**Running Dialogic® PowerMedia™ XMS in a Docker Container**

# Introduction

Dialogic’s PowerMedia XMS can run successfully on many Virtual Machines, Hypervisors and Cloud Environments, including, with only a few modifications, in a Docker container.

Docker[[1]](#footnote-1) is an open platform for developers and sysadmins to build, ship, and run distributed applications. Consisting of Docker Engine, a portable, lightweight runtime and packaging tool, and Docker Hub, a cloud service for sharing applications and automating workflows, Docker enables apps to be quickly assembled from components and eliminates the friction between development, QA, and production environments. As a result, IT can ship fast and run the same app, unchanged, on laptops, data center VMs, and any cloud.

This tech note outlines how to install Docker on a CentOS host, and then quickly get an XMS Docker image up and running. Additional information is also available on building your own XMS Docker image and storing it in your Docker Hub repository.

A set of scripts to facilitate working with XMS on Docker accompany this tech note. Docked commands can be quite complex. Hence, these scripts should simplify many commands by supplying the proper options to be used with XMS containers.

**Support for PowerMedia XMS use in Docker container environments is available for POCs, development activities, and trials, with limited support and not intended for production usage. Users interested in Docker container support for PowerMedia XMS should contact their Dialogic Sales Representative for further information. A complete list of virtual and cloud platforms supported for production use on XMS 3.1 are listed** [**here**](http://www.dialogic.com/webhelp/XMS/3.1/XMS_InstallConfig.pdf)**.**

# Some Considerations in Running XMS in a Docker Container

As a media server, XMS is highly dependent on a system’s ability to accurately schedule and perform certain operations related to processing of media streams - both incoming and outgoing - that arrive in UDP/RTP packets. The Docker daemon, by default, does not make any allowances for its containers obtaining a slice of the entire system’s CPU time reserved for real time processing. When XMS starts up, it requests this slice, and the request must be granted before XMS fully starts.

To simplify this process, a script is supplied with this tech note to handle XMS container startup. It divides the 950,000 microseconds out of each second of real time CPU available on the system among all XMS containers running on the host. For example, two XMS containers would get 475,000 microseconds each, three would get 316,666 each, etc. This does not preclude other containers from requesting real time CPU; however, adjustments would have to be made in the script to avoid requesting more than the available total of 950,000 microseconds.

The container itself cannot request these adjustments, as this would violate container security. The request must be done by a process on the host system with appropriate privileges. There is an open [Docker Issue](https://github.com/docker/docker/issues/13983) on the subject. If the enhancement is implemented, it should make it easier to adjust the rt\_runtime\_us values for each container using the Docker CLI. But it will not change the basic premise that real time CPU slices are granted from the Docker host, rather than requested by individual containers.

This tech note also assumes that the Docker host is a bare metal system. While Docker will run on a virtual machine, there would be an amount of real time CPU available to the VMs on the base system, and this would be further split among XMS containers on a single VM. Depending on system load, from the media requirements on the XMS container, to the needs on other VMs, there may be a point where XMS will not be able to keep up with real time demands of a media server – encoding/decoding media and moving RTP packets in and out of the system on a timely basis.

A second consideration is the type of CentOS 7 image used for an XMS container. A base image available from the Docker repository is centos:centos7, a minimal CentOS 7.X. But before using it as the base for an XMS image, some modifications would need to be done. Centos7 for Docker does not use “systemd”, but rather uses a stub systemd package to fulfill dependencies. XMS requires the full systemd due to its reliance on system services. [This](https://github.com/docker-library/docs/tree/master/centos) explains systemd use in containers further.

A third is container port usage. XMS can use a large number of UDP ports for audio and video media streams. Each port exposed in each container results in a proxy process that maps the single container port to a unique host port. Using the full range of XMS media ports could result in excessive processes running. For this tech note, only ten UDP ports for audio and video are set, a number which should be more than adequate for the 4-port verification license provided with XMS.

# Docker Installation

Before an XMS Docker image can be downloaded and used, Docker itself must be installed on the host system. Almost any operating system will support Docker containers; however, a non-Linux OS must do so by running the container in a virtual Linux machine on the host OS. The Docker host for this tech note was installed on CentOS 7. An installation on Ubuntu or other Linux distribution should support a CentOS container, but these have not been tested with XMS, which runs on CentOS. Thus, using CentOS 7 for the Docker host is recommended. A CentOS 7 virtual machine (run on VBox, KVM, Xen, etc.) may be used as the Docker host.

