Informatica®

Informatica® Address Verification
5.16.4

Installation and Getting Started Guide (On-Premises)

Informatica Address Verification Installation and Getting Started Guide (On-Premises)
5.16.4
March 2024

© Copyright Informatica LLC 1993, 2024

This software and documentation are provided only under a separate license agreement containing restrictions on use and disclosure. No part of this document may be reproduced or transmitted in any form, by any means (electronic, photocopying, recording or otherwise) without prior consent of Informatica LLC.

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, the use, duplication, disclosure, modification, and adaptation is subject to the restrictions and license terms set forth in the applicable Government contract, and, to the extent applicable by the terms of the Government contract, the additional rights set forth in FAR 52.227-19, Commercial Computer Software License.

Informatica, AddressDoctor, the Informatica logo, and any other Informatica-owned trademarks that appear in the document are trademarks or registered trademarks of Informatica LLC in the United States and many jurisdictions throughout the world. A current list of Informatica trademarks is available on the web at https://www.informatica.com/trademarks.html. Other company and product names may be trade names or trademarks of their respective owners.

Portions of this software and/or documentation are subject to copyright held by third parties, including without limitation: Copyright DataDirect Technologies. All rights reserved. Copyright © Sun Microsystems. All rights reserved. Copyright © RSA Security Inc. All Rights Reserved. Copyright © Ordinal Technology Corp. All rights reserved. Copyright © Aandacht c.v. All rights reserved. Copyright Genivia, Inc. All rights reserved. Copyright Isomorphic Software. All rights reserved. Copyright © Meta Integration Technology, Inc. All rights reserved. Copyright © Intalio. All rights reserved. Copyright © Oracle. All rights reserved. Copyright © Adobe Systems Incorporated. All rights reserved. Copyright © DataArt, Inc. All rights reserved. Copyright © ComponentSource. All rights reserved. Copyright © Microsoft Corporation. All rights reserved. Copyright © Rogue Wave Software, Inc. All rights reserved. Copyright © Teradata Corporation. All rights reserved. Copyright © Yahoo! Inc. All rights reserved. Copyright © Glyph & Cog, LLC. All rights reserved. Copyright © Thinkmap, Inc. All rights reserved. Copyright © Clearpace Software Limited. All rights reserved. Copyright © Information Builders, Inc. All rights reserved. Copyright © OSS Nokalva, Inc. All rights reserved. Copyright Edifecs, Inc. All rights reserved. Copyright Cleo Communications, Inc. All rights reserved. Copyright © International Organization for Standardization 1986. All rights reserved. Copyright © ej-technologies GmbH. All rights reserved. Copyright © Jaspersoft Corporation. All rights reserved. Copyright © International Business Machines Corporation. All rights reserved. Copyright © yWorks GmbH. All rights reserved. Copyright © Lucent Technologies. All rights reserved. Copyright © University of Toronto. All rights reserved. Copyright © Daniel Veillard. All rights reserved. Copyright © Unicode, Inc. Copyright IBM Corp. All rights reserved. Copyright © MicroQuill Software Publishing, Inc. All rights reserved. Copyright © PassMark Software Pty Ltd. All rights reserved. Copyright © LogiXML, Inc. All rights reserved. Copyright © 2003-2010 Lorenzi Davide, All rights reserved. Copyright © Red Hat, Inc. All rights reserved. Copyright © The Board of Trustees of the Leland Stanford Junior University. All rights reserved. Copyright © EMC Corporation. All rights reserved. Copyright © Flexera Software. All rights reserved. Copyright © Jinfonet Software. All rights reserved. Copyright © Apple Inc. All rights reserved. Copyright © Telerik Inc. All rights reserved. Copyright © BEA Systems. All rights reserved. Copyright © PDFlib GmbH. All rights reserved. Copyright © Orientation in Objects GmbH. All rights reserved. Copyright © Tanuki Software, Ltd. All rights reserved. Copyright © Ricebridge. All rights reserved. Copyright © Sencha, Inc. All rights reserved. Copyright © Scalable Systems, Inc. All rights reserved. Copyright © jQWidgets. All rights reserved. Copyright © Tableau Software, Inc. All rights reserved. Copyright© MaxMind, Inc. All Rights Reserved. Copyright © TMate Software s.r.o. All rights reserved. Copyright © MapR Technologies Inc. All rights reserved. Copyright © Amazon Corporate LLC. All rights reserved. Copyright © Highsoft. All rights reserved. Copyright © Python Software Foundation. All rights reserved. Copyright © BeOpen.com. All rights reserved. Copyright © CNRI. All rights reserved.

This product includes software developed by the Apache Software Foundation (http://www.apache.org/), and/or other software which is licensed under various versions of the Apache License (the "License"). You may obtain a copy of these Licenses at http://www.apache.org/licenses/. Unless required by applicable law or agreed to in writing, software distributed under these Licenses is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the Licenses for the specific language governing permissions and limitations under the Licenses.

This product includes software which was developed by Mozilla (http://www.mozilla.org/), software copyright The JBoss Group, LLC, all rights reserved; software copyright © 1999-2006 by Bruno Lowagie and Paulo Soares and other software which is licensed under various versions of the GNU Lesser General Public License Agreement, which may be found at http:// www.gnu.org/licenses/lgpl.html. The materials are provided free of charge by Informatica, "as-is", without warranty of any kind, either express or implied, including but not limited to the implied warranties of merchantability and fitness for a particular purpose.

The product includes ACE(TM) and TAO(TM) software copyrighted by Douglas C. Schmidt and his research group at Washington University, University of California, Irvine, and Vanderbilt University, Copyright (©) 1993-2006, all rights reserved.

This product includes software developed by the OpenSSL Project for use in the OpenSSL Toolkit (copyright The OpenSSL Project. All Rights Reserved) and redistribution of this software is subject to terms available at http://www.openssl.org and http://www.openssl.org/source/license.html.

This product includes Curl software which is Copyright 1996-2013, Daniel Stenberg, <daniel@haxx.se>. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at http://curl.haxx.se/docs/copyright.html. Permission to use, copy, modify, and distribute this software for any purpose with or without fee is hereby granted, provided that the above copyright notice and this permission notice appear in all copies.

