Informatica

Informatica® Address Verification
5.11.0

Installation and Getting Started Guide (On-Premises)

Informatica Address Verification Installation and Getting Started Guide (On-Premises)
5.11.0
April 2017

Publication Date: 2020-04-07

Table of Contents

Preface

Informatica Address Verification (On-Premises) enables you to verify postal addresses from across the globe. You can use Address Verification to verify postal addresses and to retrieve address enrichments such as geocoordinates and CAMEO consumer segmentation data.

The *Informatica Address Verification Installation and Getting Started Guide (On-Premises)* provides overviews of Address Verification software packages, reference address database packages, and the Informatica Database Download Manager. The guide provides step-by-step instructions to install and initialize Address Verification. The guide also contains information about Address Verification demonstration applications such as AddressCheck and ConsoleDemo.

Informatica Resources

Informatica Network

Informatica Network hosts Informatica Global Customer Support, the Informatica Knowledge Base, and other product resources. To access Informatica Network, visit https://network.informatica.com.

As a member, you can:

- Access all of your Informatica resources in one place.
- Search the Knowledge Base for product resources, including documentation, FAQs, and best practices.
- View product availability information.
- 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 search Informatica Network for product resources such as documentation, how-to articles, best practices, and PAMs.

To access the Knowledge Base, visit https://kb.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

To get the latest documentation for your product, browse the Informatica Knowledge Base at https://kb.informatica.com/_layouts/ProductDocumentation/Page/ProductDocumentSearch.aspx.

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

Informatica Product Availability Matrixes

Product Availability Matrixes (PAMs) indicate the versions of operating systems, databases, and other types of data sources and targets that a product release supports. If you are an Informatica Network member, you can access 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. Developed from the real-world experience of hundreds of data management projects, Informatica Velocity represents the collective knowledge of our consultants who have worked with organizations from around the world to plan, develop, deploy, and maintain successful data management solutions.

If you are an Informatica Network member, you can access 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 augment, extend, or enhance your Informatica implementations. By leveraging any of the hundreds of solutions from Informatica developers and partners, you can improve your productivity and speed up time to implementation on your projects. You can access Informatica Marketplace at https://marketplace.informatica.com.

Informatica Global Customer Support

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

To find your local Informatica Global Customer Support telephone number, visit the Informatica website at the following link:
http://www.informatica.com/us/services-and-training/support-services/global-support-centers.

If you are an Informatica Network member, you can use Online Support at http://network.informatica.com.

CHAPTER 1

Overview

This chapter includes the following topic:

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, although there are minor differences between them. 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:

Reference Address Databases and Packages

Informatica Address Verification reference address databases are proprietary-format, `.MD`, databases that contain reference addresses from countries and territories that Address Verification supports for validation.

The reference address databases are read-only and platform-independent. You can use the same database on all platforms that support Address Verification.

You can interpret the reference address database filename, `XXXYZZZZ.MD`, in the following way:

- `XXX` denotes the ISO 3166-1 alpha 3 code for the country or territory. For example, `USA` for the United States.
- `Y` denotes the Address Verification main version. For example, `5`.
- `ZZZZ` denotes the reference database type. For example, BI for Batch and Interactive or GCAP for High Precision Arrival Point Geocoding.

From Informatica Database Download Manager, you can download .zip packages of country- or territory-specific reference address databases.

The .zip file names follow the `DB5_XXXYZZZZ_NN_YYMMDD.zip` format. `XXX` in the file name denotes the ISO 3166-1 alpha 3 code for the country or territory. `ZZZZ` denotes the reference address database type such as BI or GCAP. `NN` denotes a unique, two-digit product ID. `YYMMDD` denotes the database release date.

For example, you can interpret the file name `DB5_USA5BI_01_150201.zip` in the following way:

> Informatica Address Verification Version 5 (`DB5`) Batch and Interactive (`BI`) database 1 for the United States (`USA`) that Informatica released on February 01, 2015 (`150201`).

Most countries have only one version, `01`, of a database type. However, some countries have multiple version of a database type. The two-digit package ID, such as `01`, `02`, and `03`, in the .zip package name uniquely identifies different versions of a database type.

