Informatica® Multidomain MDM
10.4

Cleanse Adapter Guide

Informatica Multidomain MDM Cleanse Adapter Guide
10.4
March 2020

Publication Date: 2021-10-28

Table of Contents

Preface

Follow the instructions in the Informatica® *Multidomain MDM Cleanse Adapter Guide* to configure the supported cleanse engines that you want to use with Multidomain MDM. You use cleanse engines to cleanse, transform, and validate your master data. The guide also includes prerequisites and test procedures for the cleanse engines.

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

Introduction

This chapter includes the following topics:

Supported Cleanse Engines

The following table lists the cleanse engines that the MDM Hub supports and the Informatica MDM adapters with which they work:

Cleanse Engine	Informatica MDM Hub Adapter
IDQ	Informatica IDQ Adapter
Informatica Address Verification	Informatica Address Verification Adapter
Trillium	Trillium Director Adapter

Prerequisites

Before you can use your cleanse engines, you might need to do the following, depending on which cleanse engine you are using:

- Obtain a license with cleanse adapter enabled from Informatica.
- Install the Hub, application server, Process Server, and cleanse adapter. In some cases, you must install the Process Server and the cleanse adapter on the same server.
- Configure the cleanse engine and MDM Hub. See individual chapters of this guide more information.
- Test your cleanse-engine configuration.

For more information related to your cleanse adapter configuration, see your third party cleanse engine documentation, your application server documentation, the *Multidomain MDM Installation Guide*, and the *Multidomain MDM Release Notes*. You can also check the Informatica Knowledge Base for information about specific issues.

CHAPTER 2

Informatica IDQ Cleanse Engine

This chapter includes the following topics:

Informatica IDQ Cleanse Engine Overview

One of the ways to access data cleansing functionality in the Informatica Data Quality product is through Web services that Informatica publishes. This chapter describes how, in your Informatica MDM Hub implementation, to set up and use the MDM Cleanse Adapter for Informatica Data Quality (Web Services) to access cleanse functions that are published as Web services.

This functionality allows you add a new type of cleanse library - an IDQ cleanse library - to your Informatica MDM Hub implementation, and then integrate cleanse functions in the IDQ library into your mappings, just as you would integrate any other type of cleanse function available in your Informatica MDM Hub implementation. Informatica MDM Hub acts as a Web service client application that consumes Informatica Web services.

If you do not use the Informatica IDQ cleanse engine, you can skip this chapter.

Prerequisites for Data Cleansing

To use this functionality, you must have installed the following software:

- Informatica Multidomain MDM
- Informatica PowerCenter Informatica
- Informatica Multidomain MDM license that enables Informatica Data Quality cleanse library functionality (`siperian.informatica_DQ=yes` in the siperian.license file)

Any tools that you use must support the correct protocols and Java version. For more information about product requirements and supported platforms, see the Product Availability Matrix on Informatica Network.

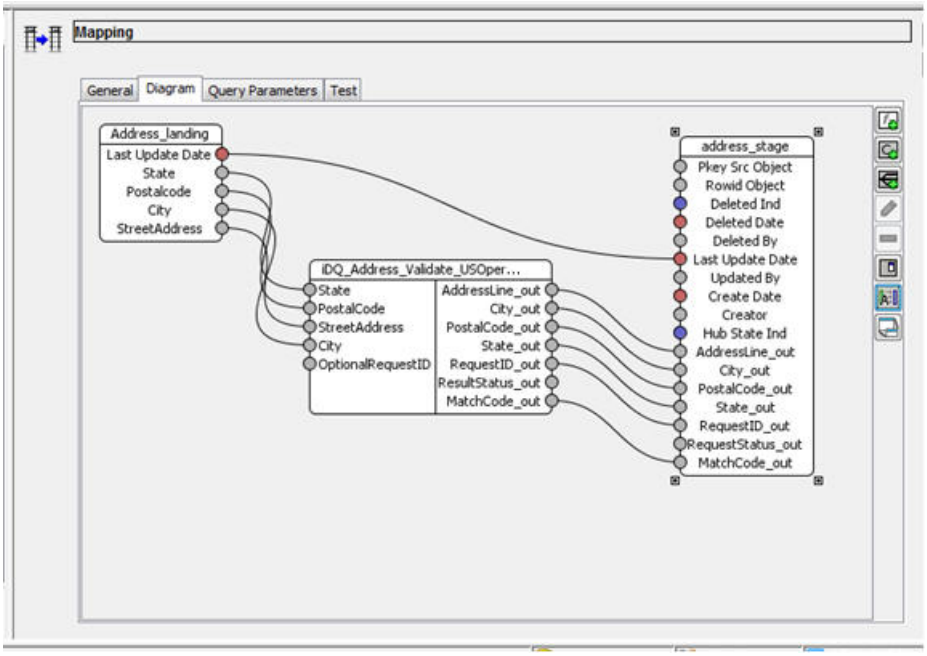
Process Overview

To use these Web services in your Informatica MDM Hub implementation, complete the following steps:

1. Create a mapping in IDQ or PowerCenter. See *PowerCenter Web Services Provider Guide* available at the Informatica MySupport website for information on how to create Web Services Description Language (WSDL) files for web services. Obtain the Informatica WSDL file for the Web services that you want to consume. Use the WSDL you created in Informatica PowerCenter.
2. In the Cleanse Functions tool of the Informatica MDM Hub Console, add an IDQ library and specify the URI of the Web Services Description Language (WSDL) file as well as connection information (service and port) to hosted Web services.

   The Cleanse Functions tool builds the IDQ library based on the WSDL, and then displays the list of cleanse functions defined in the WSDL. Each cleanse function represents a separate Web service.
3. Enable a generated library.
4. In the Mappings tool of the Informatica MDM Hub Console, use the available cleanse functions in your mappings as required. You must configure the inputs and outputs just as you would configure any other mappings.

   For performance reasons, the IDQ function should be the only one used in the mapping. You can have direct column-to-column mappings in the same mapping as an IDQ function, but you should not use other functions or conditional execution components in the same mapping as an IDQ function.



Related Topics:

- "Adding an IDQ Library in the Cleanse Functions Tool" on page 11
- "Enabling a Generated Library" on page 13
- "Run-time Behavior" on page 9

Run-time Behavior

At run time, the cleanse engine evaluates the number of records to be cleansed. If only one (1) record is sent for cleansing, then the web service is invoked for that single record. If multiple records are to be cleansed,

then the Process Server will batch a set of records together and pass that batch to the web service. This reduces latency on the web service call and results in better performance.

The maximum number of records included in a batch is determined by a parameter in the cmxcleanse.properties file:

```
cmx.server.cleanse.number_of_recs_batch= nnn
```

The default value for that parameter is 50.

Considerations

- Add the following parameter to the `cmxcleanse.properties` file if any of your IDQ functions do not support minibatch:

  ```
  cmx.server.cleanse.number_of_recs_batch=1
  ```

- Web service invocations are synchronous only. Asynchronous invocations are not supported.
- By default, web service invocations typically operate on a single record at a time. However, the MDM Hub cleanse engine will batch together records for cleansing to improve the performance. It can only do this if all the data transformation logic resides in the IDQ web service. If any transformation logic is defined in the MDM map, then the cleanse engine will not be able to use batching logic and the calls to the web service revert to being single record invocations of the web service
- The IDQ function must contain all transformation logic. This is so that no other functions are needed in the MDM map, thus allowing batches of records to be passed to IDQ for processing. The purpose of using the Web services is strictly to transform data that is passed in the request according to the associated cleanse function. Other types of Web services, such as publish/subscribe services, are not supported.
- If the Web service returns an error, Informatica MDM Hub moves the record to the reject table and saves a description of the problem (including any error information returned from the Web service).
- If the Web service is published on a remote system, the infrastructure must be in place for Informatica MDM Hub to connect to the Web service (such as a network that accesses the Internet).
- When using cleanse functions that are implemented as Web services, run-time performance of Web service invocations depends on some factors that are external to Informatica MDM Hub, such as availability of the Web service, the time required for the Web service to process the request and return the response, and network speed.
- You can run WSDL cleanse function with multi-threading. To enable this, change the thread count on the Informatica MDM Hub Process Server. Ensure that there is a sufficient number of instances of the IDQ Web Service to handle the multiple Informatica MDM cleanse threads; otherwise, records might be rejected due to timeouts.
- WSDL files must comply with the Axis2 Databinding Framework (ADB). Non-compliant WSDL files are not supported.
- When you configure mappings, you must ensure that the inputs and outputs are appropriate for the Web service you are calling. The Mappings tool does not validate your inputs and outputs – this is done by the Web service instead. If you have invalid inputs or outputs, the Web service returns an error response and processed records are moved to the reject table with an explanation of the error.

Adding an IDQ Library in the Cleanse Functions Tool

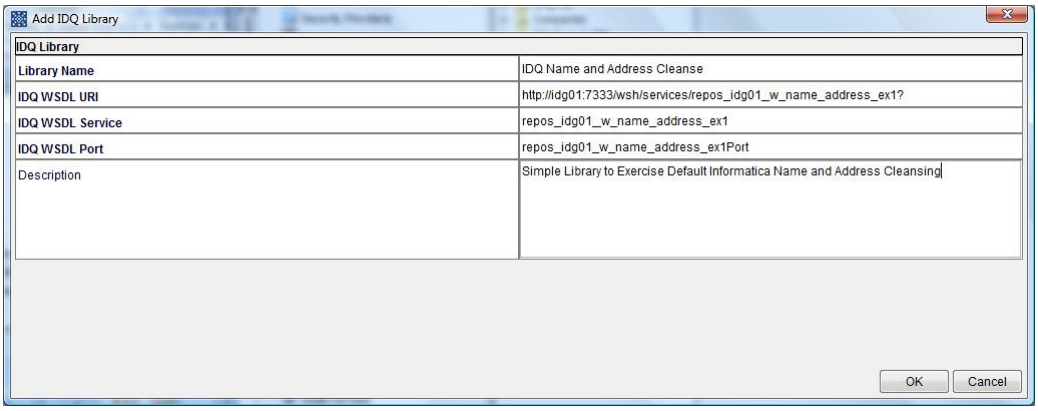
Once you have installed the prerequisite software and obtained an IDQ WSDL file, you use the Cleanse Functions tool in the Informatica MDM Hub Console to add the IDQ library to your Informatica MDM Hub implementation.

1. Launch the Informatica MDM Hub Console, if it is not already running.
2. Start the Cleanse Functions tool. You can right click anywhere in the Cleanse Functions tool to see more options.
3. Obtain a write lock (**Write Lock** > **Acquire Lock**).
4. Select the **Cleanse Functions** (root) node. Right click.
5. Choose **Cleanse Functions** > Add **IDQ Library**.
6. In the Add IDQ Library dialog, specify the following settings:

Setting	Description
Library Name	Name of this IDQ library. You can assign any arbitrary name that helps you classify and organize the collection of IDQ cleanse functions. Consider having IDQ in the name to distinguish this from other cleanse function libraries. This name appears as the folder name in the Cleanse Functions list.
IDQ WSDL URI	URI (location) of the IDQ WSDL to implement.
IDQ WSDL Service	Service of the IDQ WSDL to implement.
IDQ WSDL Port	Port of the IDQ WSDL to implement.
Description	Descriptive text for this library that you want displayed in the Cleanse Functions tool.

Note: Simple WSDLs often have only one Service and one Port. You can refer to the IDQ WSDL for the values to specify for these settings.

The following figure shows sample settings for an IDQ WSDL that invokes default name and address cleansing:

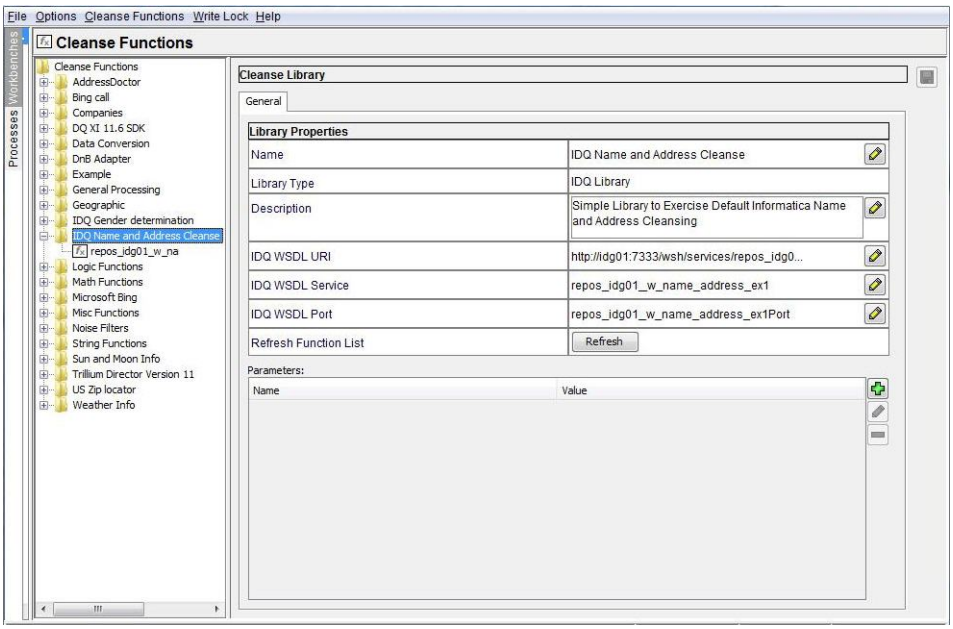


You must ensure that the IDQ WSDL response definition does not specify an array by setting the maxOccurs attribute value to " 1" , as shown in the sample:

```
<xsd:element minOccurs="1" maxOccurs="1" name="Street" type="xsd:string"/>
```

7. Click **OK** to add the metadata definition for this new IDQ library to the local ORS repository.
8. Click the **Refresh** button to generate the IDQ library.

