firmware version compatible with MWE 2.0 and the firmware upgrade procedure for each type of sensor.

For example, WhereLAN sensors require firmware v.6.5.2 or later, and DVR sensors require firmware v.5.2.1 or later. MWE's Sensor Analyzer tool can perform firmware upgrades for these sensors, as explained in separate documentation.

Configuring MWE

After installation, several configuration tasks are required for a fully operational system. Refer to the following documents for instructions:

- MWE Configuration Guide - describes configuration tasks that are done only once or seldom after installing the software, such as creating sites and site groups, uploading and calibrating site maps, defining zones and zone groups, and specifying location devices and algorithms for system use. The system is fully functional after this configuration.
- MWE User Guide - describes the basic functionality of the web client for end users, and includes configuration tasks that further customize the application or that are performed on a frequent basis such as adding users and user groups, defining access permissions, defining resource types, associating tags with resources, defining data filters, and configuring the various reports (columns displayed and column order).

Upgrading from MWE 1.4.x to MWE 2.0

The upgrade script transfers data from the SQL database in MWE 1.4 to a mongoDB database on the MWE Linux server, but not all data.

The following data is transferred and available after the upgrade:

- Users, User groups, and Group permission configuration
- Contacts
- Zones and Zone groups
- Tags associated with resources
- Resource Types
- Resources
- Named resource custom fields and their values. This applies to custom fields named in the Configuration > Token Replacement Settings report in the MWE web client. For example **`~Object Custom1~ = Color`** is transferred. However, a custom field that retains its default name, such as **`~Object Custom2~ = Resource Custom2`**, is not transferred since it is assumed not in use.

The following data is not transferred and not available after upgrade:

- The Zone and Zone Group columns in the Tags and Resources report are empty and the Site Name show **Default** until the tags blink again.
- Unassigned tags. This applies to tags not associated with a resource ID.
- Business rules (resource alerts). A script is provided that saves these rules so that they can be manually re-created after the upgrade.
- Resource custom properties with default token names. Again, a custom field that retains its default name, such as for example **`~Object Custom2~ = Resource Custom2`**, is not transferred since it is assumed not in use.
- Assignment of recipients to system alerts. If you configured what system alerts are emailed to certain recipients, you must manually re-assign recipients to system alerts after upgrade.
- Saved/custom reports must be recreated manually. In MWE 2.0.4 and later, you can create a saved report and propagate it to other users.
- Token replacement of report and column names. This is not yet supported in MWE 2.0
- Tag ID's that contain non-printable characters (such as Line Feed), which were introduced into the system by data import operations. This is rare.

The following data stored in the Elastic Search database on the Linux server in MWE 1.4.3 remains accessible after the upgrade:

- Tag blink history
- Zone change history
- Event history

Requirements

MWE 2.0 requires more memory and advanced CPU on the Linux host than MWE 1.4. Not having sufficient memory and CPU may cause the upgrade to fail or MWE services to crash after upgrade. Review server requirements in Minimum Server Specs.

For offline upgrades, the MWE 2.0 installation script requires installing the package **policycoreutils-python** on a CentOS or Red Hat server before upgrading, otherwise the installation script exits. An Ubuntu server does not require this package.

To determine if this package is installed run the following commands:

- On CentOS 7.9 and Red Hat 7.9:

```
# rpm -q policycoreutils-python
```

- On Red Hat 8.x:

```
# rpm -q policycoreutils-python-utils
```

If the server has Internet connection, run the following commands to install this package:

- On CentOS 7.9 or Red Hat 7.9:

```
# yum install policycoreutils-python
```

- On Red Hat 8.x:

```
# yum install policycoreutils-python-utils
```

Pre-Upgrade Instructions

The off-line upgrade option does not require the Linux server to have Internet access for installation. Upgrade requires copying a **tar.gz** file to the MWE Linux server. Due to its size (close to 7 GB), consider copying the file prior to a scheduled upgrade.

Before upgrading, verify the requirements in Minimum Server Specs for the MWE Linux Server are met.

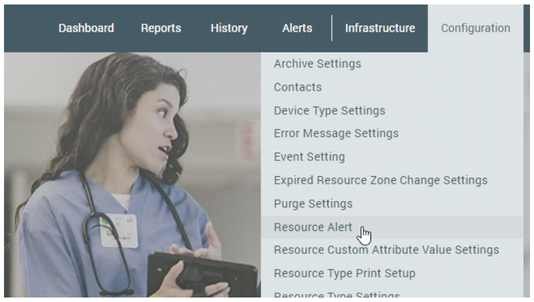
Taking a Snapshot

Take a snapshot of the MWE 1.4 Linux virtual server before upgrading, to use to quickly restore the original state of the system if upgrade fails. Upgrade does not affect the MWE 1.4 Windows application server or the SQL Server host, so a snapshot is not required for these servers.

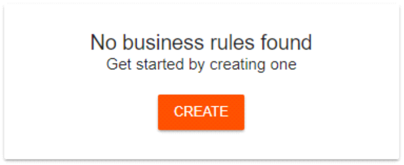
Backing Up Business Rules

The upgrade script does not preserve business rules (resource alerts) defined in MWE 1.4. The **copyBusinessRules.sh** script is provided to save the rules configuration to a file, to use after upgrade to manually add back these rules.

Verify in the Resource Alerts report if business rules (resource alerts) were defined in MWE 1.4.



If no Business Alerts exist, the following screen appears. Proceed to Stopping Services.



To save business rules:

1. Copy the **copyBusinessRules.sh** script to the **/data** directory on the Linux server.
2. Using Putty or a Terminal window, log into the MWE Linux server as root and change the file permissions to allow execution of the script:

   ```
   # cd /data
   # chmod +x copyBusinessRules.sh
   ```

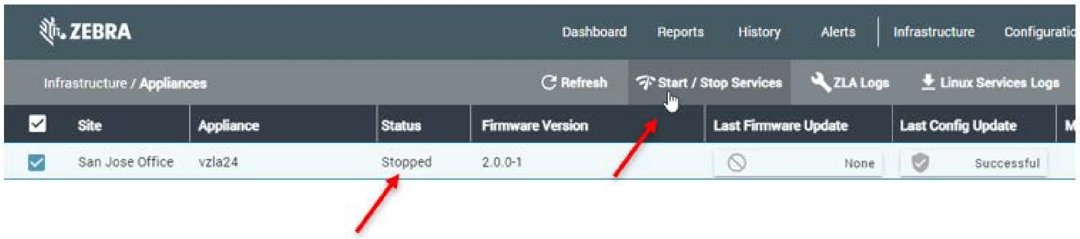
3. Run the script:

   ```
   # ./copyBusinessRules.sh
   ```

This script generates a directory **/data/saved-rules** where each existing business rule is saved as a json file.

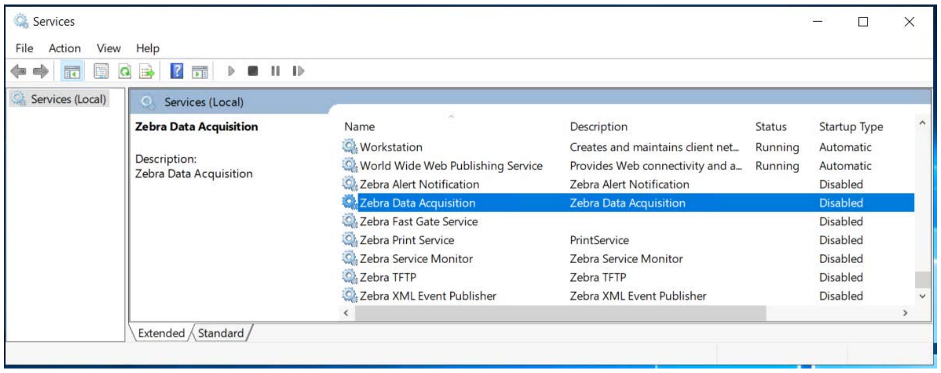
Stopping Services

1. Stop services on ZLA's listed in the Appliances report.



Site	Appliance	Status	Firmware Version	Last Firmware Update	Last Config Update
San Jose Office	vzla24	Stopped	2.0.0-1	None	Successful

2. Using Windows Services (Service Control Manager) stop and disable all Zebra services on the MWE 1.4 Windows application service. The services that appear in this screen depend on the optional modules installed on top of MWE 1.4.