For example, Australia has two versions each of the BI and FC databases. `DB5_AUS5BI_01_150201.zip` and `DB5_AUS5FC_01_150201.zip` contain the subbuilding information. `DB5_AUS5BI_02_150201.zip` and `DB5_AUS5FC_02_150201.zip` do not contain the subbuilding information.

Address Verification supports the following types of reference address databases:

BI

Batch and interactive

FC

Fast completion

Cx

Certified. x is a numeric. For example, `DB5_CAN5C1_01_150201.zip` contains the SERP-certified reference address database for Canada.

AC

Address code lookup

GC

Standard geocoding

GCAP

Arrival point geocoding

GCPC

Parcel centroid geocoding

GCRT

Rooftop geocoding

CA

CAMEO

Ex

Supplementary. x is a numeric. For example, `DB5_FRA5E1_01_150201.zip` contains the supplementary reference address database for France address enrichments such as INSEE code.

You can find multiple reference address databases for most countries and territories. For example, `DEU5BI.MD`, `DEU5FC.MD`, and `DEU5GC.MD` are some of the reference address databases for Germany addresses.

You need valid unlock codes to unlock these databases. The following table describes the unlock code types that Address Verification supports:

Unlock Code Type	Description
VALIDATION	Unlock code for batch and interactive (BI) and Fast Completion (FC) databases.
GEO_STANDARD	Unlock code for standard geocoding databases.
GEO_ARRIVAL_POINT	Unlock code for the arrival point geocoding databases.
GEO_PARCEL_CENTROID	Unlock code for the parcel centroid geocoding databases.
GEO_ROOFTOP	Unlock code for the rooftop geocoding databases.
CAMEO	Unlock code for the CAMEO databases.
ADDRESS_CODE_LOOKUP	Unlock code for the address code lookup databases.

Unlock Code Type	Description
SINGLE_LINE_VALIDATION	Unlock code for the single line address validation feature.
SUPPLEMENTARY	Unlock code for the supplementary databases that contain information about country-specific enrichments.

Informatica Address Database Download Manager Overview

You can use Informatica Address Database Download Manager (Informatica ADM) to select and download reference address databases for Informatica Address Verification. Informatica ADM is available as a web portal and as a Java based client application. You can access the Informatica ADM portal from the Informatica Passport dashboard. You can download the client application from the portal and install it on a device that has internet access.

The Informatica ADM portal and client use the Informatica Passport single sign-on. You can use your Informatica Passport user profile to seamlessly access Informatica tools and portals including Informatica ADM and Informatica Data Quality Center (Informatica DQC).

Note: If you do not have an Informatica Passport user profile, you must create an Informatica Passport user profile and assign Informatica DQC accounts to the user profile before you can use Informatica ADM.

By default, the Informatica ADM portal and client display the list of reference address databases that your Informatica Passport user profile is authorized to download. You get download rights for reference address databases when you buy unlock codes for the databases.

You can assign multiple Informatica DQC accounts to an Informatica Passport user profile. Informatica ADM lists all the databases that the Informatica DQC accounts associated with the user profile have permission to download.

From the Informatica ADM portal, you can download only one reference address database at a time. You can use the Informatica ADM client to download multiple databases together. You can use the default option of **Smart Download** in the Informatica ADM client to download the files that have been updated since you last downloaded the files.

You can also download the latest Informatica Address Verification software packages and documentation from the Informatica ADM portal.

Note: Informatica ADM returns a blank page in the following scenarios:

- Your Informatica Passport user profile is not associated with an Informatica DQC account.
- The Informatica DQC account that is associated with your Informatica Passport user profile does not have a valid unlock code provisioned for it.

You can use the filtering options available in the Informatica ADM portal and client to identify the databases that you need to download. You can also view the complete list of databases that are available for Informatica Address Verification customers by clearing the **Show Allowed** check box.

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

CHAPTER 3

System Requirements

This chapter includes the following topics:

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 varies greatly based on 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 want to 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, or Python on these examples. Informatica technical support, however, is limited to C and Java APIs. Informatica does not provide implementation-specific support.

