Infor WMS Automation Agent Administration Guide

Version 11.4.x

Important Notices

Trademark Acknowledgements

Publication Information

Release: Infor WMS Automation Agent Release level 11.4.x

Publication date: February 19, 2021

Contents

About this guide

This guide provides information for the configuration and administration of Infor WMS Automation Agent. This guide shows you how to:

- Identify components of Automation Agent
- Configure Automation Agent

Intended audience

This guide is intended for the system administrator or consultant who configures and administers Automation Agent for use with Infor WMS.

Prerequisite knowledge

To fully understand the information presented in this guide, you should first be familiar with these concepts and tools:

- Spring class loaded configurations
- Grafana data visualization tool

Related documents

You can find the documents in the product documentation section of the Infor Support Portal, as described in "Contacting Infor."

- *Infor WMS Automation Agent Installation Guide*
- *Infor WMS Automation Agent Training Guide*
- *Infor WMS Automation Agent Configuration Guide for Components*
- *Infor WMS Automation Agent Platform Support Matrix*

Contacting Infor

If you have questions about Infor products, go to Infor Concierge at https://concierge.infor.com/ and create a support incident.

If we update this document after the product release, we will post the new version on the Infor Support Portal. To access documentation, select **Search > Browse Documentation**. We recommend that you check this portal periodically for updated documentation.

If you have comments about Infor documentation, contact documentation@infor.com.

Chapter 1 Overview

Infor WMS Automation Agent directs real-time activities within warehouses and distribution centers (DC) by performing as a data facilitator between applications and material handling equipment (MHE). Multiple connection types, such as Database, API, and Socket, File, can be implemented. Automation Agent builds a library of connection types and feature flows as you continue to use the product. Automation Agent provides a uniform interface to material handling equipment such as Automated Guided Vehicles (AGV), AS/RS, carousels, conveyor systems, sorters, and palletizers.

Grafana is used as the current interface to support application monitoring and for issue resolution. These are the primary functions of Automation Agent:

- Interface to an upper level host warehouse management system (WMS) and exchange information required to manage the daily operations of the distribution center.
- Allocate work to the various material handling sub-systems to balance system activity to complete the requested workload.
- Provide real-time directives to operators and material handling equipment controllers to accomplish the order fulfillment and product routing requirements.
- Dynamically assign cartons to divert locations based on defined sortation algorithms or based on routing/order information received from the host (if applicable).
- Generate result data files for reporting and/or upload by the host system.
- Provide operational screens and functions to facilitate efficient control and management of the distribution warehouse.
- Collect statistical data on the operational performance of the system to enable operations personnel to maintain the equipment in peak performance.

Each major function is designed to work as part of an integrated process to effectively link the host systems with the lower level control system, while relieving the host from the real-time requirements such as operator screens and lower level equipment control interfaces.

Terminology

Automation Agent is a stand-alone Java application that uses initiators and processors to receive, transform, and transmit messages.

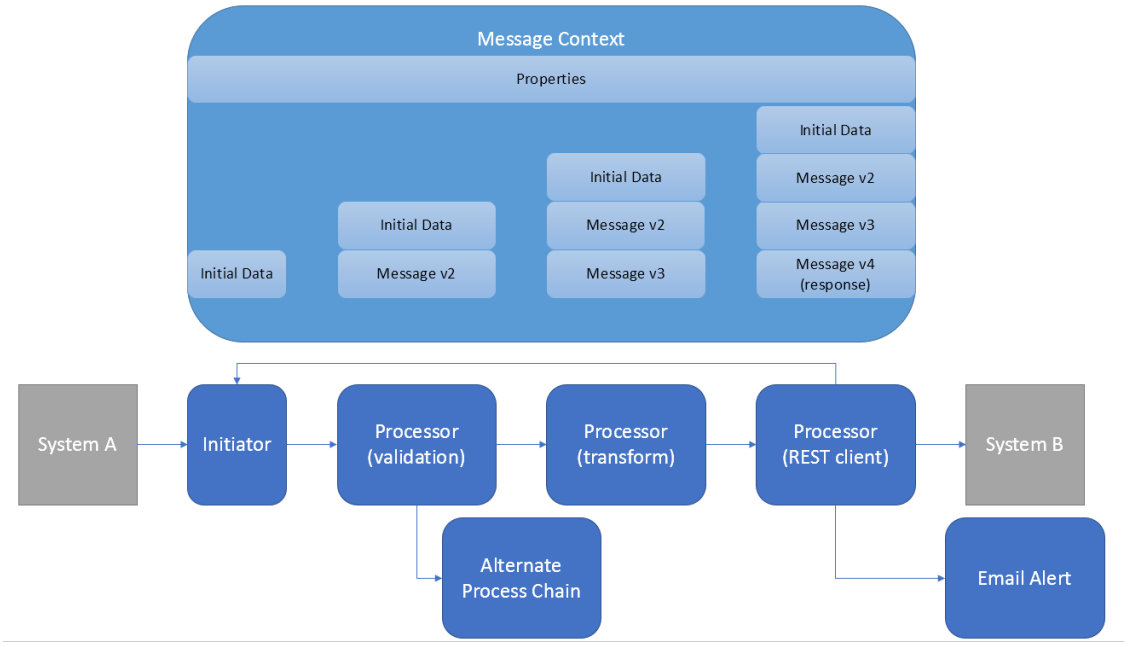
Term	Definition
Initiators	These are entry points into a flow called a Process Chain. The initiators are either pollers or listeners. A poller periodically checks for data in a file or database, while a listener waits for a message to be pushed to it by an active client.
Processors	Processors perform actions on the message including lookup and validation, transformation, changing the process chain or message flow based on data, and pushing the message to other systems. As the message is transported from the initiator through the various processors, a copy of each message state is maintained and properties are set and accessed by the various processors on a persisted context scoped for each individual message.
Alerts	These are exceptions encountered along the process and are raised to alert processes that can write logs or send emails to configurable recipients filtered by severity and functional area.

Process Flow Overview

This diagram models a basic process chain:

1. System A pushes a message to an initiator.
2. The initiator creates the initial message context with properties and a message stack.
3. The context is handed off to a lookup and validation processor that will add or modify properties and add an updated message to the stack.

   If the processor finds certain information in the message, it may redirect to a new process chain.
4. The context is handed to a transform service that will again update properties and add a new message in a different format to the stack.
5. The context is handed to a REST client processor that will post the transformed message to System B.
6. If an exception is thrown or returned, an alert is published to an email alert publisher. This can be configured. Control is then returned to the initiator in case a synchronous response is required.


Message Context
Properties
Initial Data
Initial Data
Message v2
Initial Data
Message v2
Message v3
Initial Data
Message v2
Message v3
Message v4 (response)
System A
Initiator
Processor (validation)
Processor (transform)
Processor (REST client)
System B
Alternate Process Chain
Email Alert

Chapter 2 Automation Agent Components

Automation Agent uses a Spring context class loader. These files can be configured in any format. However, we recommend that you follow our format. If you do not use our recommended formats, you may impact future upgrades.