Zebra Data Acquisition

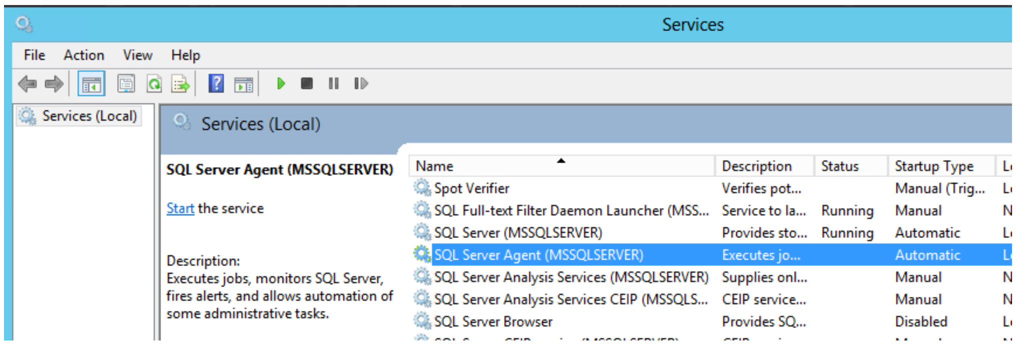
Description:
Zebra Data Acquisition

Name	Description	Status	Startup Type
Workstation	Creates and maintains client net...	Running	Automatic
World Wide Web Publishing Service	Provides Web connectivity and a...	Running	Automatic
Zebra Alert Notification	Zebra Alert Notification		Disabled
Zebra Data Acquisition	Zebra Data Acquisition		Disabled
Zebra Fast Gate Service			Disabled
Zebra Print Service	PrintService		Disabled
Zebra Service Monitor	Zebra Service Monitor		Disabled
Zebra TFTP	Zebra TFTP		Disabled
Zebra XML Event Publisher	Zebra XML Event Publisher		Disabled

3. Stop **SQL Server Agent** on the Windows server hosting SQL Server.

   IMPORTANT: Do not stop the **SQL Server** service.

   If you cannot stop SQL Server Agent because doing so would impact other non-MWE databases hosted on the same SQL Server host, just disable all MWE database scheduled jobs in SQL Server Management Studio.



Services

File Action View Help

Services (Local)

Services (Local)

SQL Server Agent (MSSQLSERVER)

Start the service

Description:
Executes jobs, monitors SQL Server, fires alerts, and allows automation of some administrative tasks.

Name	Description	Status	Startup Type	L
Spot Verifier	Verifies pot...		Manual (Trig...	L
SQL Full-text Filter Daemon Launcher (MSS...	Service to la...	Running	Manual	N
SQL Server (MSSQLSERVER)	Provides sto...	Running	Automatic	L
SQL Server Agent (MSSQLSERVER)	Executes jo...		Automatic	L
SQL Server Analysis Services (MSSQLSERVER)	Supplies onl...		Manual	N
SQL Server Analysis Services CEIP (MSSQLS...	CEIP service...		Manual	N
SQL Server Browser	Provides SQ...		Disabled	L

Upgrading

You can perform an off-line upgrade or an on-line upgrade.

- The off-line upgrade option does not require the MWE Linux server to have Internet access during installation. Because the installation package is close to 8.5 GB, copying it prior to a scheduled upgrade is recommended.
- On-line upgrade requires that the MWE Linux server have open access to the Internet during installation.

Copying and Extracting the Installation Package

1. Using WinSCP, Putty, or a similar SSH client, copy the installation package to the **/data** directory on the Linux server:
   - For an off-line upgrade, copy the **`mwe-containers-setup-2.0.n.m-offline.tar.gz`** installation package.
   - For an on-line upgrade, copy the **`mwe-containers-setup-2.0.n.m-online.tar.gz`** installation package.
2. Using a Terminal window, Putty, or another SSH client, log into the MWE server using the **root** account.
3. Change the directory to **`/data`**.

   ```
   # cd /data
   ```
4. Extract the files.
   - For an off-line upgrade, enter:

     ```
     # tar -xvf  mwe-containers-setup-2.0.n.m-offline.tar.gz
     ```
   - For an on-line upgrade, enter:

     ```
     # tar -xvf  mwe-containers-setup-2.0.n.m-online.tar.gz
     ```

   A set of files is extracted.

NOTE: Occasionally characters are misinterpreted when copying a command from a document and pasting it into a Putty or Terminal window. Some commands may require manual entry.

Running the Upgrade Script

1. Change the directory to **`/data/mwe_setup`**.

   ```
   # cd /data/mwe_setup
   ```

2. Run the upgrade script.

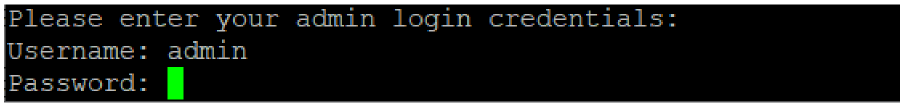
   - For an off-line upgrade, enter:

     ```
     # ./mwe_setup.sh --offline-upgrade
     ```

   - For an on-line upgrade, enter:

     ```
     # ./mwe_setup.sh --upgrade
     ```

3. The upgrade script begins upgrading the Docker images. When prompted, enter your login credentials.



```
Please enter your admin login credentials:
Username: admin
Password:
```

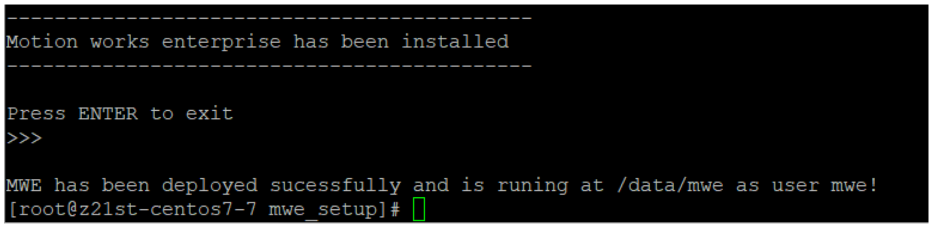
   These are the same admin account login credentials used to log in to the MWE web client. Use the admin account or another account in the MWE Administrator user group.

   The upgrade script uses these credentials to log into the SQL server and migrate the data in the SQL db_MWE database to a mongoDB database on the MWE server.

   NOTE: No characters are shown when you type the password. If you enter the wrong password, you are prompted to re-enter the login credentials.

   The upgrade/migration script runs for a few more minutes.

4. When the upgrade completes, press **`Enter`** to exit upgrade mode.



```
-------------------------------------------
Motion works enterprise has been installed
-------------------------------------------

Press ENTER to exit
>>>

MWE has been deployed sucessfully and is runing at /data/mwe as user mwe!
[root@z21st-centos7-7 mwe_setup]#
```

Running the Upgrade Again

If the upgrade script encounters an error condition, it posts an error message and exits without completing the upgrade. If this occurs, consult the following logs:

- Installation log: **/data/mwe_setup/mwe_setup-*timestamp*.log**
- Data migration log: **/var/log/zebra/mwe/db_migration.log**

If possible, correct the issue reported in the logs, restore the MWE 1.4 Linux snapshot, and repeat the upgrade process starting with Requirements.