   The Cleanse Functions tool retrieves the latest IDQ WSDL, generates the IDQ library, and displays any available cleanse functions in the Cleanse Functions list.



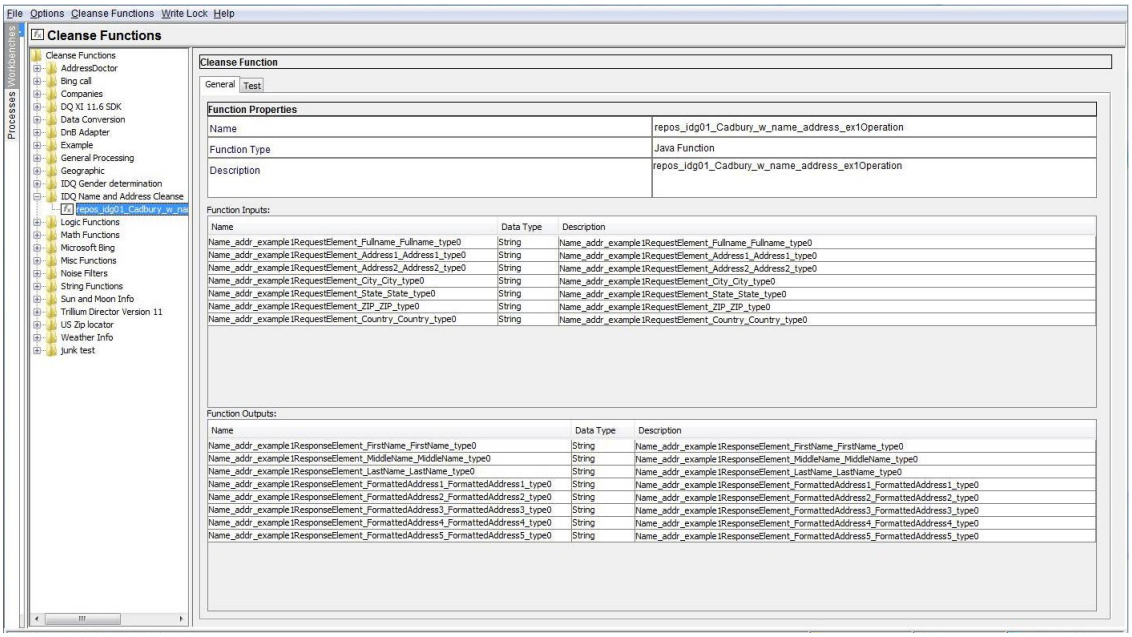
An error message is displayed in the following cases:

- If the Cleanse Functions tool cannot consume the IDQ WSDL file (for example, due to a syntax error), then it displays an error message instead. You must fix the IDQ WSDL file or obtain a valid one.

  Note:

  Ensure that the IDQ WSDL file does not contain an array in its response definition. You must set the maxOccurs attribute in the WSDL file to '1'.
- If changes in the IDQ WSDL file affect any existing mappings, the Cleanse Functions tool displays an error. You must fix the affected mappings before running the cleanse process.

9. Click a cleanse function to display its properties.

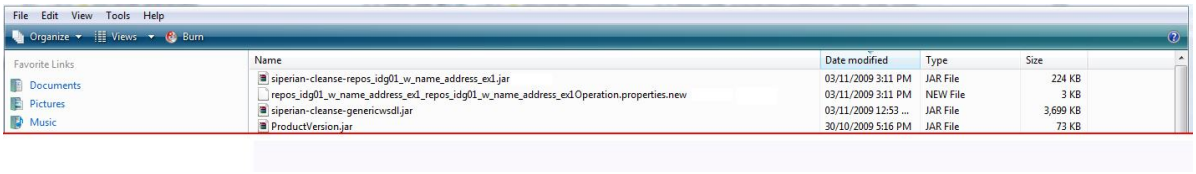


10. Test the function by clicking the **Test** tab and then clicking the **Test** button.
11. At this point, you can add these cleanse functions to your mappings in the Mappings tool, as shown in the “Process Overview” earlier in this chapter.

Configuring Generated Libraries

When the Cleanse Functions tool generates a library, it creates a set of properties files in the `SIP_HOME/cleanse/lib` directory with the following naming format:

```
siperian-cleanse-<servicename>_<functionname>.properties.new
```



Note: File names must be unique (service name + function name) within this directory.

Enabling a Generated Library

To enable a generated library, remove the `.new` extension from the file name.

Properties File Syntax

```
# endPointOverride =
#
# Inputs for <functionname>
#
```

```
#<wsdl_in_param1> = name , description
#<wsdl_in_param2> = name , description
#
# Outputs for <functionname>
#
#<wsdl_out_param1> = name , description
#<wsdl_out_param2> = name , description
```

Example Properties File

Below is an example code listing.

```
#endPointOverride =
#
# Inputs for name_Address
#
Req_Name_DITypeVarchar1024 = Req_Name_DITypeVarchar1024 , Req_Name_DITypeVarchar1024
Req_State_DITypeVarchar1024 = Req_State_DITypeVarchar1024 ,
Req_State_DITypeVarchar1024
Req_Address_DITypeVarchar1024 = Req_Address_DITypeVarchar1024 ,
Req_Address_DITypeVarchar1024
Req_City_DITypeVarchar1024 = Req_City_DITypeVarchar1024 , Req_City_DITypeVarchar1024
Req_Pcode_DITypeVarchar1024 = Req_Pcode_DITypeVarchar1024 ,
Req_Pcode_DITypeVarchar1024
#
# Outputs for name_Address
#
Res_Address_DITypeVarchar1024 = Res_Address_DITypeVarchar1024 ,
Res_Address_DITypeVarchar1024

Res_Name_DITypeVarchar1024 = Res_Name_DITypeVarchar1024 , Res_Name_DITypeVarchar1024
Res_State_DITypeVarchar1024 = Res_State_DITypeVarchar1024 ,
Res_State_DITypeVarchar1024
Res_City_DITypeVarchar1024 = Res_City_DITypeVarchar1024 , Res_City_DITypeVarchar1024
Res_Pcode_DITypeVarchar1024 = Res_Pcode_DITypeVarchar1024 ,
Res_Pcode_DITypeVarchar1024
```

Editing a Generated Properties File

You need edit a generated properties file for the following reasons:

- to change the connection endpoint
- to rename or remove extraneous parameters

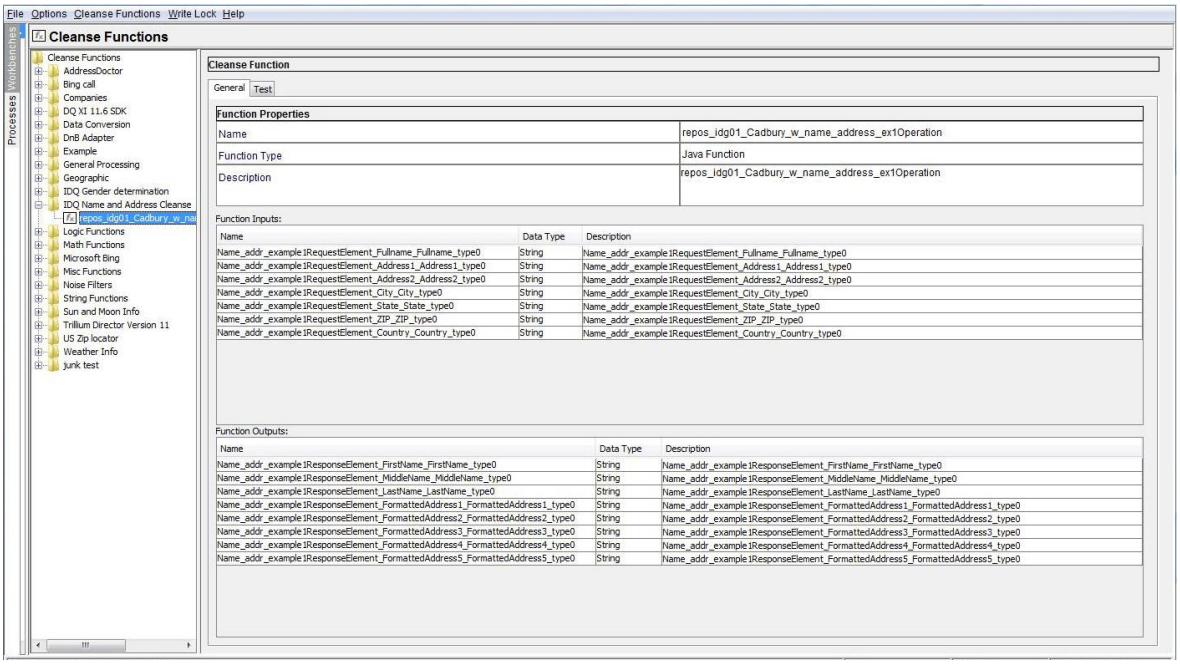
To edit a generated properties file:

1. Remove the `.new` extension from the end of the file name to enable it.
2. Open the file in a text editor.
3. Make the changes you want to the file, then save the file.
4. In the Cleanse Functions tool, select the IDQ library, then click the **Refresh** button to activate these changes in the properties file.

   Note: When you click **Refresh** to generate a library from a WSDL, the Cleanse Functions tool regenerates the file with the `.new` extension. Changes made to a file that has been renamed (without a`.new` extension) are not overwritten when the library is refreshed.

Managing Input and Output Parameters

The names generated from the WSDL can sometimes be long and difficult to read, or in an uncommon order, as shown in the following example.



You can edit the generated parameter file and rename parameters to make them easier to read and recognize. In fact, if you encounter an error indicating that one or more names are too long to be stored in the ORS repository (> 100 characters), then you must shorten these names.

You can simplify the library by removing extraneous parameters that are not needed for your Web service invocations. You can also reorder parameters.

To edit parameter names:

- For required parameters (that you want to use in your cleanse functions), uncomment by removing the # character. Any parameter that is not uncommented is *removed* from the function.
- Edit parameter names and descriptions as required. You cannot have duplicate parameter names in one function (duplicates are ignored).

The library must be refreshed for the changes to the properties file to take effect. In the Cleanse Functions tool, select the library and click **Refresh**.

For example, the following figure shows the default generated output for an IDQ library.