Configuration Files

This section describes the Automation Agent files:

- `agent.xml`: This file is the initial file called to load the Spring context. This file should contain:
  - Imports for other files
  - A single bean named controller that contains a list of initiators and a list of alerts to load at startup.
- `initiators.xml`: This file contains beans for all initiators and their configuration entries. An initiator listens for a request or actively monitors a resource and initiates a process chain when data is received or found.
- `processchains.xml`: This file contains beans for all process chains. A process chain is an ordered series of steps to perform including processors or alerts and is started by a call from an initiator.
- `processors.xml`: This file contains beans for all processors. A processor is a piece of code that performs a step based on the current message state. This can include data validations, writing to files, calling web services, raising alerts, or more.
  - `datamaps.xml`: This file contains beans defining maps from one data structure to another. These beans are not executed directly. They are called by a process.
  - `lookupvalidations.xml`: This file contains beans that allow one value to be looked up based on another, for example, to convert codes from one system to another, default values, and perform data validation. These beans are not executed directly. They are called by a process.
- `alerts.xml`: This file contains beans for supported alerts. Alerts can be referenced in the process chain to write to the WM Event Log screen, or to send emails. Emails can be filtered by several parameters on a per user basis, The users and filters are defined in this file.
- `logback.xml`: This is a standard LogBack configuration file. We recommend that the log level is set to **`WARN`**.

Automation Agent monitor interface:

Grafana is a third-party application that is used by Automation Agent to display instrumentation data that is created by the agent. You can use this tool to see detailed data about every step in a live message flow. Additionally, graphs are provided to see message volume over time.

Automation Agent Initiators:

Initiators are components that look for updated data or listen for a client to send data and start a new message process when data is found or received.

These are the supported initiators:

- File**:** The file initiator polls a directory for files or scan files for an extension type. The file initiator is used to pick up a file, read its contents, and pass those contents off to a processor.
- Socket (Server): The socket server initiator reads in a fixed length string, passes that string off to a processor, and writes the response on the socket after the end of the process chain is reached.
- Socket (Client): The socket client initiator reads in a fixed length string and passes the string off to a processor for further processing.
- FTP Client: The FTP Client initiator scans a directory for all files or an extension type to select for transferring to an FTP server.
- FTP Retrieve
- DB**:** The DB initiator will allow for the retrieval of information directly from a database.
  - Supports Header / Detail tables with or without 3rd table for initiating (Similar to the current Infor WMS process using the transmitlog table).
- JMS Listener: Listens on configured JMS queue for messages.
- Generic Poller: Polls with pre-built initial message/properties for `ExportInterface` calls with REST Client.

Automation Agent Processors:

Processors are components that manipulate message data, inspect data to make decisions about the next steps to perform, or call out to external process to write data or retrieve additional information.

These are the supported processors:

- DB: Supports header and detail level SQLs to read or update data.
- File Utilities

- Router (manages process chain runtime changes): Allows the message to be inspected and the process chain to be modified based on that message content. This allows inspection of types to run different processes or validations to potentially raise an alert.
- REST Client: Makes a REST web service call to a server. Supports OAuth 1 and 2.
- XPath Parser: Allows data elements to be selected from an XML structure to set on the message context. Example, you can pull a message type out of the XML and put it in the context headers for easy access in later steps.
- Lookup Validation: Performs lookups based on lists or validations based on datatypes, min / max values, or existence within a list.

Automation Agent Alerts:

Alerts are components that send important information to a user. The alerts can be exception information or information configured in a process flow such as the receipt of a priority order.

These are the supported alerts:

- Simple Email: Sends an email to a list of recipients. Each recipient can filter the emails that they receive based on severity, priority, and functional area.
- Event Log: Writes alert information to the WM Event Log using the REST interface.

Automation Agent Logging:

Logging is the mechanism used to determine what the Automation Agent is doing at any given time. Logging is useful for monitoring the interface or troubleshooting issues.

These are the supported logging mechanisms:

- Console Logger: This is useful during development. The Console Logger provides the current message content to the console. This should not be used during production. During production, you should rely on your instrumentation and logback.
- File Logger: Similar to the console logger, this is more useful when running as a WildFly service. This should be the standard even during development because it logs to a file instead of STDOUT. This also should not be in a production deployment.
- LogBack: Automation Agent uses standard LogBack logging. Use this for troubleshooting. The recommended log level in production is **`WARN`**.
- Instrumentation: This provides detail on what the Automation Agent is doing at any point in time and should be **`ON`** for production in most situations. See the Instrumentation on page 5 for detail.

Chapter 3 Instrumentation

Automation Agent stores runtime execution information in an Instrumentation database that can be accessed later to monitor system throughput or view and troubleshoot individual message actions.

To configure Instrumentation for Automation Agent, you must modify the `initiators.xml` file. Each initiator that is to use instrumentation will require two parameters added to connect to the PostgreSQL database. We recommend that instrumentation is enabled for all initiators.

To enable Instrumentation:

1. Go to your installation's configuration folder and open the `initiators.xml` file. The default location is `C:\infor\aa\AAPRD\agent1\conf`
2. Add the property `jndinameInstr` with a value matching the jndi name set up in your WildFly `standalone.xml` for your datasource.
3. To toggle on or off instrumentation for each initiator, set the property `instrumentationOn` to **`true`** or **`false`**. This property should always be set to **`true`** unless there is a specific reason that you cannot run instrumentation.

   Note: Running without instrumentation can make troubleshooting difficult.

initiators.xml

```
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:util="http://www.springframework.org/schema/util"
    xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-3.0.xsd
        http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-2.0.xsd">

    <bean id="socketListenerPLC" class="com.infor.cswms.automationagent.initiator.SocketClientInitiator">
        <!-- client side hostName -->
        <property name="hostName" value="127.0.0.1"/>
        <!-- client side port -->
        <property name="port" value="4011"/>
        <!-- position to start parsing message type -->
        <property name="messageTypeStart" value="18"/>
        <!-- position to stop parsing message type -->
        <property name="messageTypeEnd" value="22"/>
        <!-- filler character to be used for fixed length processing -->
        <property name="paddingChar" value="*"/>
        <!-- The name of the process chain the initiator will follow after initial parse -->
        <property name="processChain" value="RBGStatuscheckPLC"/>
        <!-- a hashmap containing the positions for a particular messagetype -->
        <property name="messageTypeInboundPositionMap" ref="PLCinboundPositionMap" />
        <!-- a hashmap containing the names to be assigned to data for a particular messagetype -->
        <property name="messageTypeInboundValueMap" ref="PLCinboundValueMap" />
        <!-- Fixed, Delimited, or Tab only fixed implemented for now -->
        <property name="messageFormat" value="Fixed"/>
        <!-- Enable to send custom comment character -->
        <property name="enableCommentChar" value="true"/>
        <!-- Specify comment character if message starts with #, to parse it else the message will not be parsed because the default comment char is # -->
        <property name="commentChar" value=" "/>
        <!-- Specify header length to parse the header part from the message-->
        <property name="headerLength" value="18"/>
        <!-- position to start parsing message data-->
        <property name="messageLengthStart" value="2"/>
        <!-- position to stop parsing message data-->
        <property name="messageLengthEnd" value="6"/>
        <property name="outputPart" value="appendedoutput3"/>
        <property name="instrumentationOn" value="true"/>
        <property name="jndinameInstr" value="java:/jdbc/postgresql/AutomationAgent1"/>
    </bean>
```