If you encounter an issue you cannot resolve, see Reverting to MWE 1.4 to restore the original MWE 1.4, and contact Zebra Product Support for assistance.

Validating the Upgrade

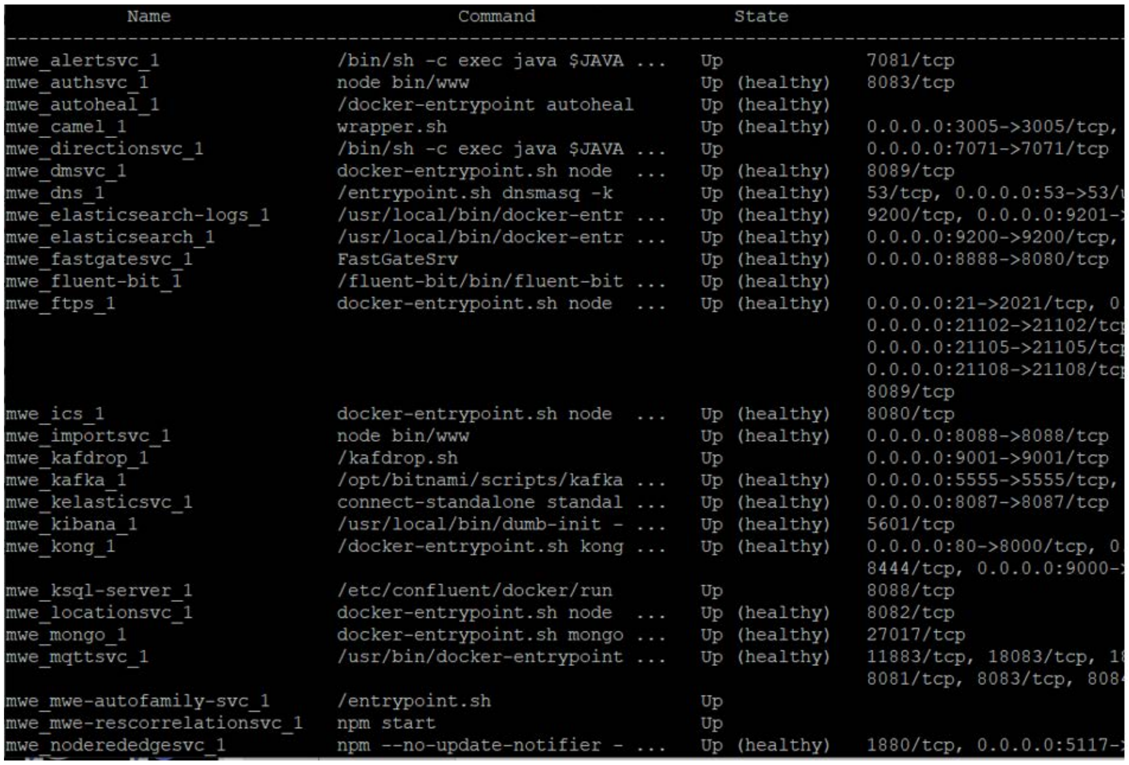
Once the upgrade script completes, verify the following:

1. Verify all MWE Services are up and running by logging into the MWE Linux server using Putty or a similar SSH client and running the **`docker-compose ps`** command as **`mwe`** user.

   Depending on the resources (CPU and memory) on your server, it may take a few minutes for services to be up and running.

   ```
   # su – mwe
   # cd /data/mwe
   # docker-compose ps
   ```

   Ensure the **State** column displays **Up** for all services.



```
          Name                            Command                State
------------------------------------------------------------------------------------------------
mwe_alertsvc_1              /bin/sh -c exec java $JAVA ...   Up               7081/tcp
mwe_authsvc_1               node bin/www                     Up (healthy)     8083/tcp
mwe_autoheal_1              /docker-entrypoint autoheal      Up (healthy)
mwe_camel_1                 wrapper.sh                       Up (healthy)     0.0.0.0:3005->3005/tcp,
mwe_directionsvc_1          /bin/sh -c exec java $JAVA ...   Up               0.0.0.0:7071->7071/tcp
mwe_dmsvc_1                 docker-entrypoint.sh node  ...   Up (healthy)     8089/tcp
mwe_dns_1                   /entrypoint.sh dnsmasq -k        Up (healthy)     53/tcp, 0.0.0.0:53->53/
mwe_elasticsearch-logs_1    /usr/local/bin/docker-entr ...   Up (healthy)     9200/tcp, 0.0.0.0:9201-
mwe_elasticsearch_1         /usr/local/bin/docker-entr ...   Up (healthy)     0.0.0.0:9200->9200/tcp,
mwe_fastgatesvc_1           FastGateSrv                      Up (healthy)     0.0.0.0:8888->8080/tcp
mwe_fluent-bit_1            /fluent-bit/bin/fluent-bit ...   Up (healthy)
mwe_ftps_1                  docker-entrypoint.sh node  ...   Up (healthy)     0.0.0.0:21->2021/tcp, 0
                                                                              0.0.0.0:21102->21102/tc
                                                                              0.0.0.0:21105->21105/tc
                                                                              0.0.0.0:21108->21108/tc
                                                                              8089/tcp
mwe_ics_1                   docker-entrypoint.sh node  ...   Up (healthy)     8080/tcp
mwe_importsvc_1             node bin/www                     Up (healthy)     0.0.0.0:8088->8088/tcp
mwe_kafdrop_1               /kafdrop.sh                      Up               0.0.0.0:9001->9001/tcp
mwe_kafka_1                 /opt/bitnami/scripts/kafka ...   Up (healthy)     0.0.0.0:5555->5555/tcp,
mwe_kelasticsvc_1           connect-standalone standal ...   Up (healthy)     0.0.0.0:8087->8087/tcp
mwe_kibana_1                /usr/local/bin/dumb-init - ...   Up (healthy)     5601/tcp
mwe_kong_1                  /docker-entrypoint.sh kong ...   Up (healthy)     0.0.0.0:80->8000/tcp, 0
                                                                              8444/tcp, 0.0.0.0:9000-
mwe_ksql-server_1           /etc/confluent/docker/run        Up               8088/tcp
mwe_locationsvc_1           docker-entrypoint.sh node  ...   Up (healthy)     8082/tcp
mwe_mongo_1                 docker-entrypoint.sh mongo ...   Up (healthy)     27017/tcp
mwe_mqttsvc_1               /usr/bin/docker-entrypoint ...   Up (healthy)     11883/tcp, 18083/tcp, 1
                                                                              8081/tcp, 8083/tcp, 808
mwe_mwe-autofamily-svc_1    /entrypoint.sh                   Up
mwe_mwe-rescorrelationsvc_1 npm start                        Up
mwe_noderededgesvc_1        npm --no-update-notifier - ...   Up (healthy)     1880/tcp, 0.0.0.0:5117-
```

2. Log in to the MWE web client as **`admin`** and verify the data in the following reports.

   - Configuration/Contacts
   - Configuration/Zone Settings
   - Configuration/Resource Types
   - Reports/Tags
   - Reports/Resources
   - Infrastructure/Site Manager/Configure zones

NOTE: If an error message occurs when opening a report, clear your browser cache.

To restart all MWE services for any reason, reboot the server or use these commands:

`# su – mwe` (if prompted for password, the default password is Zebra123)

```
# cd /data/mwe
# ./mwe --restart
```

Reverting to MWE 1.4

To revert to MWE 1.4 because the upgrade script exited due to an error or another reason:

1. Restore the MWE 1.4 snapshot (Linux server).
2. Enable and start services on the MWE 1.4 Windows application server.
3. Start the SQL Server Agent service on the SQL server host.
4. In the Appliances report in the MWE web bclient, start the ZLA services.