The product includes software copyright 2001-2005 (©) MetaStuff, Ltd. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at http://www.dom4j.org/ license.html.

The product includes software copyright © 2004-2007, The Dojo Foundation. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at http://dojotoolkit.org/license.

This product includes ICU software which is copyright International Business Machines Corporation and others. All rights reserved. Permissions and limitations regarding this software are subject to terms available at http://source.icu-project.org/repos/icu/icu/trunk/license.html.

This product includes software copyright © 1996-2006 Per Bothner. All rights reserved. Your right to use such materials is set forth in the license which may be found at http:// www.gnu.org/software/ kawa/Software-License.html.

This product includes OSSP UUID software which is Copyright © 2002 Ralf S. Engelschall, Copyright © 2002 The OSSP Project Copyright © 2002 Cable & Wireless Deutschland. Permissions and limitations regarding this software are subject to terms available at http://www.opensource.org/licenses/mit-license.php.

This product includes software developed by Boost (http://www.boost.org/) or under the Boost software license. Permissions and limitations regarding this software are subject to terms available at http:/ /www.boost.org/LICENSE_1_0.txt.

This product includes software copyright © 1997-2007 University of Cambridge. Permissions and limitations regarding this software are subject to terms available at http:// www.pcre.org/license.txt.

This product includes software copyright © 2007 The Eclipse Foundation. All Rights Reserved. Permissions and limitations regarding this software are subject to terms available at http:// www.eclipse.org/org/documents/epl-v10.php and at http://www.eclipse.org/org/documents/edl-v10.php.

This product includes software licensed under the terms at http://www.tcl.tk/software/tcltk/license.html, http://www.bosrup.com/web/overlib/?License, http://www.stlport.org/doc/ license.html, http://asm.ow2.org/license.html, http://www.cryptix.org/LICENSE.TXT, http://hsqldb.org/web/hsqlLicense.html, http://httpunit.sourceforge.net/doc/ license.html, http://jung.sourceforge.net/license.txt , http://www.gzip.org/zlib/zlib_license.html, http://www.openldap.org/software/release/license.html, http://www.libssh2.org, http://slf4j.org/license.html, http://www.sente.ch/software/OpenSourceLicense.html, http://fusesource.com/downloads/license-agreements/fuse-message-broker-v-5-3- license-agreement; http://antlr.org/license.html; http://aopalliance.sourceforge.net/; http://www.bouncycastle.org/licence.html; http://www.jgraph.com/jgraphdownload.html; http://www.jcraft.com/jsch/LICENSE.txt; http://jotm.objectweb.org/bsd_license.html; . http://www.w3.org/

Consortium/Legal/2002/copyright-software-20021231; http://www.slf4j.org/license.html; http://nanoxml.sourceforge.net/orig/copyright.html; http://www.json.org/license.html; http://forge.ow2.org/projects/javaservice/, http://www.postgresql.org/about/licence.html, http://www.sqlite.org/copyright.html, http://www.tcl.tk/software/tcltk/license.html, http://www.jaxen.org/faq.html, http://www.jdom.org/docs/faq.html, http://www.slf4j.org/license.html; http://www.iodbc.org/dataspace/iodbc/wiki/iODBC/License; http://www.keplerproject.org/md5/license.html; http://www.toedter.com/en/jcalendar/license.html; http://www.edankert.com/bounce/index.html; http://www.net-snmp.org/about/license.html; http://www.openmdx.org/#FAQ; http://www.php.net/license/3_01.txt; http://srp.stanford.edu/license.txt; http://www.schneier.com/blowfish.html; http://www.jmock.org/license.html; http://xsom.java.net; http://benalman.com/about/license/; https://github.com/CreateJS/EaselJS/blob/master/src/easeljs/display/Bitmap.js; http://www.h2database.com/html/license.html#summary; http://jsoncpp.sourceforge.net/LICENSE; http://jdbc.postgresql.org/license.html; http://protobuf.googlecode.com/svn/trunk/src/google/protobuf/descriptor.proto; https://github.com/rantav/hector/blob/master/LICENSE; http://web.mit.edu/Kerberos/krb5-current/doc/mitK5license.html; http://jibx.sourceforge.net/jibx-license.html; https://github.com/lyokato/libgeohash/blob/master/LICENSE; https://github.com/hjiang/jsonxx/blob/master/LICENSE; https://code.google.com/p/lz4/; https://github.com/jedisct1/libsodium/blob/master/LICENSE; http://one-jar.sourceforge.net/index.php?page=documents&file=license; https://github.com/EsotericSoftware/kryo/blob/master/license.txt; http://www.scala-lang.org/license.html; https://github.com/tinkerpop/blueprints/blob/master/LICENSE.txt; http://gee.cs.oswego.edu/dl/classes/EDU/oswego/cs/dl/util/concurrent/intro.html; https://aws.amazon.com/asl/; https://github.com/twbs/bootstrap/blob/master/LICENSE; https://sourceforge.net/p/xmlunit/code/HEAD/tree/trunk/LICENSE.txt; https://github.com/documentcloud/underscore-contrib/blob/master/LICENSE, and https://github.com/apache/hbase/blob/master/LICENSE.txt.

This product includes software licensed under the Academic Free License (http://www.opensource.org/licenses/afl-3.0.php), the Common Development and Distribution License (http://www.opensource.org/licenses/cddl1.php) the Common Public License (http://www.opensource.org/licenses/cpl1.0.php), the Sun Binary Code License Agreement Supplemental License Terms, the BSD License (http:// www.opensource.org/licenses/bsd-license.php), the new BSD License (http://opensource.org/licenses/BSD-3-Clause), the MIT License (http://www.opensource.org/licenses/mit-license.php), the Artistic License (http://www.opensource.org/licenses/artistic-license-1.0) and the Initial Developer's Public License Version 1.0 (http://www.firebirdsql.org/en/initial-developer-s-public-license-version-1-0/).

This product includes software copyright © 2003-2006 Joe Walnes, 2006-2007 XStream Committers. All rights reserved. Permissions and limitations regarding this software are subject to terms available at http://xstream.codehaus.org/license.html. This product includes software developed by the Indiana University Extreme! Lab. For further information please visit http://www.extreme.indiana.edu/.