As of May 2016, versions used for this tech note were:

* CentOS 7.2.1511
* XMS release 3.1.12402
* Docker version 1.11.1

The following Docker URLs will be useful for this tech note:

* [Docker Install Instructions for CentOS](https://docs.docker.com/installation/centos/)
* [Docker User Guide](https://docs.docker.com/userguide/dockerhub/) – this tech note should contain all that is needed; however consult this URL for further reference
* [Docker Hub](https://hub.docker.com/) – public container storage

All operations should be done as root, or as a sudo user.

Follow the [Install Instructions](https://docs.docker.com/installation/centos/), using the six steps in the “Install with yum” section. Do not forget to do the “yum update” once CentOS has been installed in order to get the latest Docker version.

Once Docker installation is complete, untar the XMS Docker package that accompanies this tech note in a convenient location on the Docker system:

>tar xvfz docker-xms.tgz

# Running the Docker Daemon

Before running any Docker commands, make sure the Docker daemon is running:

> systemctl start docker

# IP Addresses and Ports in XMS Containers

Containers are started with a network device of eth0 and lo. The eth0 device is assigned an IP address in the 172.17.0.X subnet. When a container is started on the host system, a bridged interface, docker0, is created. Next, a routing rule is added to the system that directs the packets for a Docker container to the docker0 interface:

Kernel IP routing table

default via 192.168.1.1 dev em2 proto static metric 100

172.17.0.0/16 dev docker0 proto kernel scope link src 172.17.0.1

192.168.1.0/24 dev em2 proto kernel scope link src 192.168.1.20 metric 100

Since Docker containers are on their own address/subnet, there must be a way of directing SIP/WebRTC signaling and media to the container. There are two ways to do this.

The simpler of the two is using a direct port mapping from the host address/port to the container address on the same port. For example a SIP port would look like this:

0.0.0.0:5060->5060/udp

Here, UDP port 5060 for all interfaces on the host is mapped to the same port, 5060, on the container. A SIP call can be made directly to the Docker host’s IP address and it will be forwarded to the same port on the XMS container. While this is a comparatively simpler technique, it has the drawback of only allowing a **single** XMS container to be started on the Docker host. If a virtual machine is used as the Docker host, some additional configuration of XMS will be needed for media to be routed correctly. In the XMS Web GUI, Network screen, NAT Configuration tab, the Behind NAT button must be selected, and the IP address of the virtual Docker host set in the Public IP address field. XMS must then be restarted using the System screen, Services tab, Restart button.

Alternately, it is possible to let Docker take care of port selection on the host, and multiple XMS containers, all using the same set of ports within each container, can be mapped to different ports on the host. Thus, multiple containers can be run, but they must be referenced by their 172.17.0.0 address.

To do this, a static route can be added to the router servicing the LAN on which the Docker Host resides. Or, this could be done on the system running the Web browser or SIP phone. The route would direct all packets with a destination of 172.17.0.0 to the Docker host system, where they would be routed through the bridged interface to the container subnet. The static route for a Docker Host with an IP address of 192.168.1.101 would look like this:

Destination IP Address 172.17.0.0

IP Subnet Mask 255.255.255.0

Gateway IP Address 192.168.1.101

Metric 10

Once the XMS container is addressable, it is ready for any further configuration using the XMS Web GUI and to accept calls.

# Running an XMS Container under Docker

Docker images, including an XMS image, are available from [Docker Hub](https://hub.docker.com/). In order to use Docker Hub, sign up for a (free) Docker account there.

Scripts in the docker-xms directory are provided with this tech note to simplify Docker operations. Here is the XMS startup sequence:

* Log in to Docker using your Docker Hub account. This will allow you to download the XMS container.
	+ > docker login
	+ Enter the requested information
* Start an XMS container. Options are as follows:

usage: ./run\_xms\_container.sh [-c] -i <image>

-c Use container address instead of host for access

-i Full docker image name

* + - –c will direct Docker to do automatic port mapping and allow for multiple containers. Each will be an independent XMS installation with its own IP address. **All containers must be started with the run\_xms\_container.sh script and use the same <docker-image-name>.**
		- Otherwise, a direct port mapping from container to host will be used and only a single container may be run.
		- -i <image> must be given. The image in the Docker Hub that goes with this tech note is jhermanski/xms:latest.
	+ Note that if there is not a local image with the specified name, Docker will search Docker Hub for the image and it will be downloaded from there.
	+ Be patient; initial download of public XMS image from Docker Hub to your local repository will take some time – perhaps 5+ minutes, depending on network speed.
	+ The container will start in the background. The XMS startup may take several minutes, so again, be patient. To verify that the system is up, use the XMS WebGUI (see below) and go to the System/Services tab to check the state of all services. All except faxservice and cdrserver should be running.
	+ The IP address of the container will be given once the script has finished starting it. In addition, instructions for making a WebRTC test call and a link to XMS documentation will be seen when the script completes and exits.
* Additional XMS information:
	+ Login for the XMS WebGUI is superadmin/admin
	+ ssh login to XMS container:
		- Log in as root with password “powermedia”
		- Port 22 if allowing Docker to map ports
		- Port 2222 if 1 to 1 port mapping with Docker host. 2222 will not interfere with the normal port 22 ssh into the host.
	+ Direct terminal access to the container can also be done using the nsenter.sh script. It requires the container ID. For example:

>docker ps

>./nsenter.sh c23b7f2a3871

* + Full documentation for XMS (version 3.1, May 2016) can be found [here.](https://www.dialogic.com/en/manuals/xms/xms3.1.aspx)

# Stopping/Removing an XMS Container

When done with the XMS container, stop it by first finding its ID:

>docker ps

Then use the ID to stop the container. For example:

>docker stop c23b7f2a3871

A stopped container may be restarted. To entirely stop and remove the container from Docker, use:

>docker rm -f c23b7f2a3871

# Building a Docker Container with XMS

When a pre-built container from a Docker repository is used – as this tech note has described - building a container is not necessary. However, if any major changes are made such as an OS or XMS upgrade, it may be desirable to build a new XMS container. To do this, a Dockerfile is used.

The Dockerfile is a sequence of commands that puts together a specific Docker image. Two good references may be found [here](https://docs.docker.com/reference/builder) and [here](https://docs.docker.com/articles/dockerfile_best-practices). Below is an outline of the process. Consult the Dockerfiles supplied with this tech note for specifics.

Begin by building a CentOS 7 image that uses systemd. This uses a separate Dockerfile invoked by:

>./build\_c7-systemd.sh

The resulting image will be stored locally and then used for the XMS image. Create a local XMS image with:

>./build\_xms.sh username/xms:latest

“username” is your Docker login name, “xms” is an online repository name. “latest” is a tag for this particular container. This is a good format to use if you intend to upload your image to an online repository. (see below) Otherwise it is generally not important. The following is done to create an XMS image:

* Yum install a number of packages that are needed for the XMS install.
* Add XMS RPMs, scripts and other files in the build directory to the image.
	+ The XMS and JRE installation packages are not included with this tech note due to their size. But they should be downloaded indo the docker-xms directory.
		- JRE 8 RPM downloads are available [here](http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html). jre-8u65-linux-x64.rpm was used for this tech note.
		- XMS 3.1 downloads are available [here.](http://www.dialogic.com/en/products/media-server-software/xms/xms-download.aspx) dialogic\_xms\_3.1.12402-0.c7-release.tgz was used for this tech note.
* Install a Java Runtime Environment used by XMS for communication with the Dialogic® PowerMedia™ Media Resource Broker. (MRB) The PowerMedia MRB is not used in the configuration described in this tech note.
* Untar the XMS RPM file and do the XMS RPM install. Note that by default, XMS is started when the install is done. This will generate a number of errors that may be disregarded. When XMS is started as the container is run, these errors will not be seen.
* Set an environment variable so that, on startup, XMS will use a kernel-based real time clock.
* Expose XMS ports needed at runtime. If more than ten media ports are needed, the Dockerfile and the script run\_xms\_container.sh may be edited to add additional ports.

# Compressing an XMS Container

Once built, an XMS container can be quite large. This should not be an issue in a local repository, especially if additional work will be done to it. Intermediate versions of the containers are used to speed the rebuild process, saving building all containers from scratch.

But if it is uploaded to the Docker Hub and then used for other downloads, a more compact container is beneficial. The following script may be used to “flatten” the container:

> flatten\_xms\_image.sh <original-docker-image-name> <flattened-docker-image-name>

The new image should be given a different name to distinguish it from the un-flattened image.

# Uploading an XMS Container to a Docker Repository

This interaction with the Docker repositories requires a Docker user name and login. One may be obtained [here](https://hub.docker.com/).

* Log into [Docker Hub](https://hub.docker.com/) using your registered username. One private repository is allowed. Name it “xms”.
* When building an image for upload, the local repository has to be named “username/xms”. Image tag can be anything. But convention seems to lean toward a “latest” in each repository. So for jhermanski, XMS would be jhermanski/xms:latest.
* Once you have an image to be saved, use the following Docker commands:
	+ Docker Host > docker login – provide username, password and email
	+ Docker Host > docker push username/xms
* XMS images are quite large, so be patient. The Docker Hub push will supply a “done” message.
* When the push is completed, verify that it was successful by going to the Docker Hub website and checking your xms repository.