Upgrading ZLA Software

After a successful upgrade of the MWE Linux for deployments including a ZLA, upgrade the ZLA software to a version compatible with the MWE version. See Upgrading ZLA Software.

Upgrading from MWE 2.0.n to MWE 2.0.m

You can perform an off-line upgrade or an on-line upgrade.

- The off-line upgrade option does not require the MWE Linux server to have Internet access during installation. Because the installation package is close to 8.5 GB, copying it prior to a scheduled upgrade is recommended.
- On-line upgrade requires that the MWE Linux server have open access to the Internet during installation.

Take a Snapshot

Following best practices, capture a snapshot of your virtual machine before upgrading, in case you need to restore the previous version.

Copying and Extracting the Installation Package

1. Using WinSCP, Putty, or a similar SSH client, copy the installation package to the **/data** directory on the Linux server:
   - For an off-line upgrade, copy the **`mwe-containers-setup-2.0.m-offline.tar.gz`** installation package.
   - For an on-line upgrade, copy the **`mwe-containers-setup-2.0.m-online.tar.gz`** installation package.
2. Using a Terminal window, Putty, or another SSH client, log into the MWE server using the **root** account.
3. Change the directory to **`/data`**.
   ```
   # cd /data
   ```
4. Extract the files.
   - For an off-line upgrade, enter:
     ```
     # tar -xvf  mwe-containers-setup-2.0.m-offline.tar.gz
     ```
   - For an on-line upgrade, enter:
     ```
     # tar -xvf  mwe-containers-setup-2.0.m-online.tar.gz
     ```

   A set of files is extracted.

NOTE: Occasionally characters are misinterpreted when copying a command from a document and pasting it into a Putty or Terminal window. Some commands may require manual entry.

Running the Upgrade Script

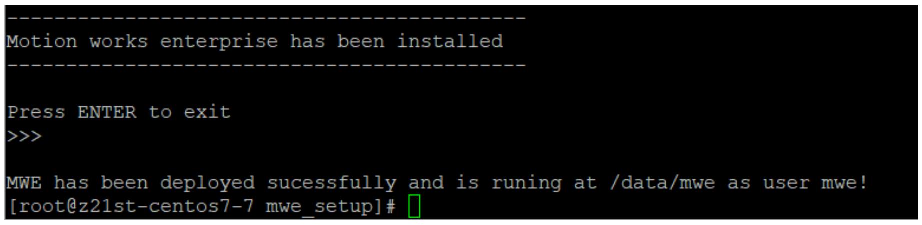
1. Change the directory to **`/data/mwe_setup`**.
   ```
   # cd /data/mwe_setup
   ```
2. If upgrading from 2.0.1, run the following command (skip this if upgrading from a higher 2.0.n version):
   ```
   # ./mwe_setup.sh -update-registry
   ```
3. Run the upgrade script.
   - For an off-line upgrade, enter:

```
# ./mwe_setup.sh --offline-upgrade
```

- For an on-line upgrade, enter:

```
# ./mwe_setup.sh --upgrade
```

4. When upgrade completes, press **Enter** to exit upgrade mode.



```
--------------------------------------------
Motion works enterprise has been installed
--------------------------------------------

Press ENTER to exit
>>>

MWE has been deployed sucessfully and is runing at /data/mwe as user mwe!
[root@z21st-centos7-7 mwe_setup]#
```

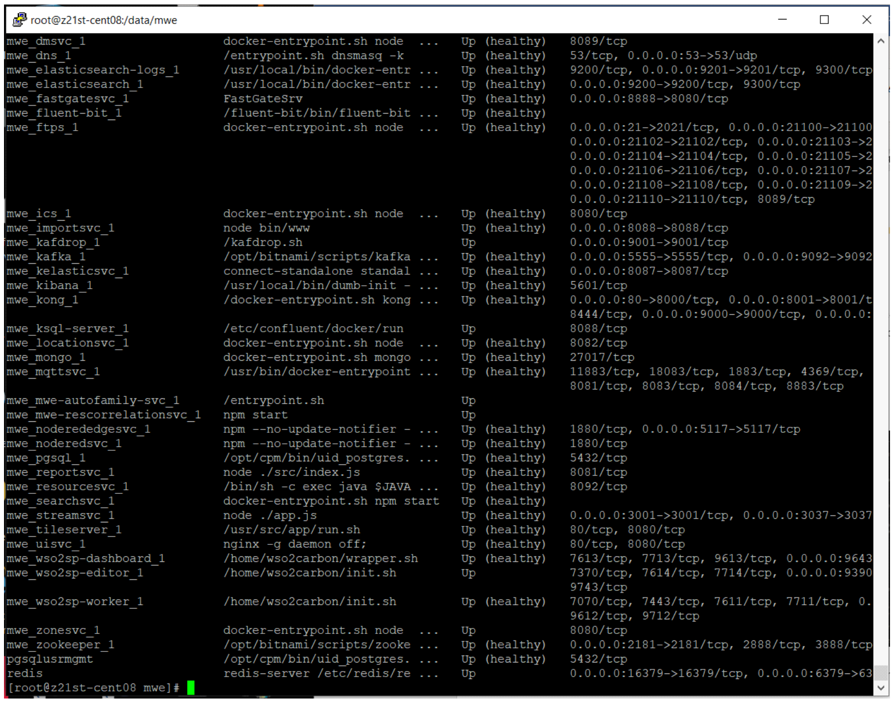
NOTE: When upgrading from 2.0.1/2.0.2 to a higher 2.0.m version, error messages may scroll in the Terminal/Putty window. Disregard these, as they do not affect installation. See Possible Error Messages During Upgrade from 2.01/2.02. If errors not included in this list appear, report them to Zebra Product Support.

Validating the Upgrade

When the upgrade is complete, check the status of the MWE services (Docker containers). Depending on you server resources (CPU and memory), it may take a couple of minutes for all services to be up and running.