This product includes software Copyright (c) 2013 Frank Balluffi and Markus Moeller. All rights reserved. Permissions and limitations regarding this software are subject to terms of the MIT license.

See patents at https://www.informatica.com/legal/patents.html.

DISCLAIMER: Informatica LLC provides this documentation "as is" without warranty of any kind, either express or implied, including, but not limited to, the implied warranties of noninfringement, merchantability, or use for a particular purpose. Informatica LLC does not warrant that this software or documentation is error free. The information provided in this software or documentation may include technical inaccuracies or typographical errors. The information in this software and documentation is subject to change at any time without notice.

NOTICES

This Informatica product (the "Software") includes certain drivers (the "DataDirect Drivers") from DataDirect Technologies, an operating company of Progress Software Corporation ("DataDirect") which are subject to the following terms and conditions:

1. THE DATADIRECT DRIVERS ARE PROVIDED "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT.
2. IN NO EVENT WILL DATADIRECT OR ITS THIRD PARTY SUPPLIERS BE LIABLE TO THE END-USER CUSTOMER FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, CONSEQUENTIAL OR OTHER DAMAGES ARISING OUT OF THE USE OF THE ODBC DRIVERS, WHETHER OR NOT INFORMED OF THE POSSIBILITIES OF DAMAGES IN ADVANCE. THESE LIMITATIONS APPLY TO ALL CAUSES OF ACTION, INCLUDING, WITHOUT LIMITATION, BREACH OF CONTRACT, BREACH OF WARRANTY, NEGLIGENCE, STRICT LIABILITY, MISREPRESENTATION AND OTHER TORTS.

The information in this documentation is subject to change without notice. If you find any problems in this documentation, report them to us at infa_documentation@informatica.com.

Informatica products are warranted according to the terms and conditions of the agreements under which they are provided. INFORMATICA PROVIDES THE INFORMATION IN THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING WITHOUT ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND ANY WARRANTY OR CONDITION OF NON-INFRINGEMENT.

Publication Date: 2024-03-18

Table of Contents

Preface. 5
Informatica Resources. 5
Informatica Network. 5
Informatica Knowledge Base. 5
Informatica Documentation. 5
Informatica Product Availability Matrices. 5
Informatica Velocity. 6
Informatica Marketplace. 6
Informatica Global Customer Support. 6

Chapter 1: Overview. 7
Informatica Address Verification (On-Premises) Overview. 7

Chapter 2: Database Download Management. 8
Informatica AddressDoctor Database Download Manager Overview. 8
Further Information. 8
Reference Databases and Packages. 9

Chapter 3: System Requirements. 11
Supported Platforms and System Requirements. 11
Memory Management. 12

Chapter 4: Installation. 14
Understanding Informatica Address Verification Software Packages. 14
Installing the Informatica Address Verification C based Package 15
Installing the Informatica Address Verification Java based Package. 16

Chapter 5: Initialization. 17
Initializing Informatica Address Verification. 17
Example: Initializing Informatica Address Verification from a C Based Implementation. 18
Example: Initializing Informatica Address Verification from a Java Based Implementation. 20

Chapter 6: Demonstration Applications. 23
Demonstration Applications. 23
AddressCheck. 23
ConsoleDemo. 24

Preface

Read the *Informatica Address Verification (On-Premises) Installation and Getting Started Guide* to learn how to install and initialize Informatica Address Verification (On-Premises), also known as Informatica AddressDoctor®.

Informatica Resources

Informatica provides you with a range of product resources through the Informatica Network and other online portals. Use the resources to get the most from your Informatica products and solutions and to learn from other Informatica users and subject matter experts.

Informatica Network

The Informatica Network is the gateway to many resources, including the Informatica Knowledge Base and Informatica Global Customer Support. To enter the Informatica Network, visit https://network.informatica.com.

As an Informatica Network member, you have the following options:

- Search the Knowledge Base for product resources.
- View product availability information.
- Create and review your support cases.
- Find your local Informatica User Group Network and collaborate with your peers.

Informatica Knowledge Base

Use the Informatica Knowledge Base to find product resources such as how-to articles, best practices, video tutorials, and answers to frequently asked questions.

To search the Knowledge Base, visit https://search.informatica.com. If you have questions, comments, or ideas about the Knowledge Base, contact the Informatica Knowledge Base team at KB_Feedback@informatica.com.

Informatica Documentation

Use the Informatica Documentation Portal to explore an extensive library of documentation for current and recent product releases. To explore the Documentation Portal, visit https://docs.informatica.com.

If you have questions, comments, or ideas about the product documentation, contact the Informatica Documentation team at infa_documentation@informatica.com.

Informatica Product Availability Matrices

Product Availability Matrices (PAMs) indicate the versions of the operating systems, databases, and types of data sources and targets that a product release supports. You can browse the Informatica PAMs at https://network.informatica.com/community/informatica-network/product-availability-matrices.

Informatica Velocity

Informatica Velocity is a collection of tips and best practices developed by Informatica Professional Services and based on real-world experiences from hundreds of data management projects. Informatica Velocity represents the collective knowledge of Informatica consultants who work with organizations around the world to plan, develop, deploy, and maintain successful data management solutions.

You can find Informatica Velocity resources at http://velocity.informatica.com. If you have questions, comments, or ideas about Informatica Velocity, contact Informatica Professional Services at ips@informatica.com.

Informatica Marketplace

The Informatica Marketplace is a forum where you can find solutions that extend and enhance your Informatica implementations. Leverage any of the hundreds of solutions from Informatica developers and partners on the Marketplace to improve your productivity and speed up time to implementation on your projects. You can find the Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

You can contact a Global Support Center by telephone or through the Informatica Network.

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
https://www.informatica.com/services-and-training/customer-success-services/contact-us.html.

To find online support resources on the Informatica Network, visit https://network.informatica.com and select the eSupport option.

CHAPTER 1

Overview

This chapter includes the following topic:

- Informatica Address Verification (On-Premises) Overview, 7

Informatica Address Verification (On-Premises) Overview

Informatica Address Verification (On-Premises) verifies addresses from more than 240 countries and territories quickly, efficiently, and effectively.

The address verification workflow includes steps to transliterate, parse, format, validate, and enrich addresses. You can use the integrated transliteration capabilities to validate addresses that you capture in different writing systems and languages. Address Verification also standardizes and formats the address data to meet the requirements that the local postal authorities specify.