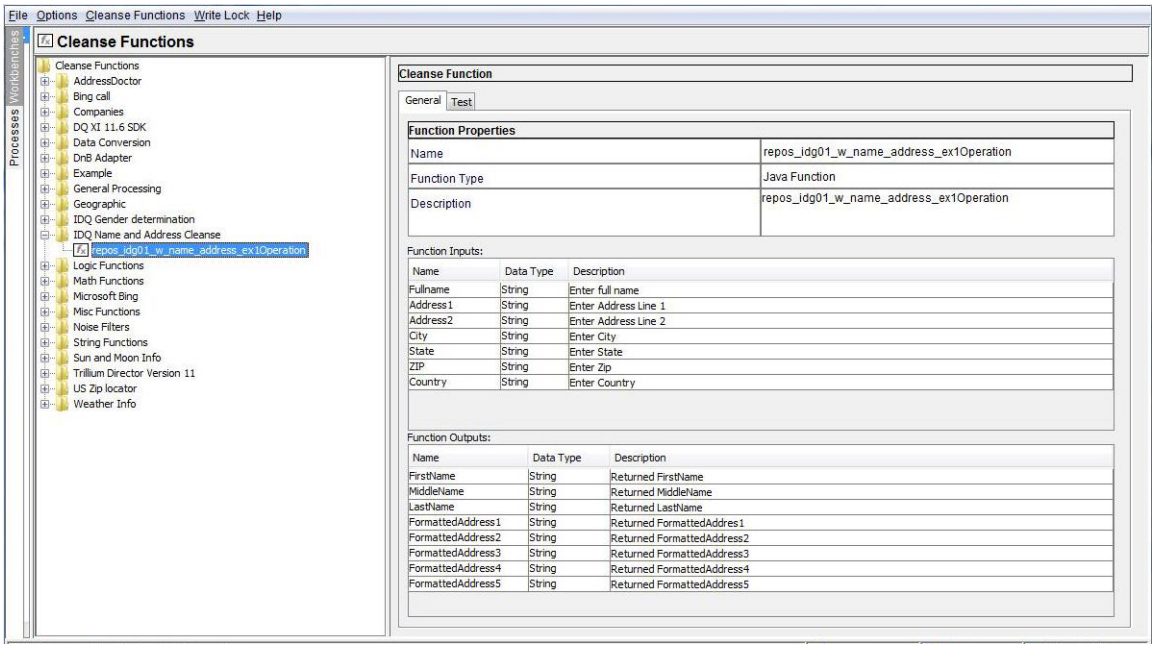
```
#endPointOverride =
#
# Inputs for repos_idg01_w_name_address_ex1Operation
#
Name_addr_example1RequestElement_Fullname_Fullname_type0 = Name_addr_example1RequestElement_Fullname_Fullname_type0 , Name_addr_example1RequestElement_Fullname_Fullname_type0
Name_addr_example1RequestElement_Address1_Address1_type0 = Name_addr_example1RequestElement_Address1_Address1_type0 , Name_addr_example1RequestElement_Address1_Address1_type0
Name_addr_example1RequestElement_Address2_Address2_type0 = Name_addr_example1RequestElement_Address2_Address2_type0 , Name_addr_example1RequestElement_Address2_Address2_type0
Name_addr_example1RequestElement_City_City_type0 = Name_addr_example1RequestElement_City_City_type0 , Name_addr_example1RequestElement_City_City_type0
Name_addr_example1RequestElement_State_State_type0 = Name_addr_example1RequestElement_State_State_type0 , Name_addr_example1RequestElement_State_State_type0
Name_addr_example1RequestElement_ZIP_ZIP_type0 = Name_addr_example1RequestElement_ZIP_ZIP_type0 , Name_addr_example1RequestElement_ZIP_ZIP_type0
Name_addr_example1RequestElement_Country_Country_type0 = Name_addr_example1RequestElement_Country_Country_type0 , Name_addr_example1RequestElement_Country_Country_type0
#
# Outputs for repos_idg01_w_name_address_ex1Operation
#
Name_addr_example1ResponseElement_FirstName_FirstName_type0 = Name_addr_example1ResponseElement_FirstName_FirstName_type0 , Name_addr_example1ResponseElement_FirstName_FirstName_type0
Name_addr_example1ResponseElement_MiddleName_MiddleName_type0 = Name_addr_example1ResponseElement_MiddleName_MiddleName_type0 , Name_addr_example1ResponseElement_MiddleName_MiddleName_typ
Name_addr_example1ResponseElement_LastName_LastName_type0 = Name_addr_example1ResponseElement_LastName_LastName_type0 , Name_addr_example1ResponseElement_LastName_LastName_type0
Name_addr_example1ResponseElement_FormattedAddress1_FormattedAddress1_type0 = Name_addr_example1ResponseElement_FormattedAddress1_FormattedAddress1_type0 , Name_addr_example1ResponseEleme
Name_addr_example1ResponseElement_FormattedAddress2_FormattedAddress2_type0 = Name_addr_example1ResponseElement_FormattedAddress2_FormattedAddress2_type0 , Name_addr_example1ResponseEleme
Name_addr_example1ResponseElement_FormattedAddress3_FormattedAddress3_type0 = Name_addr_example1ResponseElement_FormattedAddress3_FormattedAddress3_type0 , Name_addr_example1ResponseEleme
Name_addr_example1ResponseElement_FormattedAddress4_FormattedAddress4_type0 = Name_addr_example1ResponseElement_FormattedAddress4_FormattedAddress4_type0 , Name_addr_example1ResponseEleme
Name_addr_example1ResponseElement_FormattedAddress5_FormattedAddress5_type0 = Name_addr_example1ResponseElement_FormattedAddress5_FormattedAddress5_type0 , Name_addr_example1ResponseEleme
```

To rename a parameter, change its name setting as shown in the following example:

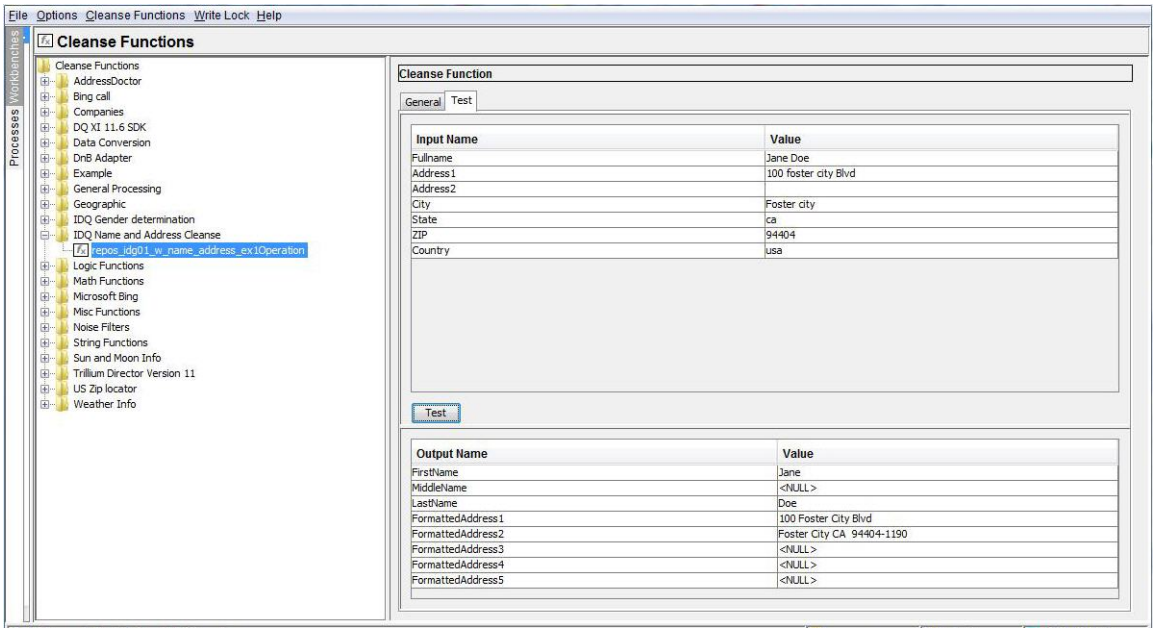
```
<parameter> = name , description
```

```
#endPointOverride =
#
# Inputs for repos_idg01_w_name_address_ex1Operation
#
Name_addr_example1RequestElement_Fullname_Fullname_type0 = Fullname , Enter full name
Name_addr_example1RequestElement_Address1_Address1_type0 = Address1 , Enter Address Line 1
Name_addr_example1RequestElement_Address2_Address2_type0 = Address2 , Enter Address Line 2
Name_addr_example1RequestElement_City_City_type0 = City , Enter City
Name_addr_example1RequestElement_State_State_type0 = State , Enter State
Name_addr_example1RequestElement_ZIP_ZIP_type0 = ZIP , Enter Zip
Name_addr_example1RequestElement_Country_Country_type0 = Country , Enter Country
#
# Outputs for repos_idg01_w_name_address_ex1Operation
#
Name_addr_example1ResponseElement_FirstName_FirstName_type0 = FirstName, Returned FirstName
Name_addr_example1ResponseElement_MiddleName_MiddleName_type0 = MiddleName , Returned MiddleName
Name_addr_example1ResponseElement_LastName_LastName_type0 = LastName , Returned LastName
Name_addr_example1ResponseElement_FormattedAddress1_FormattedAddress1_type0 = FormattedAddress1 , Returned FormattedAddres1
Name_addr_example1ResponseElement_FormattedAddress2_FormattedAddress2_type0 = FormattedAddress2 , Returned FormattedAddress2
Name_addr_example1ResponseElement_FormattedAddress3_FormattedAddress3_type0 = FormattedAddress3 , Returned FormattedAddress3
Name_addr_example1ResponseElement_FormattedAddress4_FormattedAddress4_type0 = FormattedAddress4 , Returned FormattedAddress4
Name_addr_example1ResponseElement_FormattedAddress5_FormattedAddress5_type0 = FormattedAddress5 , Returned FormattedAddress5
```

Save changes to the file and then, in the Cleanse Functions tool, click **Refresh** to update the modified cleanse function settings.



Once refreshed, be sure to test the cleanse function to verify its operation.



Updating the Connection Endpoints for Web Services

If the communication endpoint for a Web service changes, or if you must point to a different environment, update the endpoint URL.

1. Start the Hub Console.
2. From the Model workbench, click **Cleanse Functions**.

   The Cleanse Functions tool appears.
3. From the **Write Lock** menu, click **Acquire Lock**.

   The MDM Hub locks the cleanse libraries for editing.
4. Click the Informatica Data Quality library.

   The Informatica Data Quality cleanse library properties appear.
5. Update the Informatica Data Quality cleanse library properties for the web service.

   To update the properties for the Informatica Data Quality library, see the property values in the Informatica Data Quality WSDL file.

   The following table describes the Informatica Data Quality library properties that you need to update:

Property Name	Description
IDQ WSDL URI	URL of the Informatica Data Quality WSDL that you updated or implemented.
IDQ WSDL Service	Service name of the Informatica Data Quality WSDL that you updated or implemented.
IDQ WSDL Port	Port of the Informatica Data Quality WSDL that you updated or implemented.

6. Save the changes and click **Refresh**.

   The Cleanse Functions tool retrieves the latest Informatica Data Quality WSDL and generates the Informatica Data Quality library.

CHAPTER 3

Informatica Address Verification Cleanse Engine

This chapter includes the following topics:

Informatica Address Verification Cleanse Engine Overview

Integration between Informatica MDM Hub and the Informatica Address Verification cleanse engine occurs through the Informatica MDM Hub Informatica Address Verification Adapter. This adapter is an optional component.

This chapter explains how to configure your Informatica MDM Hub system to use the Informatica Address Verification Adapter and the Informatica Address Verification Cleanse Engine. This chapter assumes that you are knowledgeable about configuring and using the Informatica Address Verification software.

This chapter also discusses how to obtain the required files, how to edit the properties file, and how to configure the application server JVM settings.

Note: The Informatica MDM Hub requires a very specific Informatica Address Verification build of the Informatica Address Verification Cleanse Engine. Check the release notes for the most current information about this build.

Configuring the Informatica Address Verification Cleanse Engine

Before you can use the Informatica Address Verification Cleanse Engine, you must complete the following tasks:

1. Contact Informatica Support to obtain the following:
   - Correct unlock codes that must be set in the `SetConfig.xml` file.
   - License file (with the Informatica Address Verification cleanse adapter enabled). The license must include the Informatica Address Verification adapter.
2. Obtain the required reference data files.
3. Verify the settings in the `Parameters.xml` file.
4. Edit the cleanse properties file.
5. Add the Informatica Address Verification library to the PATH (`LIBPATH` or `LD_LIBRARY_PATH` depending on your platform) environment variable.

   Windows: Add `<infamdm_install_directory>\hub\cleanse\lib`.

   UNIX: Add `<infamdm_install_directory>/hub/cleanse/lib`.
6. Configure the application server JVM settings.

   Note: The Informatica MDM Hub requires a very specific Informatica Address Verification build of the Informatica Address Verification Cleanse Engine. For the most current information about this build, see the *Product Availability Matrix* for the release that is available at Informatica's customer support portal.

Related Topics:

- "Configuring the JVM Settings" on page 41

Obtaining the Required Files

You need to obtain Informatica Address Verification reference data files for your Informatica Address Verification cleanse engine configuration that have certifications, such as for the Software Evaluation and Recognition Program (SERP) and Address Matching Approval System (AMAS), and copy them to the Informatica Address Verification installation directory.

For example,

- **Windows:** C:\AddressDoctor\5
- **UNIX:** /u1/addressDoctor/5

Editing the Properties File

To edit the properties file:

1. Open the `cmxcleanse.properties` file for editing.

   This file is located in:

   Windows: < *infamdm_install_directory* > \hub\cleanse\resources

   UNIX: < *infamdm_install_directory* > /hub/cleanse/resources
2. Ensure that the following Informatica Address Verification 5 properties are set in the `cmxcleanse.properties` files:

Windows:

```
cleanse.library.addressDoctor.property.SetConfigFile=C:/infamdm/hub/cleanse/
resources/
AddressDoctor/5/SetConfig.xml
cleanse.library.addressDoctor.property.ParametersFile=C:/infamdm/hub/cleanse/
resources/
AddressDoctor/5/Parameters.xml
cleanse.library.addressDoctor.property.DefaultCorrectionType=PARAMETERS_DEFAULT
```

UNIX:

```
cleanse.library.addressDoctor.property.SetConfigFile=/u1/infamdm/hub/cleanse/
resources
/AddressDoctor/5/SetConfig.xml
cleanse.library.addressDoctor.property.ParametersFile=/u1/infamdm/hub/cleanse/
resources
/AddressDoctor/5/Parameters.xml
cleanse.library.addressDoctor.property.DefaultCorrectionType=PARAMETERS_DEFAULT
```

3. Copy `SetConfig.xml` and `Parameters.xml` to the location specified in the `cmxcleanse.properties` file.

   The following is a sample `SetConfig.xml` file:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<SetConfig>
  <General WriteXMLEncoding="UTF-16" WriteXMLBOM="NEVER" MaxMemoryUsageMB="1024"
MaxAddressObjectCount="10" MaxThreadCount="1"/>

  <UnlockCode>unlock_code</UnlockCode>
  <DataBase CountryISO3="ALL" Type="BATCH_INTERACTIVE" Path="<Address Verification
path>" PreloadingType="NONE"/>
</SetConfig>
```

   The following is a sample `Parameters.xml` file:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<!DOCTYPE Parameters SYSTEM 'Parameters.dtd'>
<Parameters
    WriteXMLEncoding="UTF-16"
    WriteXMLBOM="NEVER">
    <Process
        Mode="BATCH"
        EnrichmentGeoCoding="ON"
        EnrichmentCASS="ON"
        EnrichmentSERP="ON"
        EnrichmentSNA="ON"
        EnrichmentSupplementaryGB="ON"
        EnrichmentSupplementaryUS="ON" />
    <Input
        Encoding="UTF-16"
        FormatType="ALL"
        FormatWithCountry="ON"
        FormatDelimiter="PIPE" />
    <Result
        AddressElements="STANDARD"
        Encoding="UTF-16"
        CountryType="NAME_EN"
        FormatDelimiter="PIPE" />
</Parameters>
```

4. Save and close the properties file.
5. Connect to the database as the schema owner and run the following statement to enable Informatica Address Verification:

```
update C_REPOS_CL_FUNCTION_LIB set DISPLAY_IND = 1 where function_lib_name =
'AddressDoctor'
```

6. Restart your application server.
7. Verify that the application server started up properly with no errors.

   The Informatica Address Verification library is displayed in the Hub Console.