1. Run the following commands.

```
# cd /data/mwe
# docker-compose ps
```



```
root@z21st-cent08:/data/mwe
mwe_dmsvc_1                  docker-entrypoint.sh node   ...   Up (healthy)   8089/tcp
mwe_dns_1                    /entrypoint.sh dnsmasq -k         Up (healthy)   53/tcp, 0.0.0.0:53->53/udp
mwe_elasticsearch-logs_1     /usr/local/bin/docker-entr ...   Up (healthy)   9200/tcp, 0.0.0.0:9201->9201/tcp, 9300/tcp
mwe_elasticsearch_1          /usr/local/bin/docker-entr ...   Up (healthy)   0.0.0.0:9200->9200/tcp, 9300/tcp
mwe_fastgatesvc_1            FastGateSrv                       Up (healthy)   0.0.0.0:8888->8080/tcp
mwe_fluent-bit_1             /fluent-bit/bin/fluent-bit ...   Up (healthy)
mwe_ftps_1                   docker-entrypoint.sh node   ...   Up (healthy)   0.0.0.0:21->2021/tcp, 0.0.0.0:21100->21100
                                                                              0.0.0.0:21102->21102/tcp, 0.0.0.0:21103->2
                                                                              0.0.0.0:21104->21104/tcp, 0.0.0.0:21105->2
                                                                              0.0.0.0:21106->21106/tcp, 0.0.0.0:21107->2
                                                                              0.0.0.0:21108->21108/tcp, 0.0.0.0:21109->2
                                                                              0.0.0.0:21110->21110/tcp, 8089/tcp
mwe_ics_1                    docker-entrypoint.sh node   ...   Up (healthy)   8080/tcp
mwe_importsvc_1              node bin/www                      Up (healthy)   0.0.0.0:8088->8088/tcp
mwe_kafdrop_1                /kafdrop.sh                       Up             0.0.0.0:9001->9001/tcp
mwe_kafka_1                  /opt/bitnami/scripts/kafka ...   Up (healthy)   0.0.0.0:5555->5555/tcp, 0.0.0.0:9092->9092
mwe_kelasticsvc_1            connect-standalone standal ...   Up (healthy)   0.0.0.0:8087->8087/tcp
mwe_kibana_1                 /usr/local/bin/dumb-init - ...   Up (healthy)   5601/tcp
mwe_kong_1                   /docker-entrypoint.sh kong ...   Up (healthy)   0.0.0.0:80->8000/tcp, 0.0.0.0:8001->8001/t
                                                                              8444/tcp, 0.0.0.0:9000->9000/tcp, 0.0.0.0:
mwe_ksql-server_1            /etc/confluent/docker/run         Up             8088/tcp
mwe_locationsvc_1            docker-entrypoint.sh node   ...   Up (healthy)   8082/tcp
mwe_mongo_1                  docker-entrypoint.sh mongo ...   Up (healthy)   27017/tcp
mwe_mqttsvc_1                /usr/bin/docker-entrypoint ...   Up (healthy)   11883/tcp, 18083/tcp, 1883/tcp, 4369/tcp,
                                                                              8081/tcp, 8083/tcp, 8084/tcp, 8883/tcp
mwe_mwe-autofamily-svc_1     /entrypoint.sh                    Up
mwe_mwe-rescorrelationsvc_1  npm start                         Up
mwe_nodereddedgesvc_1        npm --no-update-notifier - ...   Up (healthy)   1880/tcp, 0.0.0.0:5117->5117/tcp
mwe_noderedsvc_1             npm --no-update-notifier - ...   Up (healthy)   1880/tcp
mwe_pgsql_1                  /opt/cpm/bin/uid_postgres. ...   Up (healthy)   5432/tcp
mwe_reportsvc_1              node ./src/index.js               Up (healthy)   8081/tcp
mwe_resourcesvc_1            /bin/sh -c exec java $JAVA ...   Up (healthy)   8092/tcp
mwe_searchsvc_1              docker-entrypoint.sh npm start   Up (healthy)
mwe_streamsvc_1              node ./app.js                     Up (healthy)   0.0.0.0:3001->3001/tcp, 0.0.0.0:3037->3037
mwe_tileserver_1             /usr/src/app/run.sh               Up (healthy)   80/tcp, 8080/tcp
mwe_uisvc_1                  nginx -g daemon off;              Up (healthy)   80/tcp, 8080/tcp
mwe_wso2sp-dashboard_1       /home/wso2carbon/wrapper.sh       Up (healthy)   7613/tcp, 7713/tcp, 9613/tcp, 0.0.0.0:9643
mwe_wso2sp-editor_1          /home/wso2carbon/init.sh          Up             7370/tcp, 7614/tcp, 7714/tcp, 0.0.0.0:9390
                                                                              9743/tcp
mwe_wso2sp-worker_1          /home/wso2carbon/init.sh          Up (healthy)   7070/tcp, 7443/tcp, 7611/tcp, 7711/tcp, 0.
                                                                              9612/tcp, 9712/tcp
mwe_zonesvc_1                docker-entrypoint.sh node   ...   Up             8080/tcp
mwe_zookeeper_1              /opt/bitnami/scripts/zooke ...   Up (healthy)   0.0.0.0:2181->2181/tcp, 2888/tcp, 3888/tcp
pgsqlusrmgmt                 /opt/cpm/bin/uid_postgres. ...   Up (healthy)   5432/tcp
redis                        redis-server /etc/redis/re ...   Up             0.0.0.0:16379->16379/tcp, 0.0.0.0:6379->63
[root@z21st-cent08 mwe]#
```

2. Verify that the column **State** shows **Up** for all containers.

   Some services also report a health condition and should show **healthy**.

NOTE: You may see several warnings if you run the docker-compose command under the **root** account. Disregard these. If you switch to the **mwe** account, the warnings do not appear.

```
# su - mwe
# cd /data/mwe
# docker-compose ps
```

To switch back to the root account enter:

```
# su -
```

To restart all MWE services for any reason, either reboot the server or use these commands:

```
# su - mwe  (if prompted, the default password is Zebra123)
# cd /data/mwe
# ./mwe --restart
```

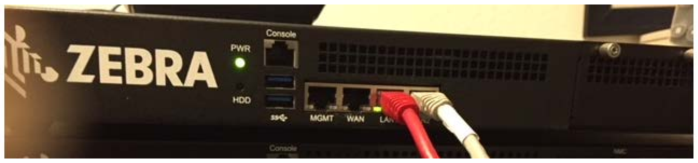
If your deployment includes a ZLA, upgrade the ZLA software to a version compatible with the MWE version. See Upgrading ZLA Software.

Reference Documents

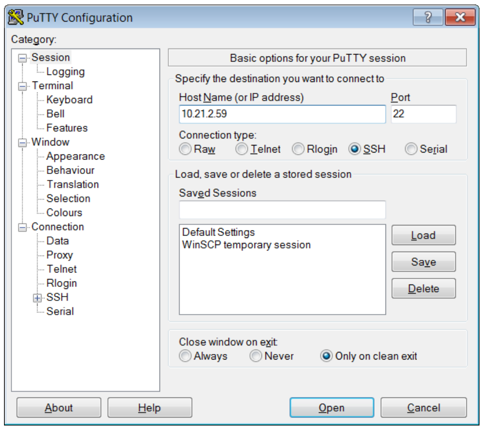
- MWE 2.0 Configuration Guide
- MWE 2.0 User Guide

Connecting to a Physical ZLA

A physical ZLA device has 4 Ethernet ports labeled MGMT, WAN, LAN1, and LAN2, providing optional connections for remote management, a WAN (Wide Area Network), and two LANs (Local Area Network) respectively.



1. To connect a laptop to a ZLA for the first time to configure network parameters, power on the ZLA and use one of the following methods:
   - The LAN1 Ethernet port on the ZLA is configured by default to acquire an IP address from a DHCP server. Connect LAN1 to the network using an Ethernet patch cable. Use Putty, WinSCP, or another SSH client to connect to the default hostname ZebraZLA. If you know the IP address assigned by the DHCP server, you can connect the SSH client to this IP rather than the ZebraZLA hostname.
   - The LAN2 Ethernet port on the ZLA is configured by default with static IP 192.168.5.1. Configure a laptop with an IP 192.168.5.x. Connect the laptop Ethernet port and LAN2 to a hub using an Ethernet patch cable, or alternatively connect the laptop Ethernet port directly to LAN2 using an Ethernet crossover cable. Use Putty, WinSCP, or another SSH client to connect to 192.168.5.1 or to the default hostname ZebraZLA.
   - Connect a USB port on the laptop to the Console port on the ZLA using an Ethernet to USB cable, or a USB to Ethernet adapter and an Ethernet cable. Then open Putty (Serial mode) or a HyperTerminal session on the laptop, and select a COM port (typically COM1 or COM3) to connect to. See the connection settings below.
2. If using Putty, select **SSH** and enter the ZLA IP address or default host name ZebraZLA.