# Further Docker Topics

Follow-on Docker tech notes for XMS are planned to include such topics as:

* XMS Docker performance considerations.
* Using XMS with the Dialogic® PowerMedia™ Media Resource Broker (MRB) in Docker containers.
* Running XMS Docker containers with the Restcomm-Docker Application Server.
* Running XMS Docker on Amazon AWS.



Copyright © 2016 Dialogic Corporation. All Rights Reserved. You may not reproduce this document in whole or in part without permission in writing from Dialogic Corporation at the address provided below.

All contents of this document are furnished for informational use only and are subject to change without notice and do not represent a commitment on the part of Dialogic Corporation and its affiliates or subsidiaries (“Dialogic”). Reasonable effort is made to ensure the accuracy of the information contained in the document. However, Dialogic does not warrant the accuracy of this information and cannot accept responsibility for errors, inaccuracies or omissions that may be contained in this document.

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH DIALOGIC® PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN A SIGNED AGREEMENT BETWEEN YOU AND DIALOGIC, DIALOGIC ASSUMES NO LIABILITY WHATSOEVER, AND DIALOGIC DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF DIALOGIC PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHT OF A THIRD PARTY.

Dialogic products are not intended for use in certain safety-affecting situations. Please see <http://www.dialogic.com/company/terms-of-use.aspx> for more details.

Due to differing national regulations and approval requirements, certain Dialogic products may be suitable for use only in specific countries, and thus may not function properly in other countries. You are responsible for ensuring that your use of such products occurs only in the countries where such use is suitable. For information on specific products, contact Dialogic Corporation at the address indicated below or on the web at *www.dialogic.com*.

It is possible that the use or implementation of any one of the concepts, applications, or ideas described in this document, in marketing collateral produced by or on web pages maintained by Dialogic may infringe one or more patents or other intellectual property rights owned by third parties. Dialogic does not provide any intellectual property licenses with the sale of Dialogic products other than a license to use such product in accordance with intellectual property owned or validly licensed by Dialogic and no such licenses are provided except pursuant to a signed agreement with Dialogic. More detailed information about such intellectual property is available from Dialogic’s legal department at 6700 Cote-de-Liesse Road, Suite 100, Borough of Saint-Laurent, Montreal, Quebec, Canada H4T 2B5. **Dialogic encourages all users of its products to procure all necessary intellectual property licenses required to implement any concepts or applications and does not condone or encourage any intellectual property infringement and disclaims any responsibility related thereto. These intellectual property licenses may differ from country to country and it is the responsibility of those who develop the concepts or applications to be aware of and comply with different national license requirements.**

Dialogic, Dialogic Pro, Dialogic Blue, Veraz, Brooktrout, Diva, BorderNet, PowerMedia, PowerVille, PowerNova, MSaaS, ControlSwitch, I-Gate, Mobile Experience Matters, Network Fuel, Video is the New Voice, Making Innovation Thrive, Diastar, Cantata, TruFax, SwitchKit, Eiconcard, NMS Communications, SIPcontrol, Exnet, EXS, Vision, inCloud9, NaturalAccess and Shiva, among others as well as related logos, are either registered trademarks or trademarks of Dialogic Corporation and its affiliates or subsidiaries. Dialogic's trademarks may be used publicly only with permission from Dialogic. Such permission may only be granted by Dialogic’s legal department at 6700 Cote-de-Liesse Road, Suite 100, Borough of Saint-Laurent, Montreal, Quebec, Canada H4T 2B5. Any authorized use of Dialogic's trademarks will be subject to full respect of the trademark guidelines published by Dialogic from time to time and any use of Dialogic’s trademarks requires proper acknowledgement.

The names of actual companies and products mentioned herein are the trademarks of their respective owners.

This document discusses one or more open source products, systems and/or releases. Dialogic is not responsible for your decision to use open source in connection with Dialogic products (including without limitation those referred to herein), nor is Dialogic responsible for any present or future effects such usage might have, including without limitation effects on your products, your business, or your intellectual property rights.

1. All references to Docker, as well as to other third party materials and web-sites, within this tech note have been verified by Dialogic as of May 2016; however, Dialogic is not responsible for updates or changes that may subsequently occur. [↑](#footnote-ref-1)