8. Update your JVM settings. This increases resources available to the JVM.

Note: You must update Informatica Address Verification settings in the `cmxcleanse.properties` file:

- If you move the `SetConfig.xml` and `Parameters.xml` files to a location other than that specified during installation.
- If you need to use a different CorrectionType. The supported CorrectionTypes are PARAMETERS_DEFAULT, PARSE_ONLY, CORRECT_ONLY, CERTIFY_ONLY, CORRECT_THEN_CERTIFY, and TRY_CERTIFY_THEN_CORRECT.

Related Topics:

- "Troubleshooting" on page 51
- "Configuring the JVM Settings" on page 41

Upgrading the Informatica Address Verification Cleanse Engine

If you are upgrading your Informatica MDM installation, you must also upgrade your Informatica Address Verification Cleanse Engine. The following sections cover configuration information for upgrading the Informatica Address Verification Cleanse Engine.

The Informatica Address Verification Cleanse Engine libraries are included as part of the Informatica MDM Hub upgrade. If you intend to use Informatica Address Verification Cleanse Engine as your cleanse engine, contact Informatica Global Customer Support for the correct unlock codes.

To configure Informatica Address Verification:

1. Obtain the required files.
2. Obtain unlock codes that must be set in the `SetConfig.xml` file.
3. Verify the settings in the `Parameters.xml` file.
4. Edit the Hub Cleanse Match properties file.
5. Replace the old Informatica Address Verification library with the new Informatica Address Verification library.
6. Configure the application server JVM settings.

Related Topics:

- "Editing the Properties File" on page 19
- "Configuring the JVM Settings" on page 41

Obtaining the Required Files

You need to obtain Informatica Address Verification reference data files for your Informatica Address Verification cleanse engine configuration that have certifications, such as SERP and AMAS, and copy them to the Informatica Address Verification installation directory.

For example,

- **Windows:** C:\AddressDoctor\5

- **UNIX:** /u1/addressDoctor/5

Editing the Properties File

Verify that the information in your `cmxcleanse.properties` files is still valid.

Related Topics:

- "Editing the Properties File" on page 19

Replacing Informatica Address Verification 4 Library with Informatica Address Verification 5 Library

You must replace Informatica Address Verification 4 library with the Informatica Address Verification 5 library.

1. Copy the Informatica Address Verification 5 library from the following location:

   Windows: `<infamdm_install_directory>\hub\cleanse\lib\upgrade\AddressDoctor`

   UNIX: `<infamdm_install_directory>/hub/cleanse/lib/upgrade/AddressDoctor`

2. Replace `JADE.dll` (or equivalent Informatica Address Verification 4 library) with the Informatica Address Verification 5 library at the following location:

   - Windows: `<infamdm_install_directory>\hub\cleanse\lib`
   - UNIX: `<infamdm_install_directory>/hub/cleanse/lib`

   For more information, refer to the `libupdate_readme.txt` document available at the following location:
   Windows: `<infamdm_install_directory>\hub\cleanse\lib\upgrade`

   UNIX: `<infamdm_install_directory>/hub/cleanse/lib/upgrade`

3. Restart the application server:

   a. Ensure that you are logged in with the same user name that is currently running the application server.

   b. Restart the application server.

   c. Check that no exceptions occur while starting the application server.

Steps to Upgrade to Informatica Address Verification 5 Integration

This section describes the upgrade process required for the MDM Hub implementation to use Informatica Address Verification 5.

Note: This section is applicable to users with a license for using Informatica Address Verification.

You must perform the following steps to upgrade to Informatica Address Verification 5 integration:

1. Open the `cmxcleanse.properties` file.This file is located at:

   Windows: `<infamdm_install_directory>\hub\cleanse\resources`

   UNIX: `<infamdm_install_directory>/hub/cleanse/resources`

2. Ensure that the following Informatica Address Verification 5 properties are set in the `cmxcleanse.properties` files:

   Windows:

   ```
   cleanse.library.addressDoctor.property.SetConfigFile=C:\infamdm\hub\cleanse\resources
   \AddressDoctor\5\SetConfig.xml
   cleanse.library.addressDoctor.property.ParametersFile=C:\infamdm\hub\cleanse
   \resources
   \AddressDoctor\5\Parameters.xml
   cleanse.library.addressDoctor.property.DefaultCorrectionType=PARAMETERS_DEFAULT
   ```

   UNIX:

   ```
   cleanse.library.addressDoctor.property.SetConfigFile=/u1/infamdm/hub/cleanse/
   resources/
   AddressDoctor/5/SetConfig.xml
   cleanse.library.addressDoctor.property.ParametersFile=/u1/infamdm/hub/cleanse/
   resources/
   AddressDoctor/5/Parameters.xml
   cleanse.library.addressDoctor.property.DefaultCorrectionType=PARAMETERS_DEFAULT
   ```

3. Save and close the properties file.
4. Copy `SetConfig.xml` and `Parameters.xml` to the location specified in the `cmxcleanse.properties` file.

   The following is a sample `SetConfig.xml` file:

   ```
   <!DOCTYPE SetConfig SYSTEM 'SetConfig.dtd'>
   <SetConfig>
       <General WriteXMLEncoding="UTF-16" WriteXMLBOM="NEVER"
       MaxMemoryUsageMB="600" MaxAddressObjectCount="10" MaxThreadCount="10" />

       <UnlockCode>79FYL9UAXAVSR0KLV1TDC6PAQVVC3KM14FZC</UnlockCode>

       <DataBase CountryISO3="ALL" Type="BATCH_INTERACTIVE" Path="c:\addressdoctor\5"
           PreloadingType="NONE" />

       <DataBase CountryISO3="ALL" Type="FASTCOMPLETION" Path="c:\addressdoctor\5"
           PreloadingType="NONE" />

       <DataBase CountryISO3="ALL" Type="CERTIFIED" Path="c:\addressdoctor\5"
           PreloadingType="NONE" />

       <DataBase CountryISO3="ALL" Type="GEOCODING" Path="c:\addressdoctor\5"
           PreloadingType="NONE" />

           <DataBase CountryISO3="ALL" Type="SUPPLEMENTARY" Path="c:\addressdoctor\5"
           PreloadingType="NONE" />
   </SetConfig>
   ```

   The following is a sample `Parameters.xml` file:

   ```
   <?xml version="1.0" encoding="iso-8859-1"?>
   <!DOCTYPE Parameters SYSTEM 'Parameters.dtd'>
   <Parameters
       WriteXMLEncoding="UTF-16"
       WriteXMLBOM="NEVER">
       <Process
           Mode="BATCH"
           EnrichmentGeoCoding="ON"
           EnrichmentCASS="ON"
           EnrichmentSERP="ON"
           EnrichmentSNA="ON"
           EnrichmentSupplementaryGB="ON"
           EnrichmentSupplementaryUS="ON" />
       <Input
           Encoding="UTF-16"
           FormatType="ALL"
           FormatWithCountry="ON"
           FormatDelimiter="PIPE" />
       <Result
   ```

```
            AddressElements="STANDARD"
            Encoding="UTF-16"
            CountryType="NAME_EN"
            FormatDelimiter="PIPE" />
    </Parameters>
```

5. Specify the Informatica Address Verification 5 unlock code in the configuration file, `SetConfig.xml`.

   For more information about the `SetConfig.xml` file and `Parameters.xml` file, refer to your Informatica Address Verification 5 documentation.

6. Copy the Informatica Address Verification 5 library from the following location:

   Windows: <*infamdm_install_directory*>\hub\cleanse\lib\upgrade\AddressDoctor

   UNIX: <*infamdm_install_directory*>/hub/cleanse/lib/upgrade/AddressDoctor

7. Replace `JADE.dll` (or equivalent Informatica Address Verification 4 library) with the Informatica Address Verification 5 library at the following location:

   Windows: <*infamdm_install_directory*>\hub\cleanse\lib

   UNIX: <*infamdm_install_directory*>/hub/cleanse/lib

   For more information, refer to the `libupdate_readme.txt` document available at:

   Windows: <*infamdm_install_directory*>\hub\cleanse\lib\upgrade

   UNIX: <*infamdm_install_directory*>/hub/cleanse/lib/upgrade

8. Restart the application server.

   Ensure that you are logged in with the same user name that is currently running the application server and that no exceptions occur while starting the application server.

9. Restart the Process Server.

   During the Process Server initialization, you should see a message similar to the following in the terminal console:

   ```
   [INFO ] com.siperian.mrm.cleanse.addressDoctor.Library: Initializing AddressDoctor5
   ```

10. Start the Cleanse Functions tool.
11. Obtain a write lock (**Write Lock** > **Acquire Lock**).
12. Select the Informatica Address Verification cleanse function.
13. Click the **Refresh** button.

    The Informatica Address Verification 5 cleanse function is added to the Informatica Address Verification cleanse functions node.

Informatica Address Verification 5 Fields and Process Status Values

The Informatica Address Verification 5 input fields, output fields, and process status values are significantly different from those used in Informatica Address Verification 4. If you have upgraded from Informatica Address Verification 4 to Informatica Address Verification 5, use the new Informatica Address Verification 5 field names and process status values. If you use the Informatica Address Verification 4 field names and process values, the MDM Hub automatically converts them to the Informatica Address Verification 5 names and values, but you will not benefit from the new features in Informatica Address Verification 5. This article provides examples and descriptions of the Informatica Address Verification 5 input and output fields and a comparison of the process status values to assist in the transition from Informatica Address Verification 4 to Informatica Address Verification 5.

Informatica Address Verification 5 Input Fields

The input field names in Informatica Address Verification 5 are different than those in Informatica Address Verification 4. See the descriptions and examples in the following table to understand what data is handled by a particular field.

The following table lists all the Informatica Address Verification 5 input fields. The 'Field #' column indicates the range of fields available. For example, the available field names for AEBuilding#COMPLETE are: AEBuilding1COMPLETE, AEBuilding2COMPLETE, AEBuilding3COMPLETE, AEBuilding4COMPLETE, AEBuilding5COMPLETE, and AEBuilding6COMPLETE.

Informatica Address Verification 5 Input Field Name	Field #	Description
ACComplete	-	Contains the formatted address, with each formatted address line separated by a vertical bar character ('|').
AEBuilding#COMPLETE	1-6	See example 3.
AEBuilding#COMPLETE_WITH_SUBBUILDING	1-6	Contains the complete building and subbuilding information.
AEBuilding#DESCRIPTOR	1-6	See example 3.
AEBuilding#NAME	1-6	See example 3.
AEBuilding#NUMBER	1-6	Contains the building number.
AEContact#COMPLETE	1-3	See examples 1, 2, 3, and 4.
AEContact#FIRST_NAME	1-3	See examples 1, 2, 3, and 4.
AEContact#FUNCTION	1-3	Contains the job function of the contact.
AEContact#GENDER	1-3	Contains the gender of the contact.
AEContact#LAST_NAME	1-3	See examples 1, 2, 3, and 4.
AEContact#MIDDLE_NAME	1-3	See examples 1, 2, and 3.
AEContact#NAME	1-3	See examples 1, 2, 3, and 4.
AEContact#SALUTATION	1-3	See examples 1, 2, 3, and 4.
AEContact#TITLE	1-3	Contains the job title of the contact.
AECountry#ABBREVIATION	1-3	Contains the abbreviated country name.
AECountry#ISO_NUMBER	1-3	See example 4.
AECountry#ISO2	1-3	See examples 1, 2, 3, and 4.
AECountry#ISO3	1-3	See examples 1, 2, 3, and 4.
AECountry#NAME	1-3	See examples 1, 2, 3, and 4.