Postal certification improves the quality of addresses and ensures that Address Verification services meet postal authority requirements.

Address Verification complies with the following postal certifications:

- Address Matching Approval System (AMAS) certification for Australia Post
- Coding Accuracy Support System (CASS) certification for the United States Postal Services
- SendRight certification for New Zealand Post
- Service National de L'Adresse (SNA) certification for La Poste of France
- Software Evaluation and Recognition Program (SERP) certification for Canada Post

About This Document

Informatica offers Address Verification as an on-premises solution and as a cloud-based solution. Both solutions use a common engine. The current document describes the Informatica Address Verification (On-Premises) solution. References to Informatica Address Verification in the current document refer to Informatica Address Verification (On-Premises).

CHAPTER 2

Database Download Management

This chapter includes the following topics:

- Informatica AddressDoctor Database Download Manager Overview, 8
- Reference Databases and Packages, 9

Informatica AddressDoctor Database Download Manager Overview

Use Informatica AddressDoctor Database Download Manager to select and download the reference databases that enable on-premises address verification. Database Download Manager is available on the web and as a Java-based client application.

The Database Download Manager web page and client application are integrated with the Informatica Passport single sign-on solution. You can download the client application from the web page.

You can use your Informatica Passport user profile to seamlessly access Informatica tools and portals, including Database Download Manager and the Data Quality Center. If you do not have a user profile, create a profile and assign one or more Data Quality Center accounts to the profile before you use Database Download Manager.

When you install the client application, you can download the reference databases through the client user interface or on the command line. Database Download Manager lists all of the reference databases that AddressDoctor supports. You can filter the list to view the databases that the Data Quality Center accounts associated with the current profile are authorized to download. You obtain download rights for reference databases when you buy a subscription to the reference data.

Further Information

For more information about Informatica ADM, see the *Informatica Address Verification Database Download Manager User Guide*.

Reference Databases and Packages

Informatica reference databases are proprietary-format database files that contain reference addresses from countries and territories that Informatica supports for address verification. The reference databases are read-only and platform-independent.

The database file names have the following format:

```
DB5_XXXYZZZZ_NN_YYMMDD.zip
```

You can interpret the file names in the following way:

- `XXX` denotes the ISO 3166-1 alpha 3 code for the country or territory to which the reference data applies. For example, `USA` for the United States.
- `Y` denotes the primary product version. For example, `5`.
- `ZZZZ` denotes the reference database type. For example, `BI` for Batch and Interactive and `GCAP` for arrival point geocoding.
- `NN` denotes the iteration of the file. Most files have the iteration number 01. Some countries use iteration numbers to identify files with additional data.

You download the files in compressed (zip) format. The zip file names follow a similar naming convention to the database file names. The zip file names also include the day, month, and year on which Informatica created the file.

Address Verification provides the following types of reference database:

BI

Batch and interactive data.

FC

Fast completion data.

Cx

Certified data. The x character represents a number. For example, `DB5_CAN5C1_01_150201.zip` contains a SERP-certified reference address database for Canada.

AC

Address code lookup data.

GC

Standard geocoding data.

GCAP

Arrival point geocoding data.

CA

CAMEO data.

Ex

Supplementary data. The x character represents a number. For example, `DB5_FRA5E1_01_181101.zip` contains the supplementary reference address database for France address enrichments.

You might download multiple reference address databases for a country.

Most countries use a single version of each database file. However, a country might have multiple version of a file. An ID number such as `01`, `02`, or `03`, in the zip package name identifies the version of the database file.

For example, the reference data zip files that you can download for Australia can include the two-digit codes _01 and _02 in the file names. Select the _02 version of the file to download reference data that includes sub-building data. Select the _01 version of the file to download reference data without sub-building data.

Unlock Codes

You need valid unlock codes for the database that you download.

The following table describes the unlock code types that the databases require:

Unlock Code Type	Description
VALIDATION	Unlock code for batch and interactive and fast completion databases.
GEO_STANDARD	Unlock code for standard geocoding databases.
GEO_ARRIVAL_POINT	Unlock code for arrival point geocoding databases.
CAMEO	Unlock code for CAMEO databases.
ADDRESS_CODE_LOOKUP	Unlock code for address code lookup databases.
SINGLE_LINE_VALIDATION	Unlock code for single line address validation feature.
SUPPLEMENTARY	Unlock code for supplementary databases that contain information about country-specific enrichments.

CHAPTER 3

System Requirements

This chapter includes the following topics:

- Supported Platforms and System Requirements, 11
- Memory Management, 12

Supported Platforms and System Requirements

Informatica Address Verification is supported on a number of hardware and software platforms. The system resource requirements for Address Verification vary according to your requirements.

Supported Platforms

Address Verification is developed using the C++ programming language. Address Verification provides different software packages to suit the hardware and software environment in which you install Address Verification. The Address Verification software packages contain C and Java based APIs.

Note: Informatica Address Verification documentation contains examples based on the C and Java interface of Address Verification. You can model Address Verification implementations for other languages, such as C++, C#, VB.Net, PHP, Perl, Ruby, and Python, on these examples. Informatica provides technical support for the C and Java APIs. Informatica does not provide implementation-specific support.

You must install a Java Development Kit on the machine that hosts the Address Verification engine.

If you install Address Verification on an AIX machine, install IBM Java 8 or later. If you install Address Verification on a Solaris machine, install Oracle Java SE 8 or later. If you install on other platforms, install OpenJDK version 8.

You can install Address Verification on machines with the following configurations:

Operating System	Processor Architecture
Windows Server 2019	x64 (64-bit)
Windows Server 2016	x64 (64-bit)
SUSE Linux Enterprise Server 15	x64 (64-bit)
SUSE Linux Enterprise Server 12	x64 (64-bit)
RedHat Enterprise Linux 8	x64 (64-bit)

Operating System	Processor Architecture
RedHat Enterprise Linux 7	x64 (64-bit)
AIX 7	POWER (64-bit)

System Requirements

Address Verification is designed to be highly efficient in its memory and resource usage. To ensure best possible performance, install Address Verification on a device that has fast input and output systems and sufficient memory.