Specify the destination you want to connect to

Host Name (or IP address): 10.21.2.59 Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

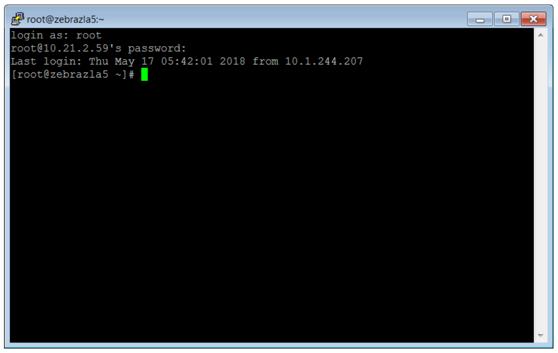
Load, save or delete a stored session

Saved Sessions

Default Settings
WinSCP temporary session

Close window on exit: Always Never [✓ Only on clean exit]

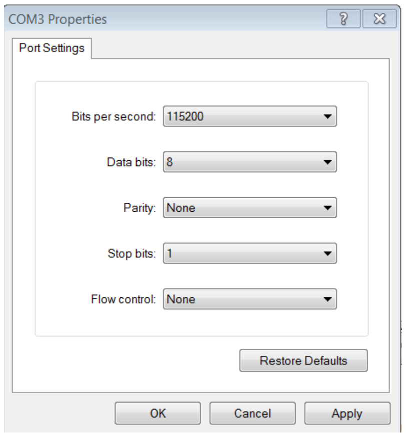
3. Click **Open**. You are prompted to enter a login account and password:



```
login as: root
root@10.21.2.59's password:
Last login: Thu May 17 05:42:01 2018 from 10.1.244.207
[root@zebrazla5 ~]#
```

You are now in a Linux command shell where you can issue commands and edit configuration files.

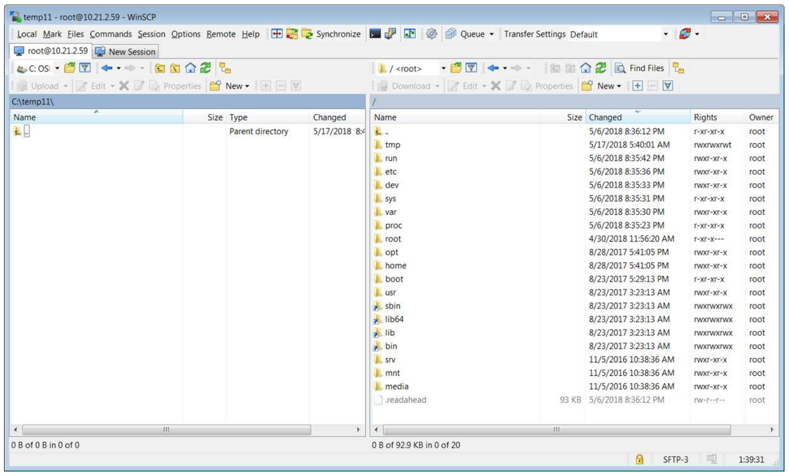
4. If you are using HyperTerminal (or Putty with Connection Type = Serial), select **COM1** or **COM3**, and use these settings.



Bits per second:	115200
Data bits:	8
Parity:	None
Stop bits:	1
Flow control:	None

This accesses a Linux command shell where you can execute commands and edit configuration files.

If using WinSCP, you have a graphical user interface similar to Windows Explorer. Double-click on text configuration files to edit them, and easily drag and drop files between the Windows laptop hosting WinSCP and the ZLA.



temp11 - root@10.21.2.59 - WinSCP
Local Mark Files Commands Session Options Remote Help
Synchronize
Queue
Transfer Settings Default
root@10.21.2.59
New Session
C: OS
Upload
Edit
Properties
New
/ <root>
Find Files
Download
SFTP-3
1:39:31

C:\temp11\

Name	Size	Type	Changed
..		Parent directory	5/17/2018 8:4

0 B of 0 B in 0 of 0

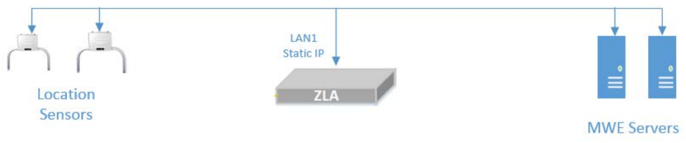
/

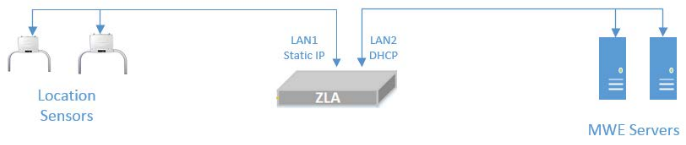
Name	Size	Changed	Rights	Owner
..		5/6/2018 8:36:12 PM	r-xr-xr-x	root
tmp		5/17/2018 5:40:01 AM	rwxrwxrwt	root
run		5/6/2018 8:35:42 PM	rwxr-xr-x	root
etc		5/6/2018 8:35:36 PM	rwxr-xr-x	root
dev		5/6/2018 8:35:33 PM	rwxr-xr-x	root
sys		5/6/2018 8:35:31 PM	r-xr-xr-x	root
var		5/6/2018 8:35:30 PM	rwxr-xr-x	root
proc		5/6/2018 8:35:23 PM	r-xr-xr-x	root
root		4/30/2018 11:56:20 AM	r-xr-x---	root
opt		8/28/2017 5:41:05 PM	rwxr-xr-x	root
home		8/28/2017 5:41:05 PM	rwxr-xr-x	root
boot		8/23/2017 5:29:13 PM	r-xr-xr-x	root
usr		8/23/2017 3:23:13 AM	rwxr-xr-x	root
sbin		8/23/2017 3:23:13 AM	rwxrwxrwx	root
lib64		8/23/2017 3:23:13 AM	rwxrwxrwx	root
lib		8/23/2017 3:23:13 AM	rwxrwxrwx	root
bin		8/23/2017 3:23:13 AM	rwxrwxrwx	root
srv		11/5/2016 10:38:36 AM	rwxr-xr-x	root
mnt		11/5/2016 10:38:36 AM	rwxr-xr-x	root
media		11/5/2016 10:38:36 AM	rwxr-xr-x	root
.readahead	93 KB	5/6/2018 8:36:12 PM	rw-r--r--	root

0 B of 92.9 KB in 0 of 20

Configuring IP Addresses

The following diagram illustrates two ways to connect a ZLA to the Location Sensors and MWE servers.

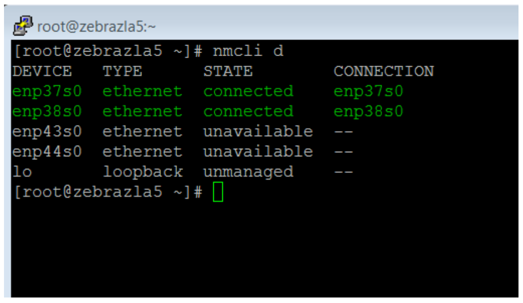




Note that the ZLA Ethernet card connecting to the sensors (LAN1 in the examples above) must be assigned a static IP address.

The ZLA operating system is CentOS Linux, so the following uses common Linux commands.