Informatica Address Verification 5 Input Field Name	Field #	Description
AEDeliveryService#ADD_INFO	1-3	Contains additional delivery service information that passes through validation unchanged.
AEDeliveryService#COMPLETE	1-3	See example 1.
AEDeliveryService#DESCRIPTOR	1-3	See example 1.
AEDeliveryService#NUMBER	1-3	See example 1.
AEKey#RECORD_ID	1-3	-
AEKey#TRANSACTION_KEY	1-3	-
AELocality#ADD_INFO	1-6	Contains locality information that passes through validation unchanged.
AELocality#COMPLETE	1-6	See examples 1, 2, 3, and 4.
AELocality#NAME	1-6	See examples 1, 2, 3, and 4.
AELocality#SORTING_CODE	1-6	Contains a sorting code that defines where mail is sorted for localities with more than one sorting location.
AENumber#ADD_INFO	1-6	Contains additional street number information that passes through validation unchanged.
AENumber#COMPLETE	1-6	See example 3.
AENumber#DESCRIPTOR	1-6	See example 3.
AENumber#NUMBER	1-6	See examples 1, 2, 3, and 4.
AEOrganization#COMPLETE	1-3	See example 4.
AEOrganization#DEPARTMENT	1-3	See example 4.
AEOrganization#DESCRIPTOR	1-3	See example 4.
AEOrganization#NAME	1-3	See example 4.
AEPostalCode#FORMATTED	1-3	See example 2.
AEPostalCode#UNFORMATTED	1-3	See example 2.
AEPostalCode#BASE	1-3	See example 2.
AEPostalCode#ADD_ON	1-3	See example 2.
AEProvince#ABBREVIATION	1-6	Contains the abbreviated province name.
AEProvince#COUNTRY_STANDARD	1-6	Contains the standard province name.

Informatica Address Verification 5 Input Field Name	Field #	Description
AEProvince#EXTENDED	1-6	See example 2.
AEProvince#ISO	1-6	Contains the ISO province code.
AEResidue#NECESSARY	1-6	In parsing mode: contains address data that is redundant.
AEResidue#SUPERFLUOUS	1-6	In batch, certified, fast completion, or interactive mode: contains address data that is redundant.
AEStreet#ADD_INFO	1-6	Contains additional street information that passes through validation unchanged.
AEStreet#COMPLETE	1-6	See examples 1, 2, 3, and 4.
AEStreet#COMPLETE_WITH_NUMBER	1-6	See examples 1, 2, 3, and 4.
AEStreet#NAME	1-6	See examples 1, 2, 3, and 4.
AEStreet#POST_DESCRIPTOR	1-6	See examples 1, 2, 3, and 4.
AEStreet#POST_DIRECTIONAL	1-6	See example 4.
AEStreet#PRE_DESCRIPTOR	1-6	Contains street descriptors that appear before the street name.
AEStreet#PRE_DIRECTIONAL	1-6	See example 2.
AESubBuilding#COMPLETE	1-6	See example 3.
AESubBuilding#DESCRIPTOR	1-6	See example 3.
AESubBuilding#NAME	1-6	See example 3.
AESubBuilding#NUMBER	1-6	See example 3.
ALCountrySpecificLocalityLine#	1-6	See example 5.
ALDeliveryAddressLine#	1-6	See example 5.
ALFormattedAddressLine#	1-19	See example 5.

Informatica Address Verification 5 Input Field Name	Field #	Description
ALRecipientLine#	1-6	See example 5.
MDMCorrectionType	-	This input has these possible values: - CORRECT_ONLY. In this mode, Informatica Address Verification performs address standardization and validates the address components against address reference tables. This is the equivalent to Batch mode. - CERTIFY_ONLY. In this mode, Informatica Address Verification validates an address according to the certification rules defined by the local postal authority. In the US, Certify_Only mode uses CASS. This is equivalent to Certified mode. - CORRECT_THEN_CERTIFY. In this mode, Informatica Address Verification process the address in Batch mode and then in Certified mode. - TRY_CERTIFY_THEN_CORRECT. In this mode, Informatica Address Verification processes the address in Certified mode. If issues are encountered, the address is processed again in Batch mode. - PARAMETERS_DEFAULT. In this mode, the correction type is derived from the `Parameters.xml` file. - PARSE_ONLY. In this mode, Informatica Address Verification parses the address, but does not correct or certify the address.

Informatica Address Verification 5 Output Fields

The output field names in Informatica Address Verification 5 are different than those in Informatica Address Verification 4. See the descriptions and examples in the following table to understand what data is handled by a particular field.

The following table lists all the Informatica Address Verification 5 output fields. The 'Field #' column indicates the range of fields available. For example, the available field names for AEBuilding#COMPLETE are: AEBuilding1COMPLETE, AEBuilding2COMPLETE, AEBuilding3COMPLETE, AEBuilding4COMPLETE, AEBuilding5COMPLETE, and AEBuilding6COMPLETE.

Informatica Address Verification 5 Output Field Name	Field #	Description
ACComplete	-	Contains the formatted address, with each formatted address line separated by a vertical bar character. For example, addressline1|addressline2|addressline3.
AEBuilding#COMPLETE	1-6	See example 3.
AEBuilding#COMPLETE_WITH_SUBBUILDING	1-6	Contains the complete building and subbuilding information.
AEBuilding#DESCRIPTOR	1-6	See example 3.
AEBuilding#NAME	1-6	See example 3.

Informatica Address Verification 5 Output Field Name	Field #	Description
AEBuilding#NUMBER	1-6	Contains the building number.
AEContact#COMPLETE	1-3	See examples 1, 2, 3, and 4.
AEContact#FIRST_NAME	1-3	See examples 1, 2, 3, and 4.
AEContact#FUNCTION	1-3	Contains the job function of the contact.
AEContact#GENDER	1-3	Contains the gender of the contact.
AEContact#LAST_NAME	1-3	See examples 1, 2, 3, and 4.
AEContact#MIDDLE_NAME	1-3	See examples 1, 2, and 3.
AEContact#NAME	1-3	See examples 1, 2, 3, and 4.
AEContact#SALUTATION	1-3	See examples 1, 2, 3, and 4.
AEContact#TITLE	1-3	Contains the title of the contact.
AECountry#ABBREVIATION	1-3	Contains the abbreviated country name.
AECountry#ISO_NUMBER	1-3	See example 4.
AECountry#ISO2	1-3	See examples 1, 2, 3, and 4.
AECountry#ISO3	1-3	See examples 1, 2, 3, and 4.
AECountry#NAME_CN	1-3	Contains the name of the country in Chinese.
AECountry#NAME_DA	1-3	Contains the name of the country in Danish.
AECountry#NAME_DE	1-3	Contains the name of the country in German.
AECountry#NAME_EN	1-3	Contains the name of the country in English. See examples 1, 2, 3, and 4.
AECountry#NAME_ES	1-3	Contains the name of the country in Spanish.
AECountry#NAME_FI	1-3	Contains the name of the country in Finnish.
AECountry#NAME_FR	1-3	Contains the name of the country in French.
AECountry#NAME_GR	1-3	Contains the name of the country in Greek.
AECountry#NAME_HU	1-3	Contains the name of the country in Hungarian.
AECountry#NAME_IT	1-3	Contains the name of the country in Italian.
AECountry#NAME_JP	1-3	Contains the name of the country in Japanese.
AECountry#NAME_KR	1-3	Contains the name of the country in Korean.

Informatica Address Verification 5 Output Field Name	Field #	Description
AECountry#NAME_NL	1-3	Contains the name of the country in Dutch.
AECountry#NAME_PL	1-3	Contains the name of the country in Polish.
AECountry#NAME_PT	1-3	Contains the name of the country in Portuguese.
AECountry#NAME_RU	1-3	Contains the name of the country in Russian.
AECountry#NAME_SA	1-3	Contains the name of the country in Arabic.
AECountry#NAME_SE	1-3	Contains the name of the country in Swedish.
AEDeliveryService#ADD_INFO	1-3	Contains additional delivery service information.
AEDeliveryService#COMPLETE	1-3	See example 1.
AEDeliveryService#DESCRIPTOR	1-3	See example 1.
AEDeliveryService#NUMBER	1-3	See example 1.
AEKey#RECORD_ID	1-3	Contains the record ID.
AEKey#TRANSACTION_KEY	1-3	Contains the transaction key.
AELocality#ADD_INFO	1-6	Contains portions of the locality that could not be validated against reference data.
AELocality#COMPLETE	1-6	See examples 1, 2, 3, and 4.
AELocality#NAME	1-6	See examples 1, 2, 3, and 4.
AELocality#SORTING_CODE	1-6	Contains a sorting code defining where mail is sorted for localities with more than one sorting location.
AENumber#ADD_INFO	1-6	Contains portions of the street number that could not be validated against reference data.
AENumber#COMPLETE	1-6	See example 3.
AENumber#DESCRIPTOR	1-6	See example 3.
AENumber#NUMBER	1-6	See examples 1, 2, 3, and 4.
AEOrganization#COMPLETE	1-3	See example 4.
AEOrganization#DEPARTMENT	1-3	See example 4.
AEOrganization#DESCRIPTOR	1-3	See example 4.
AEOrganization#NAME	1-3	See example 4.
AEPostalCode#FORMATTED	1-3	See example 2.

Informatica Address Verification 5 Output Field Name	Field #	Description
AEPostalCode#UNFORMATTED	1-3	See example 2.
AEPostalCode#Base	1-3	See example 2.
AEPostalCode#ADD_ON	1-3	See example 2.
AEProvince#ABBREVIATION	1-6	Contains the abbreviated province name.
AEProvince#COUNTRY_STANDARD	1-6	Contains the standard province name.
AEProvince#EXTENDED	1-6	See example 2.
AEProvince#ISO	1-6	Contains the ISO province code.
AEResidue#NECESSARY	1-6	In parsing mode: contains address data that is redundant.
AEResidue#SUPERFLUOUS	1-6	In batch, certified, fast completion, or interactive mode: contains address data that is redundant.
AEResidue#UNRECOGNIZED	1-6	Contains data that cannot be parsed to an address port.
AEStreet#ADD_INFO	1-6	Contains portions of the street information that could not be validated against reference data.
AEStreet#COMPLETE	1-6	See examples 1, 2, 3, and 4.
AEStreet#COMPLETE_WITH_NUMBER	1-6	See examples 1, 2, 3, and 4.
AEStreet#NAME	1-6	See examples 1, 2, 3, and 4.
AEStreet#POST_DESCRIPTOR	1-6	See examples 1, 2, 3, and 4.
AEStreet#POST_DIRECTIONAL	1-6	See example 4.
AEStreet#PRE_DESCRIPTOR	1-6	Contains street descriptors that appear before the street name.
AEStreet#PRE_DIRECTIONAL	1-6	See example 2.
AESubBuilding#COMPLETE	1-6	See example 3.
AESubBuilding#DESCRIPTOR	1-6	See example 3.
AESubBuilding#NAME	1-6	See example 3.
AESubBuilding#NUMBER	1-6	See example 3.
ALCountrySpecificLocalityLine#	1-6	See example 5.
ALDeliveryAddressLine#	1-6	See example 5.

Informatica Address Verification 5 Output Field Name	Field #	Description
ALFormattedAddressLine#	1-19	See example 5.
ALRecipientLine#	1-6	See example 5.
EDCASSStatus	-	Indicates if the address contains enough data for USPS Coding Accuracy Support System (CASS) certification.
EDGeoCodingStatus	-	Indicates the level of accuracy of the Geocode value.
EDSERPStatus	-	Indicates if the address contains enough data for Canada Post Software Evaluation and Recognition Program (SERP) certification.
EDSNAStatus	-	Indicates if the address contains enough data for France's La Post National Address Management Service (SNA) certification.
EDSupplementaryGBStatus	-	Indicates whether GB country-specific output is available.
EDSupplementaryUSStatus	-	Indicates whether US country-specific output is available.
EECASSBARCODE	-	Contains the USPS barcode number for the address.
EECASSCARRIER_ROUTE	-	Contains the USPS carrier route for a US address.
EECASSCONGRESSIONAL_DISTRICT	-	Contains the congressional district based on the United States Postal Service (USPS) Zip+4 code.
EECASSDEFAULT_FLAG	-	Indicates if a US address matches a high-rise exact, high-rise default, or rural route default address in the reference data.
EECASSDELIVERY_POINT	-	Contains a unique Delivery Point Code (DPC) assigned to each USPS address in a Zip+4 code area.
EECASSDELIVERY_POINT_CHECK_DIGIT	-	Contains a number between 0 and 9 that, when added with the digits in the USPS Zip+4 and DPC, create a sum divisible by 10.
EECASSDPV_CMRA	-	Indicates if the address is for a USPS Commercial Mail Receiving Agent (CMRA), such as a post office box, and not the physical location of a business or residence.
EECASSDPV_CONFIRMATION	-	Contains a code indicating to what degree the USPS DPC value is valid.
EECASSDPV_FALSE_POSITIVE	-	Indicates whether an address was generated from encrypted USPS reference data.
EECASSDPV_FOOTNOTE_1	-	Indicates whether the address matches the USPS Zip +4 code data.