The device on which you install Address Verification should have a minimum of 512 MB RAM.

Before you finalize the memory requirements, consider the size of the reference address databases that you require. The complete set of worldwide postal reference databases, including supplementary databases for address enrichments, consumes approximately 55 GB of storage space.

Preloading databases into memory significantly improves the performance of Address Verification. The device on which you install Address Verification must have sufficient RAM to preload the databases that you require.

As the total size of the worldwide databases is approximately 55 GB, the RAM required to preload all databases and perform address processing is approximately 60 GB. For smaller file sizes, less RAM is required. For example, the worldwide batch and interactive databases consume approximately 9 GB of space, and the geocoding databases consume an additional 11 GB. To preload and run the worldwide batch, interactive, and geocoding databases, 20 GB to 25 GB of RAM is necessary.

Tip: If fully preloading databases is not an option, use solid-state drives to store the reference address databases. Solid-state drives are faster than hard-disk drives and can significantly improve performance, especially when multithreading is used.

You set the database preloading method in the `SetConfig.xml` file. For more information on database preload settings, see the *Address Verification On-Premises Developer Guide*.

Memory Management

Informatica Address Verification stores different types of objects, such as address objects, pre-loaded reference address databases, and caches, in its memory. When you make memory allocations for Address Verification, you must consider the different objects that have specific memory requirements.

You can divide the memory requirements of Address Verification into the following blocks:

- General memory block. Allocated for general management functions. Typically, the general memory block size is 7 MB.
- Thread memory block. Allocated for address processing and verification routines. Address Verification creates as many thread memory blocks as the number of simultaneous threads you configure it to process.
  The size of a thread memory block is about 48 MB for 64-bit systems.
- Address object memory block. Allocated for storing the address objects defined. Address Verification creates as many address object memory blocks as the number of address objects you configure it to process.

The size of an address object memory block is about 4.8 MB + (0.24 MB x the value set for `MaxResultCount`) in the case of 64-bit systems.

- Memory block reserved for caching. Address Verification reserves one cache memory block for each of the verification or processing threads.
- Memory blocks for preloading reference address databases. The value for preloading memory block depends on the number and size of the databases that you want to preload.
- Unallocated memory block.

You can configure the `MaxMemoryUsageMB` parameter to specify the maximum available memory for Address Verification. The values you set for the following parameters control the memory allocation for the memory blocks:

- `MaxThreadCount`. The maximum number of threads that Address Verification can process simultaneously. The value set for this parameter controls the number of thread memory blocks and the total memory allocation for the thread blocks.
- `MaxAddressObjectCount`. The maximum number of address objects that Address Verification can store. You can set a value that is two or three times the value you configured for `MaxThreadCount`. The value you set for `MaxAddressObjectCount` controls the number of address object memory blocks and the total memory allocation for the address objects.
- `CacheSize`. The memory reserved for caching purposes. If you set `CacheSize` to `None`, Address Verification does not allocate any memory for caching. If you set `CacheSize` to `Small`, Address Verification allocates 0.4 MB of cache memory block for each of the threads. If you set `CacheSize` to `Large`, Address Verification allocates 0.75 MB of cache memory block for each of the threads. For example, if `MaxThreadCount` is set to 4 and `CacheSize` to `Small`, Address Verification allocates 1.6 MB for cache memory block.

CHAPTER 4

Installation

This chapter includes the following topics:

- Understanding Informatica Address Verification Software Packages, 14
- Installing the Informatica Address Verification C based Package , 15
- Installing the Informatica Address Verification Java based Package, 16

Understanding Informatica Address Verification Software Packages

Informatica Address Verification is available in multiple software packages that suit different implementation requirements. You can identify the right package for your server environment by decoding the filenames of the software packages. You can download the package that suits the server environment in which you install Address Verification.

The Address Verification software packages come in compressed, .zip, packages. A typical Address Verification software package file name follows the `AD5_PPP_64_YYMMDD_(X.Y.ZZ.ZZZZ).zip` format. You can interpret the file name in the following way:

- `AD5`. Address Verification major version (5).
- `PPP`. The operating system on which you can install the package. For example, AIX or WIN.

  The file name contains one of the following operating system abbreviations:

  AIX

  AIX

  ZLN_SUSE

  Linux on z systems and Suse

  RHT_SUSE

  Linux. Red Hat and Suse

  SOS

  Solaris SPARC

  WIN

  Windows
- `64`. The package is for 64-bit systems.

- `YYMMDD`. The build date. For example, 141103 for a build generated on the 3rd of November 2014.
- `X.Y.ZZ.ZZZZ`. Version and build number. For example, 5.6.0.30153.

For example, `AD5_SOS_64_141103_(5.6.0.30153).zip` indicates that the package contains Address Verification version 5.6.0 minor build 30153 fort the 64-bit Solaris SPARC operating system. The date part in the file name indicates that the package was created on November 3, 2014.

Sometimes, the package name also contains the compiler information between the minor build number and the file extension. For example, in the file name `AD5_SOS_64_141103_(5.6.0.30153)-sun.studio.11.zip` sun.studio.11 indicates that the package contains APIs for Solaris SPARC 64-bit system compiled using Sun Studio 11.

Installing the Informatica Address Verification C based Package

1. Extract the files and folders from the Informatica Address Verification software package to the device on which you install Address Verification. When you extract the items, do not alter the directory structure.

   The following folders are created on the device:

   - `bin`. Contains executable sample applications such as ConsoleDemo and AddressCheck.
   - `etc`. Contains the XML configuration file examples that you need to modify the default behavior of the ConsoleDemo application.
   - `include`. Contains the header files that are required for the application to function.
   - `lib`. Contains the `.dll` (for Windows) or `.so` (for Unix) file.
   - `src`. Contains the sample application code.
2. Based on the platform on which you install Address Verification, copy the `.dll` or `.so` file to the shared library path of your device.

   To check the shared library path of a Windows device, run the `echo %path%` command from the command prompt.

   To check the shared library path on a Unix device, enter the `echo $LD_LIBRARY_PATH` command.

   Note: If you have previous versions of Address Verification running on the device on which you install the new version, remove the older versions from the system before you unpack the new files. If you need to retain the older versions, ensure that the folders are distinctly named and that the configuration files are available in appropriate locations.
3. If you install Address Verification on a Unix based device, increase the thread stack size to at least 1 MB.