4 Open `processors.xml`. Each processor in the `processors.xml` file should have the `logFriendlyName` property set to something meaningful. This value will be inserted in `processname` column of the instrumentation table and displayed in Grafana. This approach makes it easy to spot where issues are if the process goes wrong. These are examples:

- Setting export properties
- Calling the REST processor
- Transforming data
- Sending message
- Updating export data

processors.xml

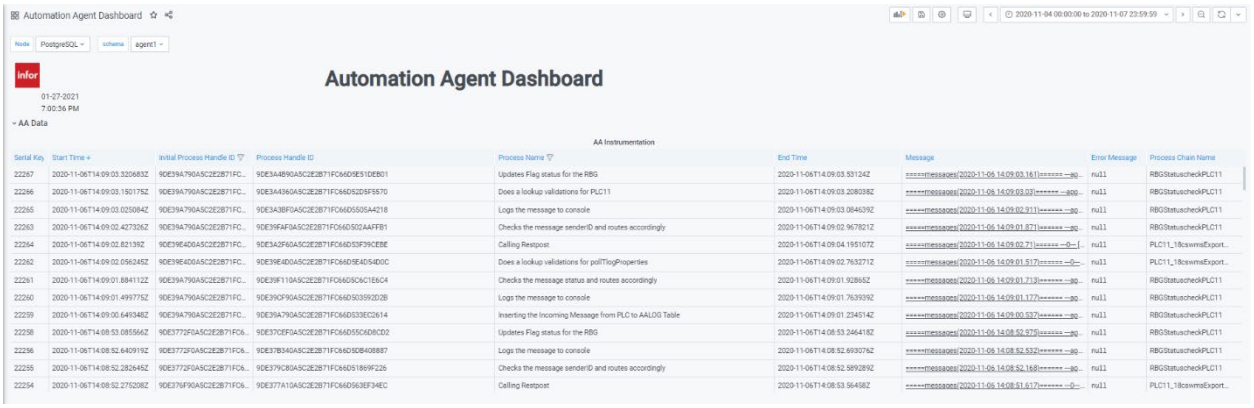
```
115
116
117     <bean id="restPost" class="com.infor.cswms.automationagent.processor.RestProcessor">
118         <property name="hostName" value="${restHostName}"/>
119         <property name="port" value="${port}"/>
120         <property name="contextRoot" value="/wmwebservice_rest/"/>
121         <property name="headers">
122             <map>
123                 <entry key="Username" value="${username}"/>
124                 <entry key="Password" value="${password}"/>
125                 <entry key="Tenant" value="${tenant}"/>
126                 <entry key="Content-Type" value="application/json"/>
127             </map>
128         </property>
129         <property name="outputPart" value="restpolleroutput"/>
130         <property name="endpointName" value="${endpoint}"/>
131         <property name="logFriendlyName" value="Calling Restpost" />
132     </bean>
```

Instrumentation table

The Instrumentation table has these columns which are displayed in the user interface:

Column name	Description
Serial Key	This is the primary key of a record in the table.
Initial Process Handle ID	A unique key that is assigned to the process initially. This property remains unique for the whole process. To follow a message from start to finish, look at all records with the same Initial Process Handle ID.
Process Handle ID	A unique key that is assigned per processor. This property remains unique for each processor in the process chain.
Process Name	The `logFriendlyName` of the processor that is configured by the user. This should be a short description.
Start Time	This records the start time of the process.
End Time	This records the end time of the process.
Message	This object contains all the information of the message stack including properties of the message and the content of the message body.
Error Message	This object contains the exceptions or error information if present.

Column name	Description
Process Chain Name	This property gives the information about process chain name of the processor.



Chapter 4 Configuring logging

This chapter describes what information is displayed in each of the supported logging types. These four forms of logging are available in Automation Agent:

- Instrumentation
- AlertProcessor
- FileLoggingProcessor
- Logback

Instrumentation

Instrumentation records the data for every step in a message process and can be viewed using Grafana.

See Instrumentation on page 13.

AlertProcessor

The `AlertProcessor` can be configured to write to the WMS' Extension Alerts screen via a REST API call, or to send an email to particular recipients based on severity, priority, and functional area. Add an `AlertProcessor` to your process chain at the desired point, and configure your `agent.xml` file with the desired alert mechanisms (`EventLogAlert` or `EmailAlert`) in the alerts list.

FileLoggingProcessor

Similar to the `AlertProcessor`, the `FileLogingProcessor` writes message data to the configured file. Place the processor in your process chain at any point you need a log, and set the file information in the configuration.

Note: This is useful during development but will have a performance impact after you are in production. Use other methods in production.

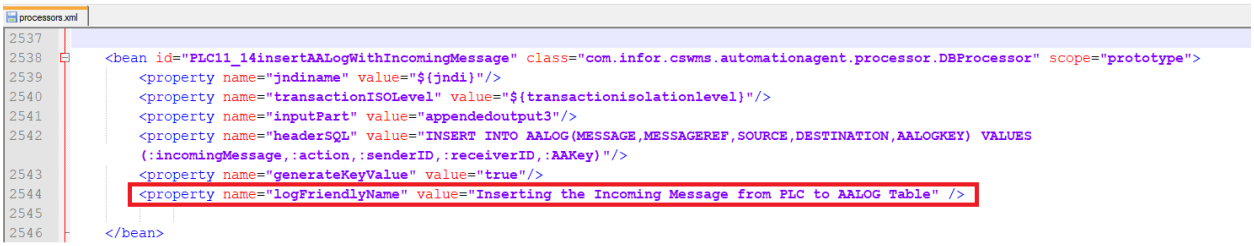
Logback

Logback is a standard logging mechanism used in Infor WMS products. A default configuration is set by the installer in your configuration directory. The default directory is `C:\infor\aa\AAPRD\agent1\conf`. We recommend running this at log level **`WARN`** in production.