1. To view a list of Network Interface Cards (NICs) installed on the ZLA and their state, use the **`# nmcli d`** command, as in the following example:

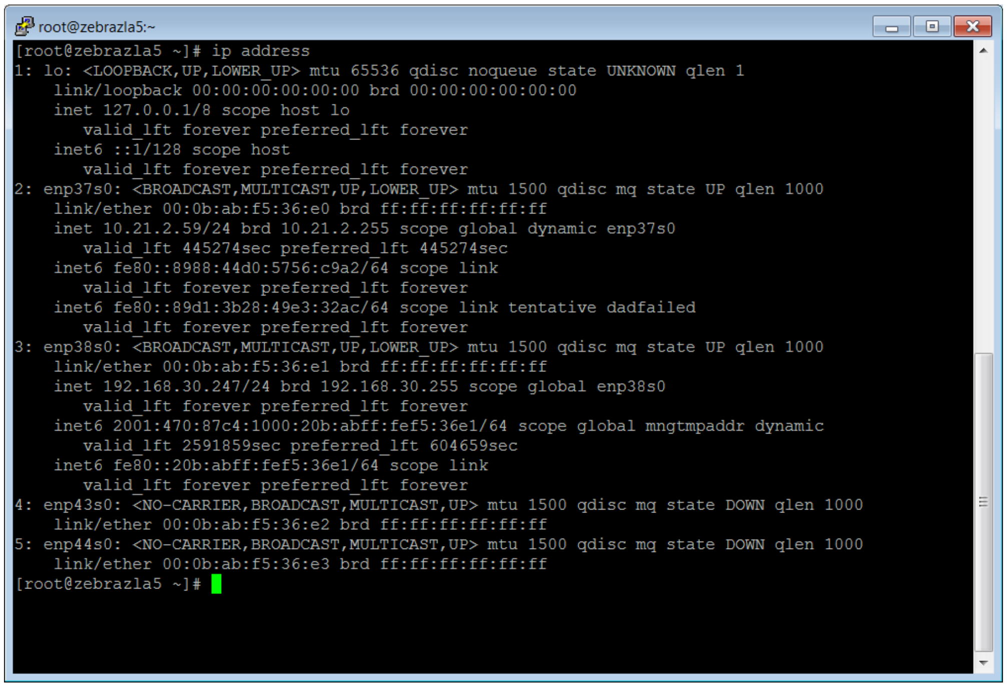


```
[root@zebrazla5 ~]# nmcli d
DEVICE    TYPE      STATE        CONNECTION
enp37s0   ethernet  connected    enp37s0
enp38s0   ethernet  connected    enp38s0
enp43s0   ethernet  unavailable  --
enp44s0   ethernet  unavailable  --
lo        loopback  unmanaged    --
[root@zebrazla5 ~]#
```

The mapping between the NICs device names assigned by Linux and the Ethernet ports on the front of the ZLA box is as follows:

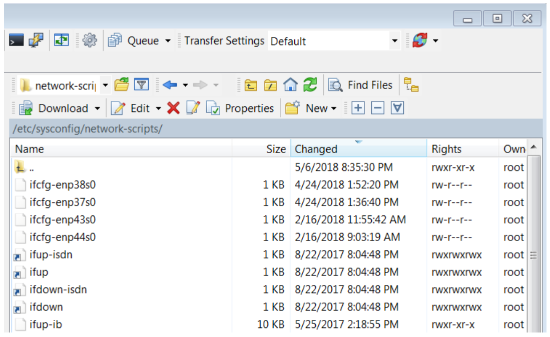
```
enp37s0:    LAN1
enp38s0:    LAN2
enp43s0:    MGMT
enp44s0:    WAN
```

2. To view the configuration of the Ethernet cards, use the **`$ ip address`** command. In the following example, the result of this command shows that the **`enp37s0`** interface uses IP 10.21.2.59 and the **`enp18s0`** interface uses IP 192.168.30.247.



```
[root@zebrazla5 ~]# ip address
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp37s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
    link/ether 00:0b:ab:f5:36:e0 brd ff:ff:ff:ff:ff:ff
    inet 10.21.2.59/24 brd 10.21.2.255 scope global dynamic enp37s0
       valid_lft 445274sec preferred_lft 445274sec
    inet6 fe80::8988:44d0:5756:c9a2/64 scope link
       valid_lft forever preferred_lft forever
    inet6 fe80::89d1:3b28:49e3:32ac/64 scope link tentative dadfailed
       valid_lft forever preferred_lft forever
3: enp38s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
    link/ether 00:0b:ab:f5:36:e1 brd ff:ff:ff:ff:ff:ff
    inet 192.168.30.247/24 brd 192.168.30.255 scope global enp38s0
       valid_lft forever preferred_lft forever
    inet6 2001:470:87c4:1000:20b:abff:fef5:36e1/64 scope global mngtmpaddr dynamic
       valid_lft 2591859sec preferred_lft 604659sec
    inet6 fe80::20b:abff:fef5:36e1/64 scope link
       valid_lft forever preferred_lft forever
4: enp43s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN qlen 1000
    link/ether 00:0b:ab:f5:36:e2 brd ff:ff:ff:ff:ff:ff
5: enp44s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc mq state DOWN qlen 1000
    link/ether 00:0b:ab:f5:36:e3 brd ff:ff:ff:ff:ff:ff
[root@zebrazla5 ~]#
```

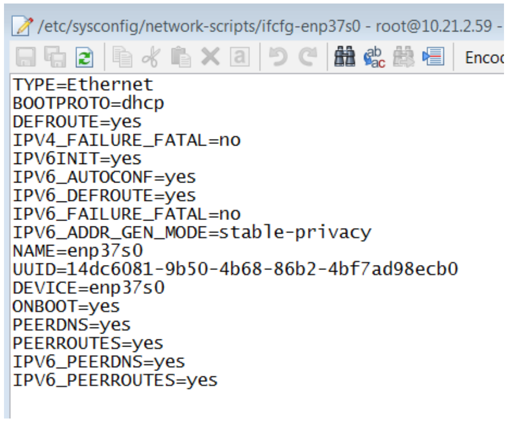
The corresponding NIC configuration files for the four NICs in the ZLA are found in the **`/etc/sysconfig/network-scripts`** directory and are named **`ifcfg-enp37s0`**, **`ifcfg-enp38s0`**, **`ifcfg-enp43s0`**, and **`ifcfg-enp44`** respectively. The following shows the four files as displayed by WinSCP.



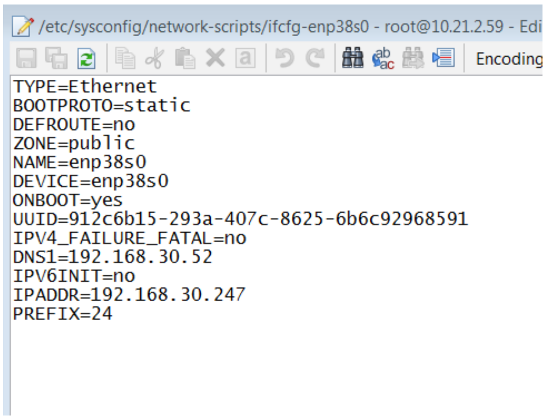
/etc/sysconfig/network-scripts/

Name	Size	Changed	Rights	Own
..		5/6/2018 8:35:30 PM	rwxr-xr-x	root
ifcfg-enp38s0	1 KB	4/24/2018 1:52:20 PM	rw-r--r--	root
ifcfg-enp37s0	1 KB	4/24/2018 1:36:40 PM	rw-r--r--	root
ifcfg-enp43s0	1 KB	2/16/2018 11:55:42 AM	rw-r--r--	root
ifcfg-enp44s0	1 KB	2/16/2018 9:03:19 AM	rw-r--r--	root
ifup-isdn	1 KB	8/22/2017 8:04:48 PM	rwxrwxrwx	root
ifup	1 KB	8/22/2017 8:04:48 PM	rwxrwxrwx	root
ifdown-isdn	1 KB	8/22/2017 8:04:48 PM	rwxrwxrwx	root
ifdown	1 KB	8/22/2017 8:04:48 PM	rwxrwxrwx	root
ifup-ib	10 KB	5/25/2017 2:18:55 PM	rwxr-xr-x	root

In this example, the contents of **`ifcfg-enp37s0`** (for LAN1 port) and **`ifcfg-enp38s0`** (for LAN2 port) are:


/etc/sysconfig/network-scripts/ifcfg-enp37s0 - root@10.21.2.59 -

```
TYPE=Ethernet
BOOTPROTO=dhcp
DEFROUTE=yes
IPV4_FAILURE_FATAL=no
IPV6INIT=yes
IPV6_AUTOCONF=yes
IPV6_DEFROUTE=yes
IPV6_FAILURE_FATAL=no
IPV6_ADDR_GEN_MODE=stable-privacy
NAME=enp37s0
UUID=14dc6081-9b50-4b68-86b2-4bf7ad98ecb0
DEVICE=enp37s0
ONBOOT=yes
PEERDNS=yes
PEERROUTES=yes
IPV6_PEERDNS=yes
IPV6_PEERROUTES=yes
```


/etc/sysconfig/network-scripts/ifcfg-enp38s0 - root@10.21.2.59 - Edi

```
TYPE=Ethernet
BOOTPROTO=static
DEFROUTE=no
ZONE=public
NAME=enp38s0
DEVICE=enp38s0
ONBOOT=yes
UUID=912c6b15-293a-407c-8625-6b6c92968591
IPV4_FAILURE_FATAL=no
DNS1=192.168.30.52
IPV6INIT=no
IPADDR=192.168.30.247
PREFIX=24
```

NOTE: **`ifcfg-enp37s0`** is configured for dhcp while **`ifcfg-enp18s0`** is configured with a static IP.