   For example, on AIX devices, set the `export AIXTHREAD_STK` to 1000000.

   Similarly, set the `ulimit -s` to unlimited.

Installing the Informatica Address Verification Java based Package

To use the Java-based version of Informatica Address Verification, you must install a Java Development Kit on the machine that hosts the Address Verification engine. If you want to develop your own applications, you must install a Java platform on the machine.

If you install Address Verification on an AIX machine, install IBM Java 8 or later. If you install Address Verification on a Solaris machine, install Oracle Java SE 8 or later. If you install on other platforms, install OpenJDK version 8.

1. To install the Java-based package on a Windows device, copy `AddressDoctor5.dll` and `AddressDoctor5.jar` to the JRE class path.

   Typically, `C:\Program Files\Java\jre\lib\ext` is saved to the system-wide class path. You can explicitly set application-specific class paths using the `-cp` switch.

   You can run the following commands after unpacking the files to the present working directory:

   ```
   java -Xss2m -cp bin;lib/AddressDoctor5.jar -Djava.library.path=lib ConsoleDemoJava
   ```

2. To install the Java-based package on a Unix-based system, copy `AddressDoctor5.jar` and `AddressDoctor5.so` to the JRE class path.

   Typically, `/usr/j2se/jre/lib/ext` is saved to the system-wide class path. You can explicitly set application-specific class paths using the `-cp` switch.

   After unpacking the files to the present working directory, run the following commands:

   ```
   java -Xss2m -cp bin:lib/AddressDoctor5.jar -Djava.library.path=lib ConsoleDemoJava
   ```

   Note: If you have previous versions of Address Verification installed on the device on which you want to install the new version, remove the older versions from the system before you unpack the new files. If you need to retain the older versions, be sure to distinctly name the folders and ensure that the configuration files are available in appropriate locations.

3. Set the thread stack size to a minimum of 2 MB and the heap size to 512 MB.

   For example, if Address Verification is installed in the `lib` folder on a Linux device on which the main class is named **MyApp**, run the following commands:

   ```
   java -cp .:lib/AddressDoctor5.jar MyApp.java

   java -Xss2m -Xms512m -cp .:lib/AddressDoctor5.jar

   -Djava.library.path=lib MyApp
   ```

4. If you install Informatica Address Verification on a Unix-based device, increase the thread stack size to at least 1 MB.

   For example, on AIX devices, set `export AIXTHREAD_STK` to 1000000.

   Similarly, set `ulimit -s` to unlimited.

   On IBM J9 JVM, increase the OS stack size by running `java -Xmso2m`.

CHAPTER 5

Initialization

This chapter includes the following topics:

- Initializing Informatica Address Verification, 17
- Example: Initializing Informatica Address Verification from a C Based Implementation, 18
- Example: Initializing Informatica Address Verification from a Java Based Implementation, 20

Initializing Informatica Address Verification

Before you can start processing addresses using Informatica Address Verification, you must initialize Address Verification by calling the `AD_Initialize()` function. For `AD_Initialize()` to succeed, you must include a valid unlock code and the path to the corresponding reference address database in the `setconfig.xml` file. After you finish processing the addresses, call `AD_DeInitialize()` to deinitialize Address Verification.

Note: This topic shows the C function calls. If you are using Java or other implementations of Address Verification, use the corresponding function calls.

Prerequisites

Before you call `AD_Initialize()`, complete the following prerequisites:

- Install Address Verification.
- Download reference address databases and extract the contents to a location on the device on which you install Address Verification.
- Update `SetConfig.xml` to include at least one valid unlock code and the path to the corresponding reference address database.
- If you are using the Java API, set the `Encoding` and `WriteXMLEncoding` attributes of Input and Result elements in `Parameters.xml` and `SetConfig.xml` to UTF-16.

Calling Initialization and Deinitialization Functions

Call the Address Verification initialization and deinitialization functions in the following sequence:

1. To initialize Address Verification, call `AD_Initialize()`. `AD_Initialize()` verifies the settings and configures Address Verification accordingly. You can call other functions such as `AD_GetAddressObject()` only after `AD_Initialize()` finishes successfully. .
2. To release `AddressObjects` after you complete address processing, call `AD_ReleaseAddressObject()`.

3. To deinitialize Address Verification, call `AD_DeInitialize()`. You can call `AD_DeInitialize()` only after you call `AD_ReleaseAddressObject()`.
   After the successful completion of `AD_DeInitialize()`, Address Verification is ready to be initialized again.

Rules and Guidelines for Initialization and Deinitialization

Consider the following rules and guidelines when you initialize and deinitialize Address Verification to process an address:

- When you call AD_Initialize in C or AddressDoctor.initialize in Java, the time to initialize depends on the number of files to open and the level of preloading involved.

  For example, if you have BATCH_INTERACTIVE reference data files and a worldwide license, initialization might take 30 seconds. If you have a complete set of reference data files, including FASTCOMPLETION, GEOCODING, and other files, initialization might take one minute.

  The sample times refer to a fast CPU and local solid state disk storage. The complete data file size is approximately 40 GB.

- Due to the time to initialize, and because the processing time for a single address is in the millisecond range, Informatica strongly recommends that you initialize the engine when the customer program starts up. Also, errors and warnings can occur when you initialize the engine that do not occur later at other times. Deinitialize the engine when the customer program shuts down.

Example: Initializing Informatica Address Verification from a C Based Implementation

Before you can start processing addresses using Informatica Address Verification, you must call `AD_Initialize()` to initialize Address Verification. After you initialize Address Verification, you can call other functions such as `AD_GetAddressObject()`. After you complete address processing, call `AD_ReleaseAddressObject()` and `AD_DeInitialize()` to release the address object and close the Address Verification process.