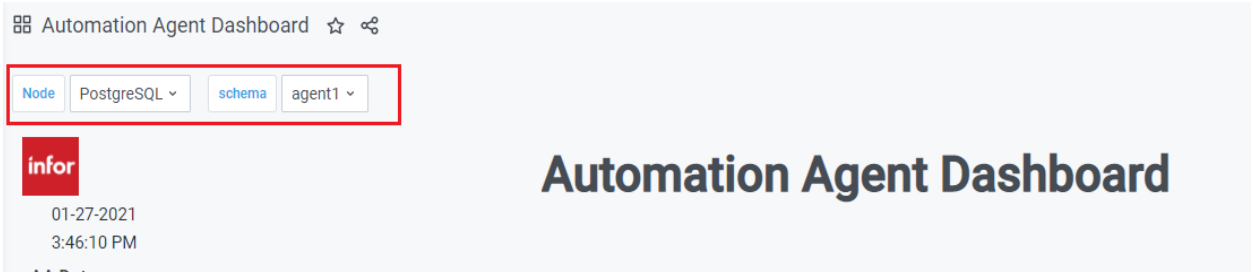
Chapter 5 Using the Grafana dashboard

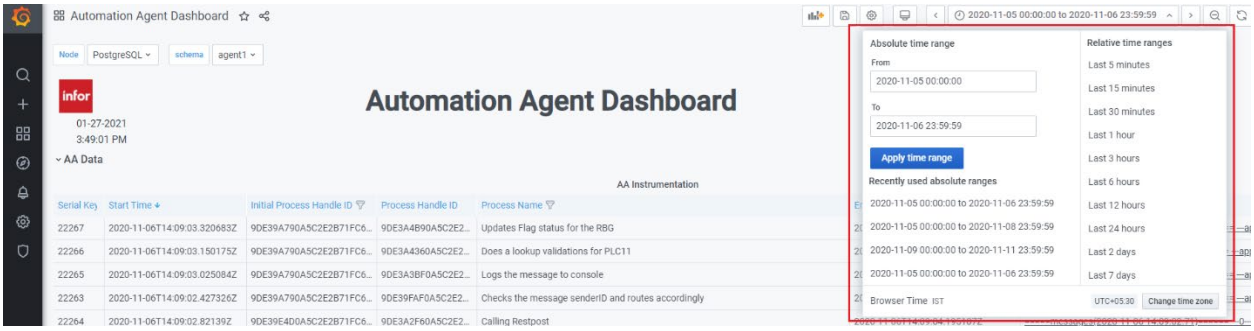
This section describes how to use the Grafana dashboard to check instrumentation logs. Use this dashboard for troubleshooting.

1 Navigate to `processors.xml` file. Locate the `logFriendlyName` property for one of the processors that belong to a process chain for which you want to see logs.

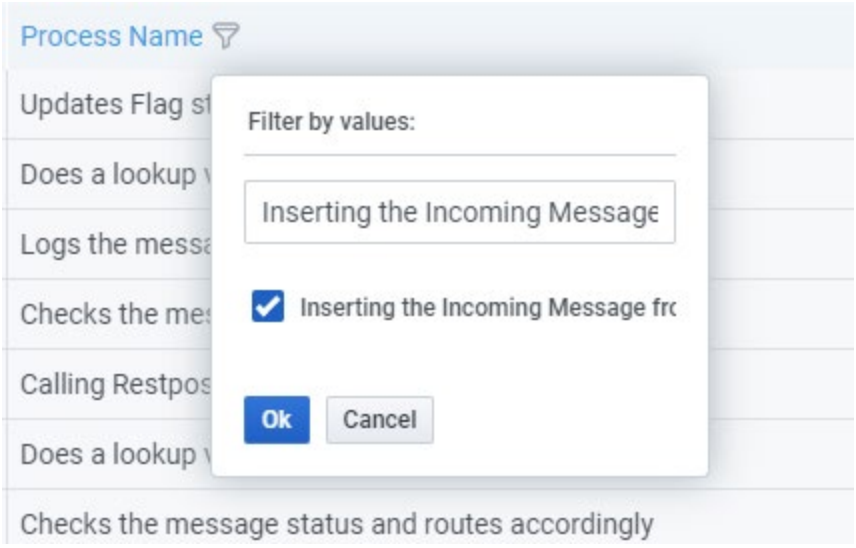


2 Navigate to the Automation Agent dashboard and select the appropriate node, schema, and time range.

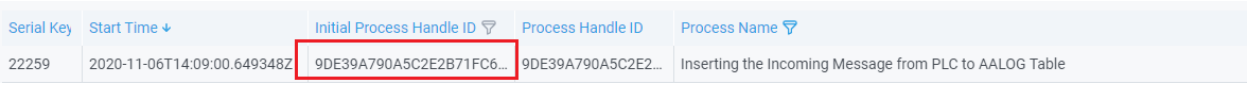




3 Click the filter icon on the Process Name column and search for `logfriendlyname` that was displayed in step 1. Select the value and click **OK**.

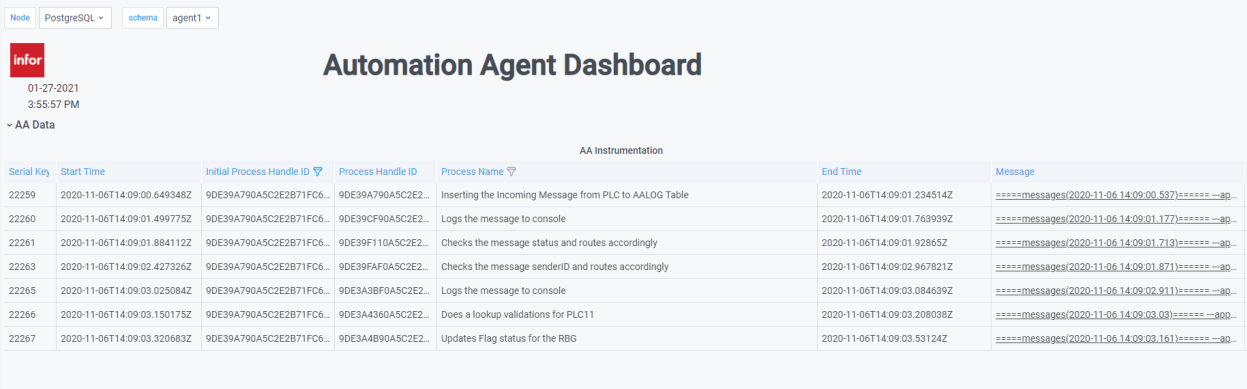


4 Locate the **Initial Process Handle ID** for the record and copy.

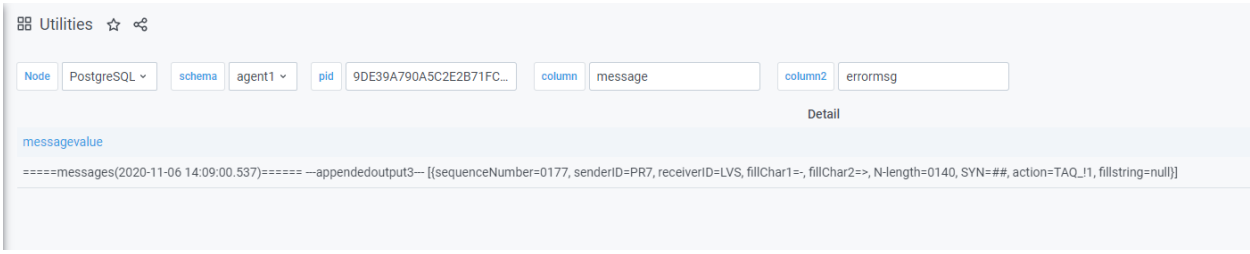


Serial Key	Start Time	Initial Process Handle ID	Process Handle ID	Process Name
22259	2020-11-06T14:09:00.649348Z	9DE39A790A5C2E2B71FC6...	9DE39A790A5C2E2...	Inserting the Incoming Message from PLC to AALOG Table

5 Apply the filter on **Initial Process Handle ID** column with the copied value and click **OK.** Clear the filter on **Process Name** column to locate all the process steps related to this `processhandleid`.