Informatica Address Verification 5 Output Field Name	Field #	Description
EECASSDPV_FOOTNOTE_2	-	Contains a DPV status code.
EECASSDPV_FOOTNOTE_3	-	Indicates whether an address is a USPS CMRA.
EECASSDPV_FOOTNOTE_COMPLETE	-	Contains the combined data for the DPV footnotes 1, 2, and 3.
EECASSDSF2_NOSTATS_INDICATOR	-	Indicates whether a USPS address is valid but undeliverable.
EECASSDSF2_VACANT_INDICATOR	-	Indicates whether a USPS address is inactive.
EECASSERRORCODE	-	Contains the CASS error code.
EECASSEWS_RETURNCODE	-	Indicates whether an address is in the USPS Early Warning System (EWS) list of new addresses that are not yet referenced to a ZIP+4 level.
EECASSHIGHRISE_DEFAULT	-	Indicates whether a USPS address matches a high-rise record in the address reference data and does not contain a unit identifier.
EECASSHIGHRISE_EXACT	-	Indicates whether a USPS address matches a high-rise record in the address reference data and also contains a unit identifier.
EECASSLACS	-	Indicates whether a United States address is present in the USPS Locatable Address Conversion Service (LACS) table.
EECASSLACSLINK_INDICATOR	-	Indicates if the address validator checks the address against the USPS LACS reference database.
EECASSLACSLINK_RETURNCODE	-	Indicates the degree to which the input address matches USPS LACS data and whether the address validation process updated the address.
EECASSRECORDTYPE	-	Contains additional information about the deliverable status of non-DPC USPS addresses.
EECASSRURALROUTE_DEFAULT	-	Indicates if the address is a valid rural route but exact data is unavailable.
EECASSRURALROUTE_EXACT	-	Indicates if the address matches a rural route address in the USPS address reference data set.
EECASSSUITELINK_RETURNCODE	-	Identifies high-rise business addresses in the United States that lack suite identification information.
EECASSZIPMOVE_RETURNCODE	-	Indicates if the USPS has recently changed the ZIP+4 Code assigned to the address.
EEGeoCodingCOMPLETE	-	Contains the complete geocode coordinates for the output address.

Informatica Address Verification 5 Output Field Name	Field #	Description
EEGeoCodingLATITUDE	-	Contains the latitude coordinate of the address.
EEGeoCodingLONGITUDE	-	Contains the longitude coordinate of the address.
EEGeoCodingLAT_LONG_UNIT	-	Contains the latitude and longitude unit of measure.
EESERPCATEGORY	-	Contains the SERP Certification status code.
EESNACATEGORY	-	Contains the SNA Certification status code.
EESupplementaryGBDELIVERY_POINT_ SUFFIXES	-	Contains the two-character suffix assigned to a mailbox in a UK post code area (Royal Mail).
EESupplementaryUSCBSA_ID	-	Contains a USPS Core-Based Statistical Area (CBSA) identification number. A CBSA identifies an urban area with a population greater than 10,000.
EESupplementaryUSCOUNTY_FIPS_CODE	-	Contains the US Federal Information Processing Standard (FIPS) Code that identifies a county or county equivalent in the United States and United States possessions.
EESupplementaryUSFINANCE_NUMBER	-	Contains the code assigned to United States post offices and other postal facilities to enable collection of cost and statistical data.
EESupplementaryUSMSA_ID	-	Contains the USPS Metropolitan Statistical Area identification number. This number identifies an urban area with a population greater than 50,000.
EESupplementaryUSRECORD_TYPE	-	Contains a single-character code that describes the type of mailbox.
EESupplementaryUSSTATE_FIPS_CODE	-	Contains the FIPS Code that identifies a state or state equivalent in the United States and United States possessions.
MDMLastError	-	The MDM Last Error output. Not associated with the Informatica Address Verification result.
RDElementInputStatus	-	Indicates the similarities between input address data and the address reference data.
RDElementRelevance	-	Indicates if an address element is required for postal delivery.
RDElementResultStatus	-	Contains a description of changes made to input address data elements during address validation.
RDMailabilityScore	-	Contains a single digit that represents the likelihood of successful delivery to the validated address, based on overall validation results.
RDResultPercentage	-	Indicates the percentage likelihood of successful delivery to the validated address.

Informatica Address Verification 5 Output Field Name	Field #	Description
RPCount	-	Contains the number number of results available.
RPCountOverflow	-	Indicates whether there are more results available than the 20 results that were returned.
RPCountryISO3	-	Contains the ISO3 country code.
RPModeUsed	-	Indicates which mode was used to process the address data.
RPPreferredLanguage	-	Identifies which language should be returned.
RPPreferredScript	-	Identifies which alphabet should be returned.
RPProcessStatus	-	Contains the process status value.

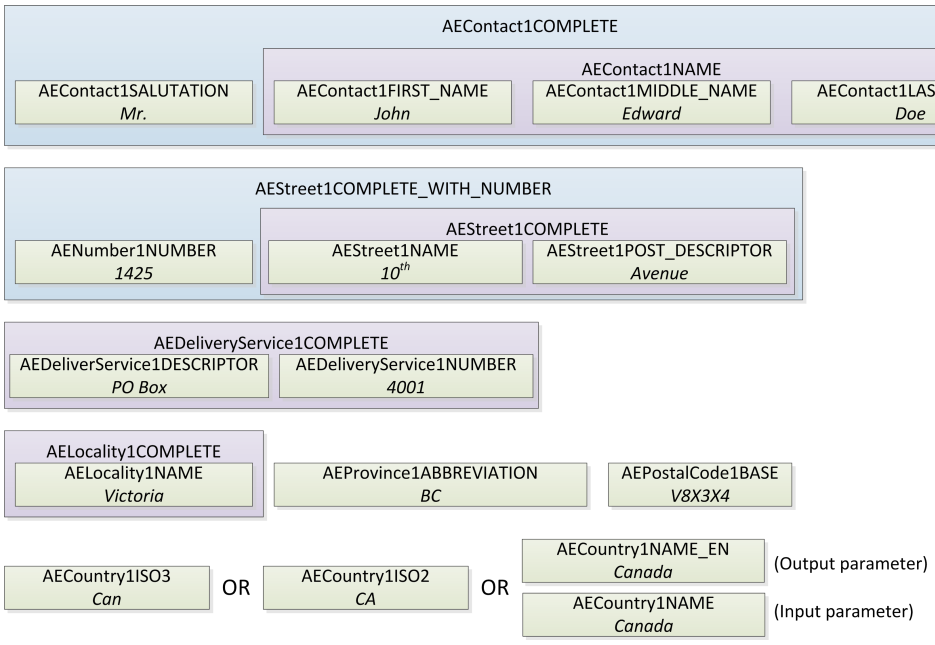
Address Examples

The following examples show how an address is broken down into specific address elements in the Informatica Address Verification 5 address element fields.

Example 1: Canadian Address

The sample Canadian address used in the example can be broken down into address element fields, as shown in the following illustration:

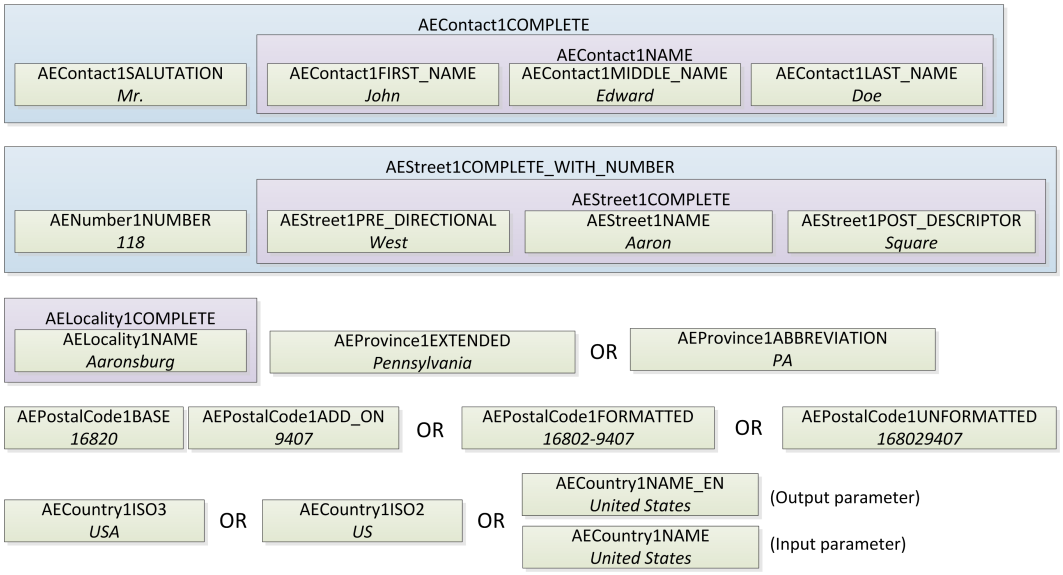
```
Mr. John Edward Doe
1425 10th Avenue
PO Box 4001
Victoria BC V8X 3X4
Canada
```



Example 2: US Address

The sample US address used in the example can be broken down into address element fields, as shown in the following illustration:

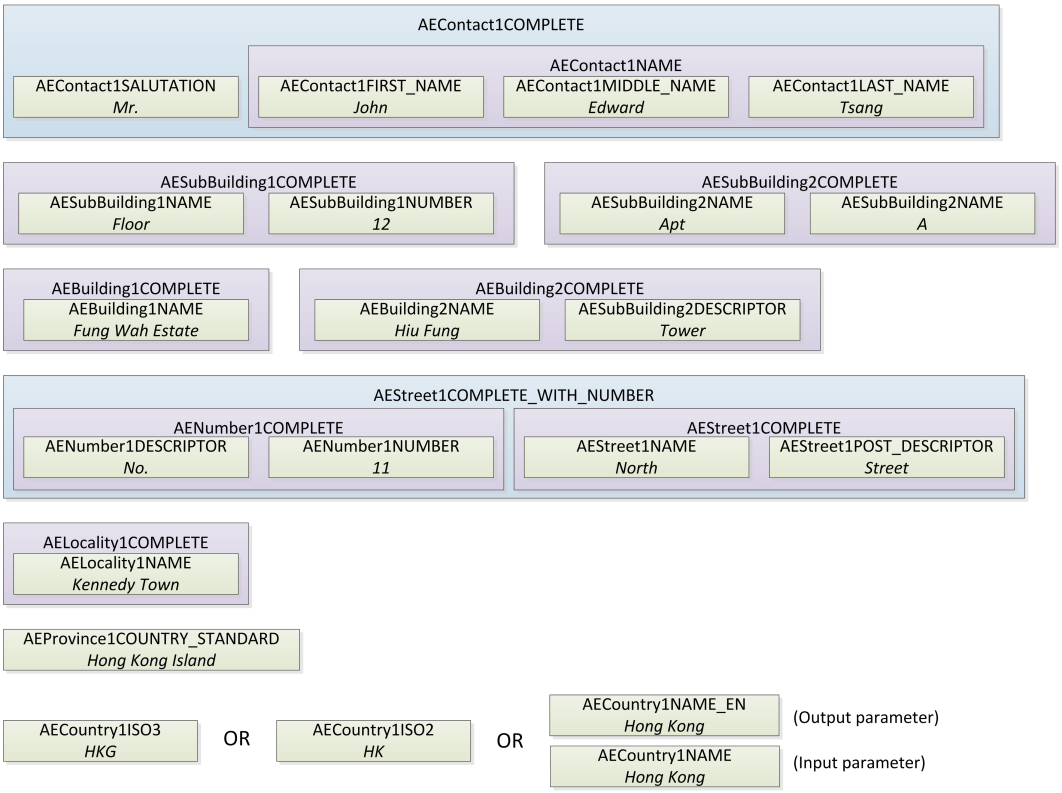
```
Mr. John Edward Doe
118 West Aaron Square
Aaronsburg PA 16820-9407
United States
```



Example 3: Hong Kong Address with Buildings and Subbuildings

The sample Hong Kong address used in the example can be broken down into address element fields, as shown in the following illustration:

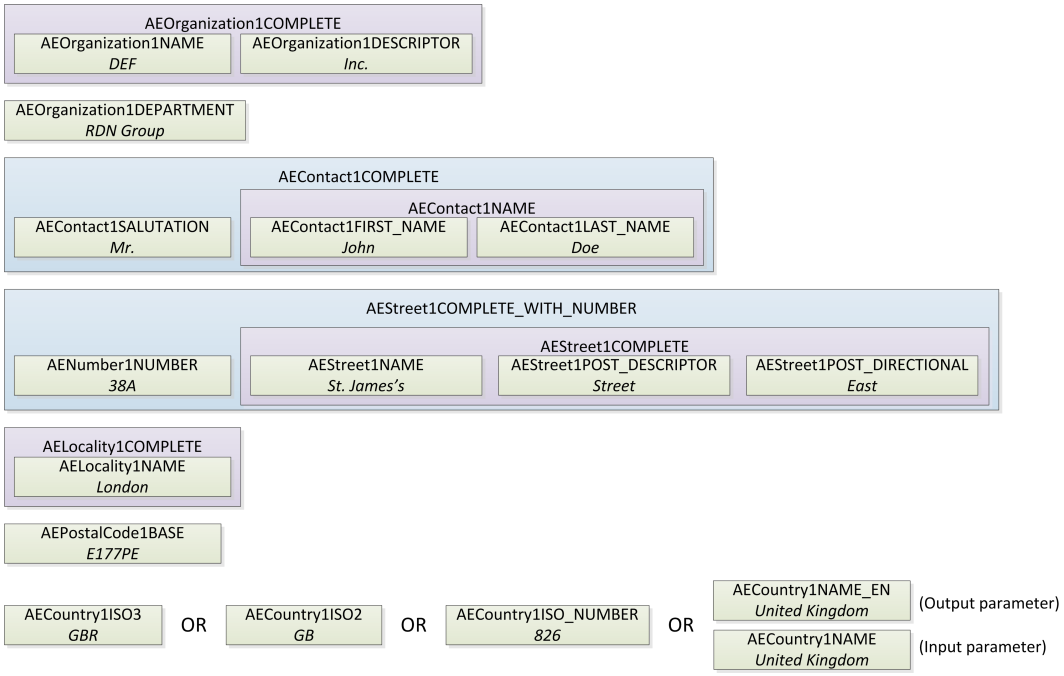
```
Mr. John Edward Tsang
Floor 12, Apt A
Fung Wah Estate, Hiu Fung Tower
No. 11 North Street
Kennedy Town
Hong Kong Island
HKG
```



Example 4: UK Business Address

The sample UK business address used in the example can be broken down into address element fields, as shown in the following illustration:

```
DEF Inc.
RDN Group
Mr. John Doe
38A St. James's Street East
London
E17 7PE
United Kingdom
```



Example 5: Using Address Line Fields to Simplify Address Entry

Inputting address data into address line fields is much easier than inputting the address into individual address elements. The following example shows how an address is broken down into address lines in the Informatica Address Verification 5 address line fields. The recipient, delivery address, and country-specific locality field names indicate the type of address data contained in the field. Use the formatted address line fields when the address data entered into a line field is not constant, such as when there are a combination of residential and business addresses. The type of address data in a given address line varies by country. Ensure that the address data is entered into the appropriate address line.

```
John Doe
123 Main St NW STE 12
Anytown NY 12345
```

ALRecipientLine1 **or**
ALFormattedAddressLine1
John Doe

ALDeliveryAddressLine1 **or**
ALFormattedAddressLine2
123 Main St NW STE 12

ALCountrySpecificLocalityLine1 **or**
ALFormattedAddressLine3
Anytown NY 12345

Differences Between Informatica Address Verification 4 and Informatica Address Verification 5 Process Status Values

The process status values indicate the reliability of the output address, the address verification result, and if the input address was corrected. Some process status values have changed in Informatica Address Verification 5 and some process status values were introduced to provide more detailed information.

The following table compares the Informatica Address Verification 4 and Informatica Address Verification 5 process status values.

Informatica Address Verification 4 Process Status Value	Informatica Address Verification 5 Process Status Value	Description
V	-	Verified. The input data is correct.
-	V4	Verified. The input data is correct. All relevant elements were checked and the input data matched the reference data.
-	V3	Verified. The input data is correct but some or all elements were standardized or the input contains outdated names or exonyms.
-	V2	Verified. The input data is correct but some elements could not be verified because of incomplete reference data.
-	V1	Verified. The input data is correct but incorrect element user standardization has deteriorated deliverability. For example, the postcode length chosen is too short. Not set by validation.
C	-	Corrected.
-	C4	Corrected. All relevant elements have been checked.
-	C3	Corrected. However, some elements could not be checked.
-	C2	Corrected. However, the delivery status is unclear due to a lack of reference data.
-	C1	Corrected. However, the delivery status is unclear because user standardization is incorrect. Not set by validation.
P3	-	Data cannot be corrected, but likely to be deliverable.
-	I4	Data could not be corrected, but is likely to be deliverable. An address element does not match the single reference value available. For example, the house number is incorrect and there is one house number in the reference data.
-	I3	Data could not be corrected, but is likely to be deliverable. An address element does not match the multiple reference values available. For example, the house number is incorrect and there is more than one house number in the reference data.
P2	I2	Data could not be corrected, but there is a slim chance that the address is deliverable.

Informatica Address Verification 4 Process Status Value	Informatica Address Verification 5 Process Status Value	Description
P1	I1	Data cannot be corrected and unlikely to be deliverable.
-	N5	Validation Error. Validation was not performed because the reference database is too old. Contact Informatica Address Verification to get updated reference data.
N4	-	Validation method not yet called after parsing operation.
-	N4	Validation Error. Validation was not performed because the reference database is corrupted or in the wrong format.
N3	N3	Validation Error. Validation was not performed because the country could not be unlocked.
N2	N2	Validation Error. Validation was not performed because the required reference database is not available.
N1	N1	Validation Error. Validation was not performed because the country was not recognized.
Q3	Q3	FastCompletion Status. The suggested address is complete.
Q2	Q2	FastCompletion Status. The suggested address is complete. However, elements were deleted or added from the input.
Q1	Q1	FastCompletion Status. The suggested address is not complete. Enter more information.
Q0	Q0	FastCompletion Status. Suggestions cannot be generated. The input data is insufficient.
-	S4	Parsed perfectly.
-	S3	Parsed with multiple results.
-	S2	Parsed with errors. Elements changed position.
-	S1	Parse error. Input format mismatch.
-	RB	Country recognized from abbreviation.
-	RA	Country recognized from ForceCountryISO3 setting.
-	R9	Country recognized from DefaultCountryISO3 setting.
-	R8	Country recognized from name without errors.
-	R7	Country recognized from name with errors.
-	R6	Country recognized from territory.

Informatica Address Verification 4 Process Status Value	Informatica Address Verification 5 Process Status Value	Description
-	R5	Country recognized from province.
-	R4	Country recognized from major town.
-	R3	Country recognized from format.
-	R2	Country recognized from script.
-	R1	Country not recognized. Multiple matches.
-	R0	Country not recognized.

Configuring the JVM Settings

If you use Informatica Address Verification as your cleanse engine to run any batch process, you must ensure that the stack size for the JVM is sufficient.

If an incorrect stack size is set for your application server, Informatica Address Verification might cause an initialization failure when you start the Process Server. Also, it might throw an exception during a cleanse operation when the Process Server is up and running.

Setting the JVM Size for WebSphere on Windows/UNIX

To set the JVM size for WebSphere:

1. Open the WebSphere Console.
2. Navigate to Servers> Application Server> <*Your Server*>>Process Definition> Java Virtual Machine.
3. Add the following parameters to the Generic JVM Arguments:

```
-Xss1024k - Parameter to set JVM stack size of 1 MB
-Xms512m - Parameter to set JVM heap with a start (minimum) size of 512 MB
-Xmx1024m - Parameter to set JVM heap to allow growth to maximum of 1 GB
```

4. If you use Informatica Address Verification 5.4 or later, add the following parameter to the Generic JVM Arguments:

```
-Xmso2048k - Parameter to increase the operating system thread stack size
```

5. Save the configuration.
6. Restart the WebSphere application server profile that runs the Process Server.

Setting the JVM Size for WebLogic Server on Windows

To set the JVM size for the WebLogic server on Windows:

1. Go to your WebLogic home directory.
2. Open `setDomainEnv.cmd` in a text editor.

3. Set the `MEM_ARGS` variable as follows:

```
set MEM_ARGS=%MEM_ARGS% -Xmx1024m
```

4. Save and close the `setDomainEnv.cmd` file.

Setting the JVM Size for WebLogic Server on UNIX

To set the JVM size for WebLogic server on HP-UX and Solaris:

1. Go to your WebLogic home directory.
2. Open setDomainEnv.sh in a text editor.
3. Set the `MEM_ARGS` variable as follows:

```
set MEM_ARGS=$MEM_ARGS -Xmx1024m
```

4. Save and close the `setDomainEnv`.sh file.

Setting the JVM Size for JBoss on Windows

To set the JVM size for the JBoss server on Windows:

1. Go to your JBoss home directory.
2. Navigate to the `bin` directory.
3. Open `run.bat` in a text editor.
4. Set the JAVA_OPTS variable as follows:

```
JAVA_OPTS=%JAVA_OPTS% -Xmx1024m
```

5. Save and close the `run.bat` file.

Setting the JVM Size for JBoss on UNIX

To set the JVM size for JBoss on UNIX:

1. Go to your JBoss installation directory.
2. Navigate to the `bin` directory.
3. Open `run.sh` in a text editor.
4. Set the JAVA_OPTS variable as follows:

```
JAVA_OPTS="$JAVA_OPTS -Xmx1024m
```

5. Save and close the `run.sh` file.

   After you make configuration changes, restart your Informatica MDM Hub and Process Servers.

CHAPTER 4

Trillium Director Cleanse Engine

This chapter includes the following topics:

Trillium Director Cleanse Engine Overview

Integration between Informatica MDM Hub and the Trillium Director cleanse engine occurs through the Trillium Director Adapter. This adapter is an optional component.

This chapter explains how to configure your Informatica MDM Hub system to use the Trillium Director Adapter and the Trillium Director cleanse engine. This chapter assumes that you are knowledgeable about configuring and using the Trillium Director software. To learn more about Trillium Director, see your Trillium Director documentation.

Note: Ensure that you have the latest Trillium patches installed before you work with the Trillium Director Adapter. Contact Informatica Global Customer Support for more information.

About Trillium Director Integration

Informatica MDM Hub integrates with Trillium Director by treating Trillium Director projects as cleanse functions. Informatica MDM Hub uses a Trillium Director configuration file to determine the Trillium Director

projects that are available for integration into the Hub Console library. The default configuration files are located in the following directory:

- Windows:

  `<infamdm_install_directory>\cleanse\resources\Trillium\samples\director\`

- UNIX:

  `<infamdm_install_directory>/cleanse/resources/Trillium/samples/director/`

The process of integrating with Trillium Director involves installing and configuring Trillium Director for use with Informatica MDM Hub.

To integrate the Trillium Director cleanse engine with the Process Server:

1. Ensure that the prerequisites for Trillium Director Adapter are met.
2. Install Trillium Director. To learn more, see your Trillium Director documentation.

   Important: We recommend that you install the Process Server and Trillium Director on the same machine for improved performance.
3. Install your application server. To learn more, see your application server documentation.
4. Install Process Server as explained in the *Multidomain MDM Installation Guide*.
5. Configure Trillium Director to use the Informatica MDM Hub.
6. Check your installation and configuration.

Before You Install

Before you install:

1. Ensure that your Informatica MDM Hub license has the cleanse adapter and the Trillium Director Adapter enabled. Contact Informatica Global Customer Support to obtain this license.
2. Ensure that the following Trillium Director components are installed on the same machine as the Process Server:
   - Trillium Director Server
   - Trillium Director data files for the countries with which you are working. Contact Trillium for these files.

Configuring Trillium Director and the Cleanse Match Server

In order to successfully integrate Trillium Director 11 or later with the Informatica MDM Hub, you must configure both Trillium Director 11 or later and the Process Server to work together.

1. Set the Trillium Director port, so that it does not conflict with any other port that is used on the machine. To learn more, see your Trillium Director documentation.
2. Go to the following directory:

On Windows:

```
<Cleanse_Install_Directory>\resources\TrilliumDirector11\samples\director\
```

On UNIX:

```
<Cleanse_Install_Directory>/resources/TrilliumDirector11/samples/director/
```

Check to ensure you have the following files:

- td11_default_config_Global.txt
- td11_default_config_US_detail.txt
- td11_default_config_US_summary.txt

These files define the Trillium Director projects you use with the Process Server.

If you do not have these files, contact Informatica Global Customer Support.

Note: You can choose to create your own configuration files.

3. Open the `cmxcleanse.properties` file for editing:

   On Windows:

   ```
   <Cleanse_Match_Server_Install_Dir>\resources\cmxcleanse.properties
   ```

   On UNIX:

   ```
   <Cleanse_Match_Server_Install_Directory>/resources/cmxcleanse.properties
   ```

4. Check the following values for your Trillium Director version.

   Sample values for Trillium Director 11:

   ```
   cleanse.library.trilliumDir11.property.config.file.1=<infamdm_install_directory>/
   hub /cleanse/resources/TrilliumDirector11/samples/director/
   td11_default_config_Global.txt
   cleanse.library.trilliumDir11.property.config.file.2=<infamdm_install_directory>/
   hub /cleanse/resources/TrilliumDirector11/samples/director/
   td11_default_config_US_detail.txt
   cleanse.library.trilliumDir11.property.config.file.3=<infamdm_install_directory>/
   hub /cleanse/resources/TrilliumDirector11/samples/director/
   td11_default_config_US_summary.txt
   ```

5. Save and close the properties file.
6. Connect to the database as the schema owner and run the following statement to enable Trillium Director:

   ```
   update C_REPOS_CL_FUNCTION_LIB set DISPLAY_IND = 1 where function_lib_name =
   'Trillium Director Version 11'
   ```

7. Add the Trillium Director `bin` or `bin64` directory path to the `CLASSPATH` and the `PATH` environment variables:

   - On 32-bit Windows:

     <Trillium Director install directory>\TSQuality*<Trillium version>*\Software\bin

   - On 64-bit Windows:

     Trillium Director install directory\TSQuality*<Trillium version>*\Software\bin64

   - On UNIX:

     /apps/oracle/TrilliumSoftware/tsq11r5s/Software/bin

8. Also check that the `TRILLDIRPORT` and `TRILLDIRADDR` environment variables are set to an appropriate value. For example:

   ```
   TRILLDIRADDR=localhost
   TRILLDIRPORT=14445
   ```

9. If you have not done so already, start the Trillium Director and Trillium Cleansing Server services.
10. Start the Trillium Director and Trillium Cleansing Server services.