The following example contains sample C code that contains `AD_Initialize()`, `AD_GetAddressObject()`, `AD_ReleaseAddressObject()`, and `AD_DeInitialize()` function calls. The sample code also contains one Singapore address that you can validate to verify that the Address Verification implementation is working correctly.

```
AD_AOHandle hAOHandle;
   char sResultXML[ 16 * 1024 ];
AD_Initialize(
   "<?xml version='1.0' encoding='iso-8859-1' ?>\n"
   "<!DOCTYPE SetConfig SYSTEM 'SetConfig.dtd'>\n"
   "<SetConfig>\n"
   "<General />\n"
   "<UnlockCode>(Enter Code here)</UnlockCode>\n"
   "<DataBase CountryISO3='ALL' Type='BATCH_INTERACTIVE'
   Path='/ADDB' PreloadingType='NONE'/>\n"
   "</SetConfig>\n",
   NULL,
   NULL,
   NULL
);
AD_GetAddressObject( &hAOHandle );
AD_SetInputDataXML( hAOHandle,
   "<?xml version='1.0' encoding='ISO-8859-1'?>\n"
   "<!DOCTYPE InputData SYSTEM 'InputData.dtd'>\n"
```

```
    "<InputData>\n"
    "<AddressElements>\n"
    "<Country Item='1' Type='NAME'>SGP</Country>\n"
    "<Locality Item='1' Type='COMPLETE'>Singapore</Locality>\n"
    "<PostalCode Item='1' Type='FORMATTED'>048624</PostalCode>\n"
    "<Street Item='1' Type='COMPLETE'>Raffles Place</Street>\n"
    "<Number Item='1' Type='COMPLETE'>80</Number>\n"
    "<Building Item='1' Type='COMPLETE'>#50-01 UOB Plaza 1</Building>\n"
    "<Organization Item='1' Type='NAME'>AddressDoctor GmbH</Organization>\n"
    "</AddressElements>\n"
    "</InputData>\n"
  );
  AD_Process( hAOHandle );
  AD_GetResultXML( hAOHandle, sResultXML, sizeof( sResultXML ) );
  AD_ReleaseAddressObject( hAOHandle );
  AD_DeInitialize();
```

Address Verification returns the processed output in the format defined in `Result.dtd`. For more information about the XML format used for results, see the `Result.dtd` file. The following example shows the results generated for the `AD_GetResultXML()` call in the first code sample shown in the preceding example:

```
<?xml version="1.0" encoding="UTF-16"?>

<Result
  ProcessStatus="V2"
     ModeUsed="BATCH"
     Count="1"
     CountOverflow="NO"
     CountryISO3="SGP"
     PreferredScript="DATABASE"
     PreferredLanguage="DATABASE">

<ResultData
        ResultNumber="1"
     MailabilityScore="4"
     ResultPercentage="100.00"
     ElementResultStatus="F0F000F0F000404440E0"
     ElementInputStatus="60600060600020222060"
     ElementRelevance="10100010100000000010">
    <AddressElements>
        <Country Type="NAME_EN" Item="1">SINGAPORE</Country>
        <Locality Item="1">SINGAPORE</Locality>
        <PostalCode Item="1">048624</PostalCode>
        <Street Item="1">RAFFLES PLACE</Street>
        <Number Item="1">80</Number>
        <Building Item="1">UOB PLAZA 1</Building>
        <SubBuilding Item="1"># 50</SubBuilding>
        <SubBuilding Item="2">01</SubBuilding>
        <Organization Item="1">ADDRESSDOCTOR GMBH</Organization>
    </AddressElements>
    <AddressLines>
        <RecipientLine Line="1">ADDRESSDOCTOR GMBH</RecipientLine>
        <DeliveryAddressLine Line="1">80 RAFFLES PLACE</DeliveryAddressLine>
        <DeliveryAddressLine Line="2">#50-01 UOB PLAZA 1</DeliveryAddressLine>
        <CountrySpecificLocalityLine Line="1">SINGAPORE 048624</
CountrySpecificLocalityLine>
        <FormattedAddressLine Line="1">ADDRESSDOCTOR GMBH</FormattedAddressLine>
        <FormattedAddressLine Line="2">80 RAFFLES PLACE</FormattedAddressLine>
        <FormattedAddressLine Line="3">#50-01 UOB PLAZA 1</FormattedAddressLine>
        <FormattedAddressLine Line="4">SINGAPORE 048624</FormattedAddressLine>
    </AddressLines>
    <AddressComplete>ADDRESSDOCTOR GMBH
80 RAFFLES PLACE
#50-01 UOB PLAZA 1
SINGAPORE 048624
</AddressComplete>
</ResultData>
</Result>
```

After you successfully test the Address Verification implementation, you can compile your application. The procedure for compiling the application varies greatly among different platforms and compilers.

The following example shows the command for compiling the ConsoleDemo C++ application, which comes in the `src` folder of the download package, on a Linux device by using the GNU compiler collection (gcc) compiler:

```
gcc -Iinclude -Llib -lAddressDoctor5 -lpthread -o bin/ConsoleDemo src/ConsoleDemo.cpp
```

Example: Initializing Informatica Address Verification from a Java Based Implementation

Before you can start processing addresses using Address Verification in a Java based implementation, you must initialize Address Verification by calling `AddressDoctor.initialize()`. After you initialize Address Verification, you can call other functions such as `AddressDoctor.getAddressObject()`. After you complete address processing, call `AddressDoctor.releaseAddressObject()` and `AddressDoctor.deinitialize()` to release the address object and close the Address Verification process, respectively.