You can install Address Verification on devices that run any of the following configurations:

Operating System	Processor Architecture	Java Development Kit
Windows Server 2008 SP2	x86 (32-bit)	Sun SE 7
Windows Server 2008 R2 Windows Server 2008 SP2 Windows Server 2012	x64 (64-bit)	Sun SE 7
SUSE Linux Enterprise Server 10 and 11	x86 (32-bit) x64 (64-bit)	Sun SE 7
RedHat Enterprise Linux 6 and 7	x86 (32-bit) x64 (64-bit)	Sun SE 7
RedHat Enterprise Linux 6 and 7	System z (64-bit)	IBM SE 7

Operating System	Processor Architecture	Java Development Kit
AIX 6 and 7	POWER (64-bit)	IBM SE 7
Solaris 10	Intel (64-bit)	Sun SE 7
Solaris 11	SPARC (64-bit)	Sun SE 7
HP-UX 11	Intel Itanium (64-bit)	HP SE 5

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 are required for your specific needs. Preloading databases significantly improves the performance of Address Verification. The device on which you install Address Verification must have sufficient RAM to preload all the required databases.

The complete set of worldwide postal reference databases including supplementary databases for address enrichments requires around 40 GB of storage space. However, for typical installations that do not require all the databases, 20 to 25 GB of RAM should be sufficient. If you need to preload databases that together have a size of 3 GB or more, use a 64-bit operating system that offers you more flexibility with the RAM size. The maximum available RAM for a 32-bit operating system is 3.2 GB.

Tip: If full preloading of 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.

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 38 MB for 32-bit systems and 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 3.7 MB + (0.24 MB x the value of `MaxResultCount`) in the case of 32-bit systems. For 64-bit systems, the size of an address object memory block is about 4.8 MB + (0.24 MB x the value set for `MaxResultCount`).

- 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

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_32/64_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

  HPU

  HP Unix

  RHT_SUSE

  Linux. Red Hat and Suse

  SOI

  Solaris Intel

  SOS

  Solaris SPARC

  WIN

  Windows

- `32/64`. Whether the package is for 32-bit systems or 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_32_141103_(5.6.0.30153).zip` indicates that the package contains Address Verification version 5.6.0 minor build 30153 fort the 32-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_32_141103_(5.6.0.30153)-sun.studio.11.zip` sun.studio.11 indicates that the package contains APIs for Solaris SPARC 32-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), `.so`, or `.sl` (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` or `.sl` 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 or on HP-UX, set the `export PTHREAD_DEFAULT_STACK_SIZE` 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 have the Java Runtime Environment (JRE) Version 7 set up on the device on which you install Address Verification. Address Verification versions earlier than Version 5.5.0 work with the JRE Version 6. If you want to develop your own applications, you must install the Java platform (JDK) SE 7 on the device. However, on HP-UX, you can continue to use HP SE 5 version of the JDK.

You can download the JRE package from the Sun Java website. Informatica does not officially support Versions 5.5.0 and later installations that run on JRE versions earlier than Version 7.

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.

   Note: For HP-UX devices, use `AddressDoctor5.sl` instead of `AddressDoctor5.so`.

   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
   ```

   Note: On 32-bit operating systems, Java Virtual Machine might exhaust the heap memory that can be assigned to an application when the heap memory assigned to applications exceeds 1.5 GB.

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 the `export AIXTHREAD_STK` to 1000000 or on HP-UX, set the `export PTHREAD_DEFAULT_STACK_SIZE` to 1000000.

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

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

CHAPTER 5

Initialization

This chapter includes the following topics:

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

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`.
- To call the Java API from Unix-based devices except IBM J9 JVM, run `java -Xss2048k -cp bin:lib/AddressDoctor5.jar -Djava.library.path=lib ConsoleDemoJava`.
  On IBM J9 JVM, run `java -Xmso2m -cp bin:lib/AddressDoctor5.jar -Djava.library.path=lib ConsoleDemoJava`.
- To call the Java API from Windows devices, run `java -Xss2048k -cp bin;lib\AddressDoctor5.jar -Djava.library.path=lib ConsoleDemoJava`.