Testing Your Trillium Director Configuration

To test your Trillium Director configuration:

1. Go to the Informatica MDM Hub console.
2. Choose the Cleansing tool from the workbench.
3. Choose cleansing functions.
4. Refresh the Trillium Director library by using the Refresh tab on the right-hand side in the Hub Console. You should see your Trillium Director functions.

   If you do not see those functions, check to ensure that you have successfully completed the steps in previous configuration sections.

You can now use these functions to make a mapping and cleanse your data.

Sample Configuration Files

Use the configuration files to configure information such as the function name, server name, and the input and output parameters.

You can use the sample configuration files, or create your own configuration files. If you use a sample configuration file, update the function name, system ID, and server name for your environment. Informatica supplies the following sample configuration files that work with Trillium Director 11 and later:

td11_default_config_Global.txt

Provides output fields for international addresses.

td11_default_config_US_detail.txt

Provides output fields for U.S. addresses.

td11_default_config_US_summary.txt

Provides the most common output fields for U.S. addresses.

The sample configuration files are in the following locations:

- On Windows:

  ```
  <infamdm installation directory>\hub\cleanse\resources\TrilliumDirector11\samples
  \director\
  ```

- On UNIX:

  ```
  <infamdm installation directory>/hub/cleanse/resources/TrilliumDirector11/samples/
  director/
  ```

The following information is in the sample configuration files:

- function name
- function description
- input parameters
- output parameters
- update parameters

The following table describes the entries in this configuration file:

Name	Value
TD_FUNCT_NAME	The name of your Trillium Director function.
TD_FUNCT_DESCR	A description of the function. This appears when you look at the available functions using the Cleansing Tool.
TD_SYSTEM_ID	The Trillium Director system ID. This defines which Trillium Director process to use.
TD_SERVER_NAME	The name of the server process within the Trillium Director system indicated in `TD_SYSTEM_ID`. This is the name of the Trillium Director server that is running the project you are using. This is the Cleanser instance where your project was deployed.
TD_INPUT_PARAM_*	This is a group of entries, one for each input parameter for the function. Add or remove entries as necessary.
TD_OUTPUT_PARAM_*	This is a group of entries, one for each output parameter for the function. Add or remove entries as necessary.
TD_UPD_PARAM_*	This is a group of entries, one for each update parameter for the function. Add or remove entries as necessary.

The following example shows a sample configuration file for Trillium Director 11 and later:

```
TD_FUNCT_NAME = TrilliumDirectorGlobal
TD_FUNCT_DESCR = Trillium Director adapter

TD_SYSTEM_ID = G
TD_SERVER_NAME = Cleanser

TD_INPUT_PARAM_0 = MRMRS
TD_INPUT_PARAM_1 = BUSINESSNAME
TD_INPUT_PARAM_2 = StreetAddress
TD_INPUT_PARAM_3 = StreetAddress2
TD_INPUT_PARAM_4 = StreetAddress3
TD_INPUT_PARAM_5 = StreetAddress4
TD_INPUT_PARAM_6 = City
TD_INPUT_PARAM_7 = State
TD_INPUT_PARAM_8 = PostalCode
TD_INPUT_PARAM_9 = Country

-- Derived Global Output Fields (country specific output fields also available)

TD_OUTPUT_PARAM_0 = DR_HOUSE_NUMBER1
TD_OUTPUT_PARAM_1 = DR_STREET_NAME
TD_OUTPUT_PARAM_2 = DR_HOUSE_NUMBER2
TD_OUTPUT_PARAM_3 = DR_CITY_NAME
TD_OUTPUT_PARAM_4 = DR_REGION_NAME
TD_OUTPUT_PARAM_5 = DR_POSTAL_CODE
TD_OUTPUT_PARAM_6 = DR_ADDRESS
TD_OUTPUT_PARAM_7 = DR_COUNTRY_NAME
TD_OUTPUT_PARAM_8 = DR_ADDR_MAIL
```

Upgrading Trillium Director

You can upgrade the Trillium Director cleanse engine if required.
Note: When you upgrade the Informatica MDM Hub installation, it is not required to upgrade Trillium Director.

1. If upgrading to Trillium 11 or later versions, obtain an updated license file.
2. Edit the Hub Cleanse Match properties file.

Using Trillium on a Remote Server

You must configure Trillium Director before it can run on a remote server.

1. Copy the Trillium Director files from the remote server to the MDM Hub Process Server.
   - The following table shows the files and server paths for Windows 32-bit systems:

Files/Paths	Description
Files	`TGenClient.dll` `xerces-c*.dll`
Remote server path	`<Trillium Director install directory>\TSQuality\<Trillium version>\Software\bin`
Process Server path	`<infamdm install directory>\hub\cleanse\lib`

   - The following table shows the files and server paths for Windows 64-bit systems:

Files/Paths	Description

Files	`TGenClient.dll`
	`xerces-c*.dll`
	`triTGenClientLibrary.dll`

Remote server path	`<Trillium Director install directory>\TSQuality\<Trillium version>\Software\bin64`
Process Server path	`<infamdm install directory>\hub\cleanse\lib`

- The following table shows the files and server paths for UNIX systems:

Files/Paths	Description
Files	libTGenClientLibrary.so libexerces-c*.so
Remote server path	<*Trillium Director install directory*>/TSQuality/<*Trillium version*>/Software/bin
Process Server path	<*infamdm install directory*>/hub/cleanse/lib

2. Ensure all the file permissions are set correctly for user and group:
   - Add <*infamdm_install_directory*>\cleanse\lib to the LIBPATH or LD_LIBRARY_PATH environment variable.

     Note: If using the WebSphere application server, do not put the LIBPATH or LD_LIBRARY_PATH environment variable into the classpath of JVM. For example, **Application servers** > **server1** > **Process Definition** > **Java Virtual Machine: Classpath variable**.
   - Ensure the following Trillium environment variables are set correctly:

```
TRILLDIRADDR=<trillium_server_machine_address>
TRILLDIRPORT=<Trillium server port>
TRILLCONFIG=<Trillium Director install directory>\TSQuality\Trillium version
\director_proj\settings\TrilXML.cfg
TS_CONFIG: <Trillium Director install directory>\TSQuality\<Trillium version>
\director_proj\settings\Config.tbl
TS_QUALITY=<Trillium Director install directory>\TSQuality\<Trillium version>
\Software
```

Configuring Trillium Director for Multithreading

If you have the Trillium Director installed on a machine with more than one processor, you can use multithreading to take advantage of your hardware.

To configure your Trillium Director for performance:

- Turn on caching for performance benefits.
- By default, Trillium Director is set to be single-threaded. Trillium Director reverts to this default if you set an invalid value for the number of threads, such as a number equal to or less than 0 or a very large number. Set the number of threads to equal the number of CPUs in the machine where Trillium Director is installed.

You can confirm the thread settings by turning debug on. This shows you threads that are spawned and other information. To learn more, see your Trillium Director documentation.

Important: Turn debug off when you are in a production environment. Debug can have a significant negative effect on performance.

Setting the Threading Pool

When the Trillium adapter works with large data sets, it uses thread-pool connections in the application server. These threads are shared by all the jobs that the server handles. The thread-pool level is designated in the following property: `cleanse.library.trilliumDir.property.td.project.pool.MaxActive=nn` where nn is the same or slightly higher than the total thread count of all jobs running in parallel.

The default setting for the `MaxActive` property allows as many connections as are requested. To restrict the number of connections, reset the `MaxActive` property value.

Increasing the Number of Network Connection Retries

When network traffic is heavy and a high number of cleanse threads are being used, the network connection for a thread between the Informatica MDM Hub and the Trillium server can time out while a record is being processed. The server automatically tries to restart the connection five times, and issues an error message after the fifth try if the connection attempt fails.

Note: The batch cleanse still runs through, with the record marked as rejected.

The number of restart tries is determined by the following property, which can be added to the `cmxcleanse.properties` file:

cleanse.library.trilliumDir.property.set_maximum_retry_count=15

The default value for this property is five (5). To ensure that the Hub tries to establish the network connection enough times to account for network traffic, set this property to a high number (such as 15).

CHAPTER 5

Troubleshooting

This chapter includes the following topics:

Informatica Address Verification Initialization Issues

If you cannot initialize Informatica Address Verification after you install the cleanse match server, use the following suggestions to troubleshoot the issue:

1. Check the `cmxserver.log` file to confirm that Informatica Address Verification initialization failed.
2. Ensure that your Informatica MDM license has Informatica Address Verification enabled. If it does not, contact Informatica Support. When you obtain the correct license, restart your application server and begin the installation again.
3. Try rebooting your machine (Windows) or refreshing your environment (UNIX).
4. UNIX: Ensure that the SSAPR environment variable has been added to your environment, and that it points to the current Informatica MDM files. Restart the application server.
5. Ensure you have the correct `jade.dll` file (Windows) or `libJADE.so` file (UNIX) and that it is the correct size, then restart the application server.

   Note: Check the system path for older `jade.dll` files and delete them. Accumulation of these older files is a common occurrence in development environments.

Cleanse Engine Initialization Fails

This section discusses workarounds and tips for initialization failures.

In general, cleanse issues fall into the following categories:

- Issues that occur during the application server initialization:
  - Lack of a license file.
  - Erroneous or missing cmxcleanse.properties file entries.
  - Errors when loading system files.
- Issues that occur when the Hub is making calls to the cleanse tool in batch or real time:
  - Uninitialized adapter.
  - Port conflicts.
  - No response from the cleanse tool server.

Remote Initialization Fails

If your remote initialization (initializing cleanse engines that are on a remote machine) fails, use the following suggestions to troubleshoot the issue:

- Ensure that the cleanse engines are running, especially if the cleanse engine you are using has a service associated with it.
- Check that your properties file already points to the specific machine that is running that cleanse engines.
- Restart the application server.
- Ensure the remote machine running the cleanse engine services is started before the application server is turned on.

Trillium Errors

In general, Trillium errors fall into the following categories:

- Lack of read, write, execute permissions
- Port conflicts
- Missing Trillium Director client library `TGenClient.lib`. This library should exist on the same machine that is running the Process Server.

Initialization Fails for MDM Console

If your Trillium initialization fails for the MDM Console, (such as when you get the following error: `Cannot borrow data from the POOL`), try stopping the application server and the Trillium Director services. Restart the Trillium Director services first, then the application server.

Note: See the Informatica Knowledge Base (https://communities.informatica.com) for additional information.

Remote Initialization Fails

If your remote initialization (initializing cleanse engines that are on a remote machine) fails, use the following suggestions to troubleshoot the issue:

- Ensure that the cleanse engines are running, especially if the cleanse engine you are using has a service associated with it.
- Check that your properties file already points to the specific machine that is running that cleanse engines.
- Restart the application server.
- Ensure the remote machine running the cleanse engine services is started before the application server is turned on.

Trillium Director Integration

The primary tool for troubleshooting your Trillium Director integration is the working files written by Trillium Director. You can also check the Process Server files.

About Trillium Director Working Files

When a batch stage job is executed, the Process Server saves working files. When a Trillium Director function is used, a Trillium Director file is created for the purpose of auditing and troubleshooting.

Finding the Location of the Trillium Director Work Files

Trillium Director writes logs and work files to the logs directory and other places defined by the project. By default, the logs directory is:

- Windows:

  ```
  C:\tril7v8\Director\logs
  ```

- UNIX:

  ```
  /u1/tril7v8/Director/logs
  ```

To find the location of the work files, look at the following parameter in the properties file:

```
cmx.server.datalayer.cleanse.working_files.location
```

Setting the Location of the Process Server Work Files

Process Server writes working files to a directory specified in the `cmxcleanse.properties` file.

To set the location of the work files, set the following parameter in the properties file:

```
cmx.server.datalayer.cleanse.working_files.location
```

Setting Whether Working Files are Kept

To set whether work files are kept, set the following parameter in the cmxcleanse.properties file:

```
cmx.server.datalayer.cleanse.working_files=KEEP
```

Set this property value to `FALSE` to discard these working files. If you specify that the working files are not kept, the value for the location of the files is irrelevant.

Setting the Number of Connections in a Connection Pool

When you use large data sets with the Trillium adapter, the adapter uses a pool of connections, which is shared by all jobs in an application server.

The following parameter controls the pool level:

```
cleanse.library.trilliumDir.property.td.project.pool.MaxActive=nn
```

By default, the adapter allows as many connection attempts as are requested. However, you can set this parameter if you must restrict the connections.

INDEX