The following example shows sample Java code that contains `AddressDoctor.initialize()`, `AddressDoctor.getAddressObject()`, `AddressDoctor.releaseAddressObject()`, and `AddressDoctor.deinitialize()` function calls. The sample code also contains one Singapore address to validate.

```
    private static AddressObject m_oAO;

    public static void main(String[] args) {
      int iLastError = 0;
      String sResultXML = "";

      try
      {
         AddressDoctor.initialize(
               "<?xml version='1.0' encoding='UTF-16' ?>"+
               "<!DOCTYPE SetConfig SYSTEM 'SetConfig.dtd'>"+
               "<SetConfig><General WriteXMLEncoding='UTF-16' />"+
               "    <UnlockCode>(Enter Code here)</UnlockCode>"+
               "    <DataBase CountryISO3='ALL' Type='BATCH_INTERACTIVE'"+
               "     Path='/ADDB' PreloadingType='NONE' />"+
               "</SetConfig>", null,
               "<?xml version='1.0' encoding='UTF-16' ?>"+
               "<!DOCTYPE SetConfig SYSTEM 'Parameters.dtd'>"+
               "<Parameters WriteXMLEncoding='UTF-16'>"+
               "    <Input Encoding='UTF-16' />"+
               "    <Result Encoding='UTF-16' />"+
               "</Parameters>", null);
         iLastError = AddressDoctor.getLastError();
         System.out.println("Using AddressDoctor version: " + AddressDoctor.getVersion());
         System.out.println("Init returned " + iLastError);
       } catch (AddressDoctorException ex)
       {
         System.out.println("Exception while initializing "+
                "AddressDoctor: " + ex.toString());
         System.out.println("Further processing not possible, "+
                "application ends!");
         return;
       }

       try
       {
         m_oAO = AddressDoctor.getAddressObject();
       } catch (AddressDoctorException ex)
```

```
{
  System.out.println("Exception while trying to get an "+
         "AddressObject: " + ex.toString());
  System.out.println("Further processing not possible, "+
         "application ends!");

  try
  {
    AddressDoctor.deinitialize();
  } catch (AddressDoctorException ex2){}
  return;
}

try
{
 m_oAO.setInputDataXML(
 "<?xml version='1.0' encoding='UTF-16'?>"+
 "<!DOCTYPE InputData SYSTEM InputData.dtd'>"+
 "<InputData>"+
 "<AddressElements>"+
 "   <Key>4711</Key>"+
 "   <Country Item='1' Type='NAME'>SGP</Country>"+
 "   <Locality Item='1' Type='COMPLETE'>Singapore</Locality>"+
 "   <PostalCode Item='1' Type='FORMATTED'>048624</PostalCode>"+
 "   <Street Item='1' Type='COMPLETE'>Raffles Place</Street>"+
 "   <Number Item='1' Type='COMPLETE'>80</Number>"+
 "   <Building Item='1' Type='COMPLETE'>#50-01 UOB Plaza 1</Building>"+
 "   <Organization Item='1' Type='NAME'>AddressDoctor GmbH</Organization>"+
 "</AddressElements>"+
 "</InputData>");
} catch (Exception ex)
{
  System.out.println("Data could not be assigned! Closing "+
         "application: " + ex.toString());
  try
  {
    AddressDoctor.releaseAddressObject(m_oAO);
    AddressDoctor.deinitialize();
  } catch (AddressDoctorException ex2){}
    return;
}

try
{
  AddressDoctor.process(m_oAO);
  iLastError = AddressDoctor.getLastError();
  System.out.println("Process returned " + iLastError);
} catch (AddressDoctorException ex)
{
  System.out.println("Exception during process: " +
  ex.toString());
}

if (iLastError == 0)
{
  try
  {
    sResultXML = m_oAO.getResultXML();
  } catch (AddressDoctorException ex)
  {
    System.out.println("Exception while trying to get "+
    "ResultXML: " + ex.toString());
    return;
  }
  System.out.println(sResultXML);
}

try
{
  AddressDoctor.releaseAddressObject(m_oAO);
  AddressDoctor.deinitialize();
```

```
    } catch (AddressDoctorException ex)
    {
      System.out.println("Exception while releasing the AO and "+
      "de-initializing AddressDoctor: " + ex.toString());
    }
   }
}
```

CHAPTER 6

Demonstration Applications

This chapter includes the following topic:

- Demonstration Applications, 23

Demonstration Applications

The Informatica Address Verification software package includes two demonstration applications: AddressCheck and ConsoleDemo. You can use these applications to quickly test Address Verification features and functionality.

AddressCheck works on Windows-based devices. ConsoleDemo works on Windows-based devices and Unix-based devices.
Note: Address Verification provides AddressCheck and Console Demo applications on an as-is basis and does not provide any warranty or support for these applications. The applications are available only with standalone Address Verification packages.

AddressCheck

AddressCheck is a Windows based application that comes in the `bin` directory of the Informatica Address Verification software package. You can use the AddressCheck GUI to interactively enter fielded, partially fielded, or unfielded address data and validate addresses in the fast completion, interactive, batch, or certified mode. You can use the menu options in the AddressCheck GUI to configure the required parameters.

You can also use the AddressCheck application to generate Address Verification `InputData`, `GetConfig`, `Parameters`, and `Result` DTD files. These files provide useful information that Informatica Support can use to analyze and troubleshoot issues. The **Status Help** button is another useful feature that helps you analyze the element input status, element result status, address resolution code, and extended element result status values.

For AddressCheck to function correctly, complete the following prerequisites:

- Install Microsoft.NET Framework 2.0 or later on the device on which you install AddressCheck.
- Copy the following files to the `bin` directory:

  - `AddressDoctor5.dll` from `lib`

  - `SetConfig.xml` from `etc/Java`
- Update `SetConfig.xml` to include a valid unlock code, which you received along with the Address Verification license, and the path to the reference address database to which the unlock code maps.
- Set the value of the MaxAddressObjectCount parameter in `SetConfig.xml` to at least 6.

ConsoleDemo

The ConsoleDemo is a CLI-based application that you can use to parse and validate addresses. The source code and the executable for ConsoleDemo are provided in the Informatica Address Verification software package. The `src` directory of the Informatica Address Verification software package contains the source code for ConsoleDemo. The `bin` directory contains the ConsoleDemo executable.

For ConsoleDemo to work correctly, complete the following prerequisites:

- Copy the example XML files, including `InputData.xml` and `SetConfig.xml`, from the `etc` directory to your working directory. You can edit these files as required to experiment with the settings. `InputData.xml` contains address examples that you can process using consoledemo -xml or consoledemojava -xml.
- Update `SetConfig.xml` to include a valid unlock code, which you received along with the Address Verification license, and the path to the reference address database to which the unlock code maps. Alternatively, copy or link at least the Swiss reference database (`CHE5BI.MD`) to the working directory before running the ConsoleDemo executable. The ConsoleDemo application attempts to validate a sample address from Switzerland and requires the Swiss database to complete the validation. If it does not find the Swiss database, ConsoleDemo only parses the input address.
- Add the contents of the `lib` directory to the shared library path of your device. On Windows devices, run the set PATH=%PATH%;.\lib. On Unix-based devices, run `export LD_LIBRARY_PATH= $LD_LIBRARY_PATH:./lib`.