3. To modify the NIC configuration edit these files using the Linux **`vi`** command, the WinSCP tool, or another command or tool of your choice.

Applying Changes

After modifying the NIC's configuration files, use the following command to restart the network service to implement the changes.

```
# systemctl restart network
```

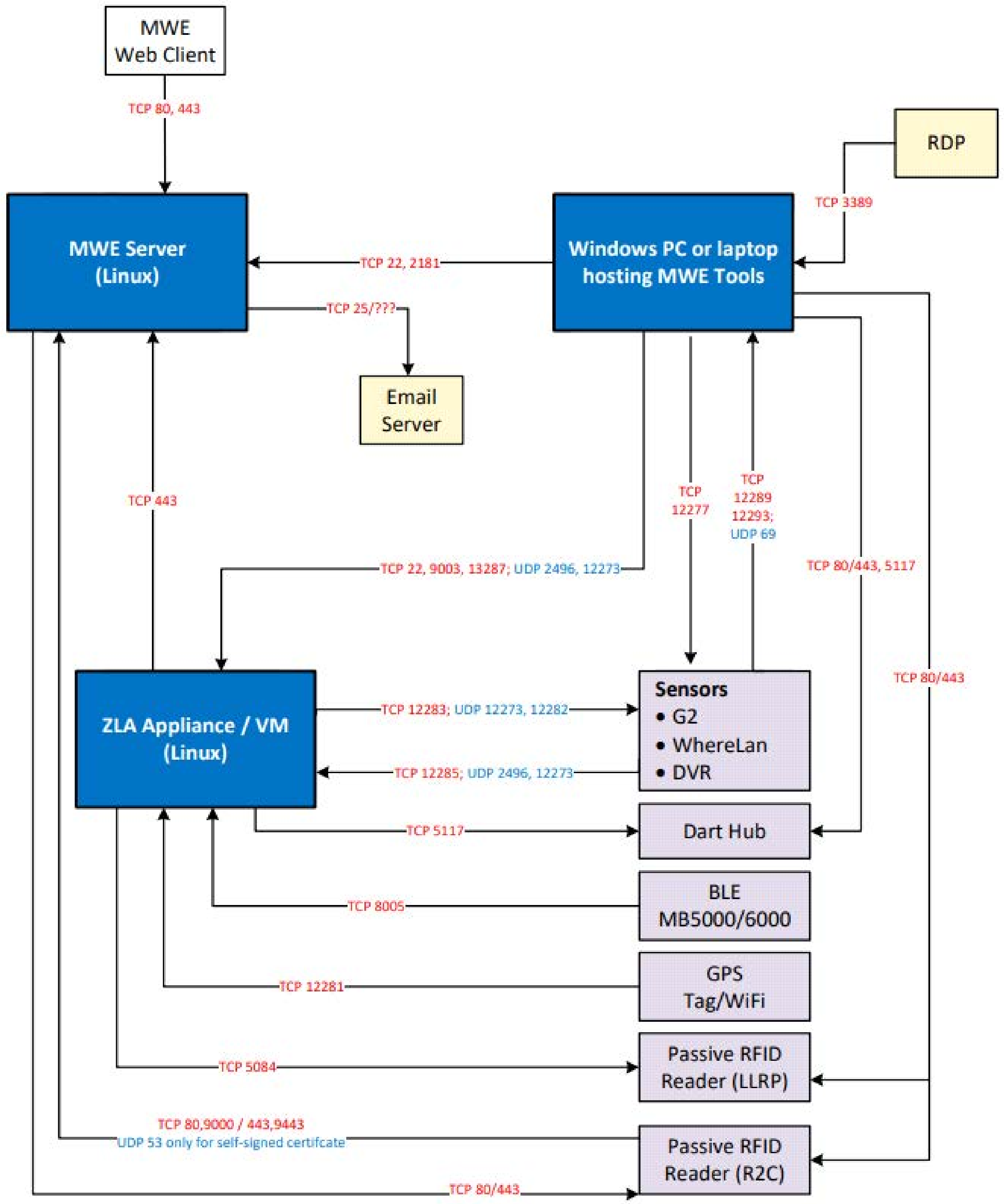
Possible Error Messages During Upgrade from 2.01/2.02

When upgrading from 2.0.1/2.0.2 to a higher 2.0.x, the following error messages may scroll in the Terminal/Putty window. Disregard these, as they do not affect installation. If other errors not included in the list appear, report them to Zebra Product Support.

```
ERROR:  could not open extension control file "/usr/local/share/postgresql/extension/pgaudit.control": No such file or directory

ERROR:  extension "pgaudit" does not exist

ERROR:  must be owner of extension pgcrypto

ERROR:  permission denied to create extension "pg_stat_statements"

ERROR:  extension "pg_stat_statements" does not exist

ERROR:  could not open extension control file "/usr/local/share/postgresql/extension/pgaudit.control": No such file or directory

ERROR:  extension "pgaudit" does not exist

ERROR:  must be owner of extension pgcrypto

Error: No such container: mwe_wso2sp-worker_1

ERROR: Failed to check mongo version: MongoDB not running. Will attempt to upgrade from 4.0 to 4.4.

error while removing network: network mwe_default id a4157ef19739f5babeaad83b07f705f825fcb4796269689388a3030dd52e7382 has active endpoints

Error response from daemon: Container 2c8707588725aa5a553e8c4fc54fbb6e3bc3436b172d5c6dfc833ab21cf2bfee is not running

error while removing network: network mwe_default id a4157ef19739f5babeaad83b07f705f825fcb4796269689388a3030dd52e7382 has active endpoints

Error response from daemon: No such container: temporal_postgres

Error: No such container: temporal_postgres
```

MWE Network Block Diagram

The following diagram shows ports that must be open in an MWE 2.0.x deployment. See Ports for more details.


MWE Web Client
TCP 80, 443
RDP
TCP 3389
MWE Server (Linux)
TCP 22, 2181
Windows PC or laptop hosting MWE Tools
TCP 25/???
Email Server
TCP 443
TCP 12277
TCP 12289 12293; UDP 69
TCP 22, 9003, 13287; UDP 2496, 12273
TCP 80/443, 5117
TCP 80/443
ZLA Appliance / VM (Linux)
TCP 12283; UDP 12273, 12282
TCP 12285; UDP 2496, 12273
Sensors
• G2
• WhereLan
• DVR
TCP 5117
Dart Hub
TCP 8005
BLE MB5000/6000
TCP 12281
GPS Tag/WiFi
TCP 5084
Passive RFID Reader (LLRP)
TCP 80,9000 / 443,9443
UDP 53 only for self-signed certifcate
Passive RFID Reader (R2C)
TCP 80/443