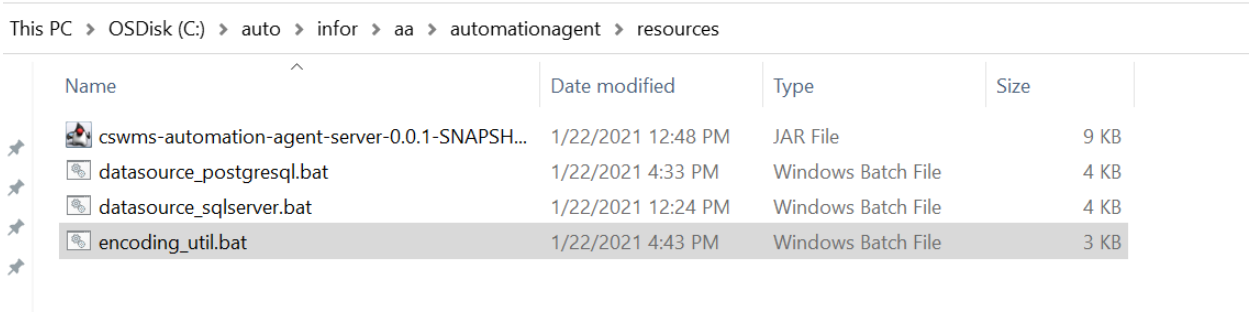
Serial Key	Start Time	Initial Process Handle ID	Process Handle ID	Process Name	End Time	Message
22259	2020-11-06T14:09:00.649348Z	9DE39A790A5C2E2B71FC6...	9DE39A790A5C2E2...	Inserting the Incoming Message from PLC to AALOG Table	2020-11-06T14:09:01.234514Z	=====messages(2020-11-06 14:09:00.537)====== ---ap...
22260	2020-11-06T14:09:01.499775Z	9DE39A790A5C2E2B71FC6...	9DE39CF90A5C2E2...	Logs the message to console	2020-11-06T14:09:01.763939Z	=====messages(2020-11-06 14:09:01.177)====== ---ap...
22261	2020-11-06T14:09:01.884112Z	9DE39A790A5C2E2B71FC6...	9DE39F110A5C2E2...	Checks the message status and routes accordingly	2020-11-06T14:09:01.92865Z	=====messages(2020-11-06 14:09:01.713)====== ---ap...
22263	2020-11-06T14:09:02.427326Z	9DE39A790A5C2E2B71FC6...	9DE39FAF0A5C2E2...	Checks the message senderID and routes accordingly	2020-11-06T14:09:02.967821Z	=====messages(2020-11-06 14:09:01.871)====== ---ap...
22265	2020-11-06T14:09:03.025084Z	9DE39A790A5C2E2B71FC6...	9DE3A3BF0A5C2E2...	Logs the message to console	2020-11-06T14:09:03.084639Z	=====messages(2020-11-06 14:09:02.911)====== ---ap...
22266	2020-11-06T14:09:03.150175Z	9DE39A790A5C2E2B71FC6...	9DE3A4360A5C2E2...	Does a lookup validations for PLC11	2020-11-06T14:09:03.208038Z	=====messages(2020-11-06 14:09:03.03)====== ---app...
22267	2020-11-06T14:09:03.320683Z	9DE39A790A5C2E2B71FC6...	9DE3A4B90A5C2E2...	Updates Flag status for the RBG	2020-11-06T14:09:03.53124Z	=====messages(2020-11-06 14:09:03.161)====== ---ap...



Appendix A Configuring password encryption utility

This section describes how to add encoded passwords in the configuration files for the flows running through Automation Agent.

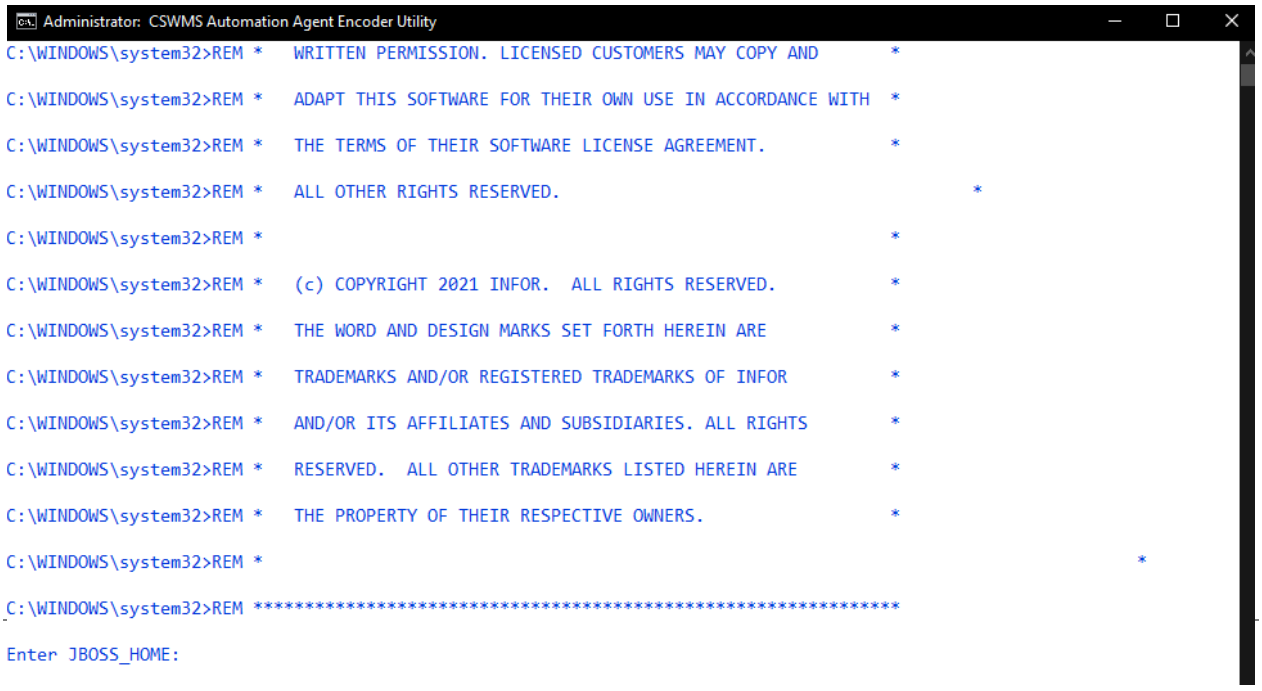
The Password Encryption Utility is located in `encoding_util.bat` in the default folder structure: `<AA_inst_loc>\aa\automationagent\resources.` For example, `C:\auto\infor\aa\automationagent\resources.`



To encode normal texts/strings to encoded text/strings:

1 Navigate to and select the `encoding_util.bat` file. Right click and select **Run as administrator**.

The Automation Agent Encoding utility is opened in a command prompt window.



2 When prompted for **JBOSS_HOME** value, specify the Automation Agent Wildlfy home path and press **Enter**.

3 When prompter specify **String**, specify the text/string to convert to encoded string and press **Enter**. For example, specify **`password`**.

```
C:\WINDOWS\system32>REM *   AND/OR ITS AFFILIATES AND SUBSIDIARIES. ALL RIGHTS        *

C:\WINDOWS\system32>REM *   RESERVED.  ALL OTHER TRADEMARKS LISTED HEREIN ARE         *

C:\WINDOWS\system32>REM *   THE PROPERTY OF THEIR RESPECTIVE OWNERS.                  *

C:\WINDOWS\system32>REM *                                                                                *

C:\WINDOWS\system32>REM ************************************************************

Enter JBOSS_HOME: C:\Wilfly20test\infor\wildfly-20.0.1.Final
```

The encoded string is displayed on the console.

```
C:\WINDOWS\system32>REM *   RESERVED.  ALL OTHER TRADEMARKS LISTED HEREIN ARE         *

C:\WINDOWS\system32>REM *   THE PROPERTY OF THEIR RESPECTIVE OWNERS.                  *

C:\WINDOWS\system32>REM *                                                                                *

C:\WINDOWS\system32>REM ************************************************************

Enter JBOSS_HOME: C:\Wilfly20test\infor\wildfly-20.0.1.Final

Enter String: password
```

4 Copy the encoded string from the console and paste it in the property value in AA configuration files.

After the encoded string is copied to the configuration files. Close the Encoding utility command window by pressing any key or clicking on close button on window. Plain text passwords can now be encoded passwords and the risk of exposing the passwords is eliminated.

Appendix B Configuring OAuth in Automation Agent

This section contains instructions for configuring OAuth within Automation Agent when making REST calls. All ids, keys, and secrets in the examples are for example only and do not represent values that can be used in any system.

OAuth 1.0a

OAuth 1.0.a uses a single key and secret combination to build a signature that is sent along with the request. The receiving application verifies this signature and accepts or rejects the request depending on its validity. Direct connections to Infor WMS use this method to connect.

To obtain the keys, go to Infor WMS's master db instance and select from the enterprise schema configuration table where the column **parm** is **SCE_REST_CI** for your client id and **SCE_REST_CS** for your client secret.

In your `processors.xml` file add an **OAuthClientConfig** bean that looks like this example, but with your CI and CS keys.

```
<bean id="oauth1_mylocalsce"
class="com.infor.cswms.automationagent.util.OAuthClientConfig">

<property name="oauthVersion" value="1.0"/>

 <property name="clientId" value="tenant~rTGNAt_w62n4927kLu9c8I-
qpoIdKHraD9k_-02RQ0Z"/>

 <property name="clientSecret"
value="V78aQjaSd9yvQKpFikwnxBQJcoXeKUZg7V7EURKU3lHu6lC"/>

</bean>
```

Placement is not critical, but we recommend that you place it immediately after your **RestProcessor** configuration bean.

In your **RestProcessor** configuration bean, add the following property:

```
<property name="oauth">

<ref bean="oauth1_mylocalsce" />

</property>
```

The name of the **OAuthClientConfig** bean can be anything that makes sense, such as oauth1_servername, but it must match the name specified in this property.

OAuth 1.0a is now configured.

OAuth 2.0

OAuth 2.0 is required when communicating through ION API. It also uses a key and secret, but adds an authorization endpoint and a username and password for that endpoint. When a request is to be made, the caller must first call that authorization endpoint with the user and password to get a temporary authorization token, which is then used to generate the signature on the request to the system. This is be handled automatically.

To get the necessary entries, in ION API go to Authorized Apps. If you do not already have an appropriate client, create one with the connection information to your Infor WMS application. If you do have one, open it. Towards the bottom of the page, click **Download Credentials**. A confirmation screen is displayed. You must associate these credentials with a user. Enable the slider to associate a user and enter the user in the user field. Download the credentials.

You should now have a file with the name of your Authorized Client and an `ionapi` extension with a JSON sting as the content. The content will be on one line but if formatted will look like this example:


```
{
"ti": "tenantid",
"cn": "PostmanTesting",
"dt": "12",
"ci": "SERVER_NAME~rJPEGt_w62n5762kLu9c8I-ktdIdKHraD9k_-02DE0Z",
"cs":
"V78aQjaSd9yvQKpFikwnxBQJcoXeKUZg7V7EURKU3lHu6lC4Y3X6XQr8qcKn5hcGpais13lbcNkC
NYS7NH2SdpA",
"iu": "https://qac-ionapi.qac.awsdev.infor.com",
"pu": "https://qac-sso.qac.awsdev.infor.com:443/TENANT/as/",
"oa": "authorization.oauth2",
"ot": "token.oauth2",
"or": "revoke_token.oauth2",
"ev": "M1528929412",
"v": "1.0",
"saak":
"SERVER_NAME#FjsaneA_KH5L8U60c2W55e1I3pP4jP565C3ZxUzO0cxj4iy3aPmdzyo5gZR9i4qu
LvaH1pIai",
```

```
"sask": "Q7YDiamndM1zmYoRjc3u0Mtxl8urIa-
aghwy1bzXp1VDz6maZNssOxH9eLgp6LMfgw_u0uaAkS95TF_evJjkauw"

}
```

If you do not have the saak and sask entries, you did not associate the credentials with a user and the file is not usable. Try the download again and ensure that the slider is enabled and a user is specified.

In your processors.xml file add an `OAuthClientConfig` bean that looks like this example, but with the information from your downloaded `ionapi` file.

```
<bean id="oauth2_SERVER"
class="com.infor.cswms.automationagent.util.OAuthClientConfig">

<property name="oauthVersion" value="2.0" />

<property name="authorizationEndpoint" value="https://qac-
sso.qac.awsdev.infor.com:443/TENANT/as/token.oauth2" />

<property name="accountName"
value="SERVER_NAME#FjsanU60c2W55e1I3pP4jP565C3aH1pIainZ1ELSqPw"/>

<property name="accountPassword" value="Q7YDiamndM1zmYoRjc3u0Mtxl8urIa-
aghwy1bzXp1VDz6maZNssO"/>

<property name="clientId" value="SERVER_NAME~rTGNAt_w62n4927kLu9c8I-
qpoIdKHraD9k_-02RQ0Z" />

<property name="clientSecret"
value="V78aQjaSd9yvQKpFikwnxBQJcoXeKUZg7V7EURKU3lHu6lC4Y3X6XQr8"/>

</bean>
```

See these explanations:

- authorizationEndpoint is built by concatenating the `pu` and `ot` values from your downloaded `ionapi` file.
- accountName is the `saak` value.
- accountPassword is the `sask` value.
- clientId is the `ci` value.
- clientSecret is the `cs` value.

Placement of this bean is not critical, but we recommend that you place it immediately after your `RestProcessor` configuration bean.

In your `RestProcessor` configuration bean, add the following property:

```
<property name="oauth">

<ref bean="oauth2_SERVER" />

</property>
```

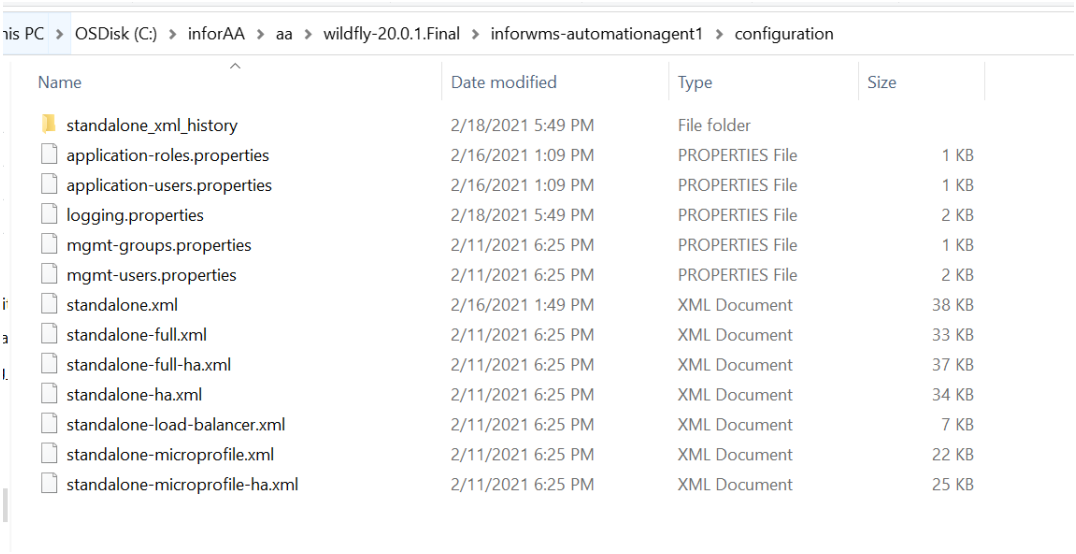
The name of the `OAuthClientConfig` bean can be anything that makes sense, such as oauth2_servername, but it must match the name specified in this property. When specifying the host in your `RestProcessor` bean, ensure that you use the host from the `iu` property and not `pu`. The `pu` property is for generating the authorization token, and while it will be similar to `iu`, the values are different and not interchangeable.

OAuth 2.0 is now configured.

Appendix C Configuring JMS for Automation Agent

This section describes how to set up Java Message Service (JMS) within Automation Agent.

1 Navigate to the Automation Agent deployment folder and locate the configuration folder. The default path is `<AA_INSTALLATION_PATH>\wildfly-<Version>\inforwms-automationagent1\configuration`. For example, `C:\inforAA\aa\wildfly-20.0.1.Final\inforwms-automationagent1\configuration`

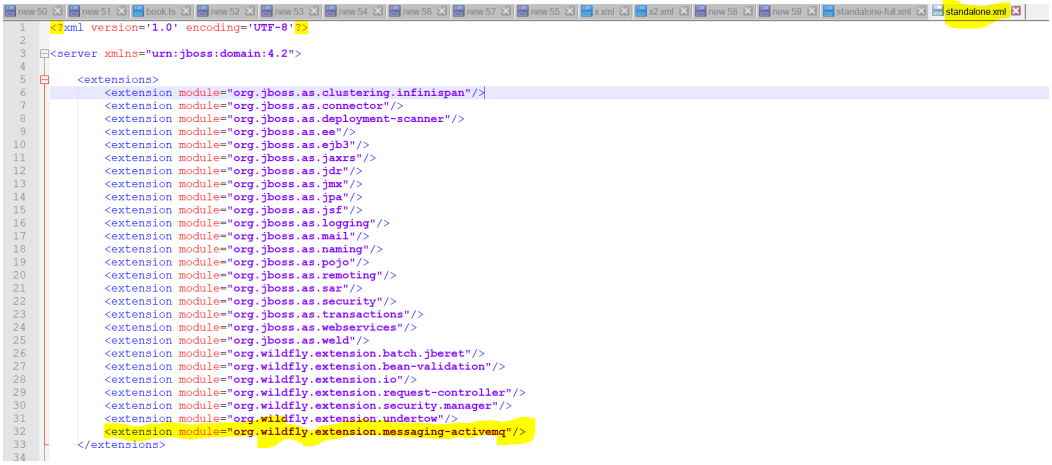


2 Open `standalone.xml`, and `standalone-full.xml` in any editor such as Notepad++.

3 Copy these items from `standalone-full.xml` and paste it to `standalone.xml`.

   a Copy `messaging-activemq` extension module and paste it under **extensions** in `standalone.xml`.

```
<extension module="org.wildfly.extension.messaging-activemq"/>
```



b Copy `messaging-activemq` subsystem and paste it under **profile** in `standalone.xml`

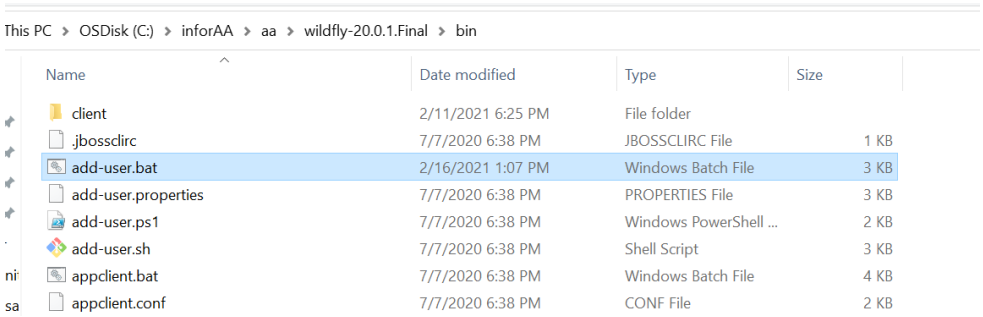
```
<subsystem xmlns="urn:jboss:domain:messaging-activemq:10.0">
<server name="default">
<statistics enabled="${wildfly.messaging-activemq.statistics-enabled:${wildfly.statistics-enabled:false}}"/>
<security-setting name="#">
<role name="guest" send="true" consume="true" create-non-durable-queue="true" delete-non-durable-queue="true"/>
</security-setting>
<address-setting name="#" dead-letter-address="jms.queue.DLQ" expiry-address="jms.queue.ExpiryQueue" max-size-bytes="10485760" page-size-bytes="2097152" message-counter-history-day-limit="10"/>
<http-connector name="http-connector" socket-binding="http" endpoint="http-acceptor"/>
<http-connector name="http-connector-throughput" socket-binding="http" endpoint="http-acceptor-throughput">
<param name="batch-delay" value="50"/>
</http-connector>
<in-vm-connector name="in-vm" server-id="0">
<param name="buffer-pooling" value="false"/>
</in-vm-connector>
<http-acceptor name="http-acceptor" http-listener="default"/>
<http-acceptor name="http-acceptor-throughput" http-listener="default">
<param name="batch-delay" value="50"/>
<param name="direct-deliver" value="false"/>
</http-acceptor>
<in-vm-acceptor name="in-vm" server-id="0">
<param name="buffer-pooling" value="false"/>
</in-vm-acceptor>
<jms-queue name="ExpiryQueue" entries="java:/jms/queue/ExpiryQueue"/>
<jms-queue name="DLQ" entries="java:/jms/queue/DLQ"/>
<connection-factory name="InVmConnectionFactory" entries="java:/ConnectionFactory" connectors="in-vm"/>
<connection-factory name="RemoteConnectionFactory" entries="java:jboss/exported/jms/RemoteConnectionFactory" connectors="http-connector"/>
<pooled-connection-factory name="activemq-ra" entries="java:/JmsXA java:jboss/DefaultJMSConnectionFactory" connectors="in-vm" transaction="xa"/>
</server>
</subsystem>
```

```
<subsystem xmlns="urn:jboss:domain:messaging-activemq:10.0">
    <server name="default">
        <statistics enabled=
        "${wildfly.messaging-activemq.statistics-enabled:${wildfly.statistics-enabled:false}}"/
        <security-setting name="#">
            <role name="guest" send="true" consume="true" create-non-durable-queue="true"
            delete-non-durable-queue="true"/>
        </security-setting>
        <address-setting name="#" dead-letter-address="jms.queue.DLQ" expiry-address=
        "jms.queue.ExpiryQueue" max-size-bytes="10485760" page-size-bytes="2097152"
        message-counter-history-day-limit="10"/>
        <http-connector name="http-connector" socket-binding="http" endpoint="http-acceptor"/>
        <http-connector name="http-connector-throughput" socket-binding="http" endpoint=
        "http-acceptor-throughput">
            <param name="batch-delay" value="50"/>
        </http-connector>
        <in-vm-connector name="in-vm" server-id="0">
            <param name="buffer-pooling" value="false"/>
        </in-vm-connector>
        <http-acceptor name="http-acceptor" http-listener="default"/>
        <http-acceptor name="http-acceptor-throughput" http-listener="default">
            <param name="batch-delay" value="50"/>
            <param name="direct-deliver" value="false"/>
        </http-acceptor>
        <in-vm-acceptor name="in-vm" server-id="0">
            <param name="buffer-pooling" value="false"/>
        </in-vm-acceptor>
        <jms-queue name="ExpiryQueue" entries="java:/jms/queue/ExpiryQueue"/>
        <jms-queue name="DLQ" entries="java:/jms/queue/DLQ"/>
        <jms-queue name="MessageQueue2" entries="java:jboss/exported/jms/MessageQueue2" durable=
        "true"/>
        <jms-queue name="MessageQueue3" entries="java:jboss/exported/jms/MessageQueue3" durable=
        "true"/>
        <jms-queue name="MessageQueue1" entries="java:jboss/exported/jms/MessageQueue1" durable=
        "true"/>
        <connection-factory name="InVmConnectionFactory" entries="java:/ConnectionFactory" connectors
        ="in-vm"/>
        <connection-factory name="RemoteConnectionFactory" entries=
        "java:jboss/exported/jms/RemoteConnectionFactory" connectors="http-connector"/>
        <pooled-connection-factory name="activemq-ra" entries="java:/JmsXA
        java:jboss/DefaultJMSConnectionFactory" connectors="in-vm" transaction="xa"/>
    </server>
</subsystem>
```

4 Create a guest user and password required to establish a connection to send/receive messages. These credentials will be required in coding part to send and receive messages.

a Navigate to `wildfly bin folder<AA_INSTALLATION_PATH>\wildfly-<Version>\bin`

For example, `C:\inforAA\aa\wildfly-20.0.1.Final\bin`



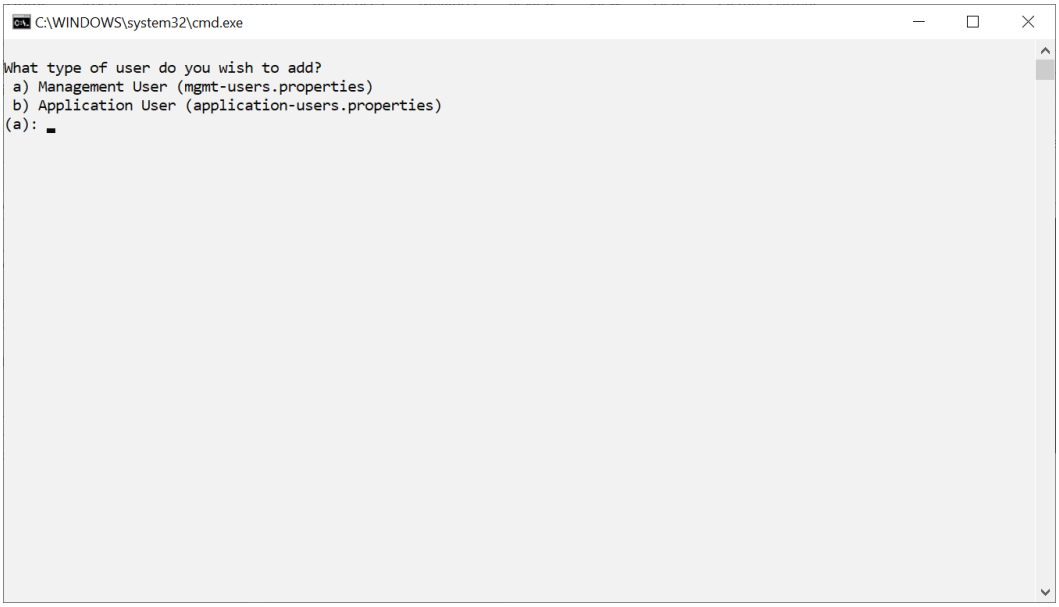
b Open `add-user.bat` in any editor such as Notepad++. Modify the `jboss.server.config.user.dir` directory entry to point to your automation agent deployment folder and save.

```
rem Uncomment to override standalone and domain user location
rem set "JAVA_OPTS=%JAVA_OPTS% -Djboss.server.config.user.dir=..\standalone\configuration -Djboss.domain.config.user.dir=..\domain\configuration"
```

```
43   echo JAVA_HOME is not set. Unexpected results may occur.
44   echo Set JAVA_HOME to the directory of your local JDK to avoid this message.
45 ) else (
46   set "JAVA=%JAVA_HOME%\bin\java"
47 )
48
49 rem Find jboss-modules.jar, or we can't continue
50 if exist "%JBOSS_HOME%\jboss-modules.jar" (
51     set "RUNJAR=%JBOSS_HOME%\jboss-modules.jar"
52 ) else (
53   echo Could not locate "%JBOSS_HOME%\jboss-modules.jar".
54   echo Please check that you are in the bin directory when running this script.
55   goto END
56 )
57
58 rem Set default module root paths
59 if "x%JBOSS_MODULEPATH%" == "x" (
60   set "JBOSS_MODULEPATH=%JBOSS_HOME%\modules"
61 )
62
63 rem Uncomment to override standalone and domain user location
64 set "JAVA_OPTS=%JAVA_OPTS% -Djboss.server.config.user.dir=
   C:\AutomationAgent\infor\wildfly-10.1.0.Final\cswms-automationagent1\configuration -Djboss.domain.config.user.dir=..\domain\configuration"
65
```

c Open `cmd` and run as administrator.

d This prompt will display:**`What type of user do you wish to add`**?. Select **`b`** to add an Application User.

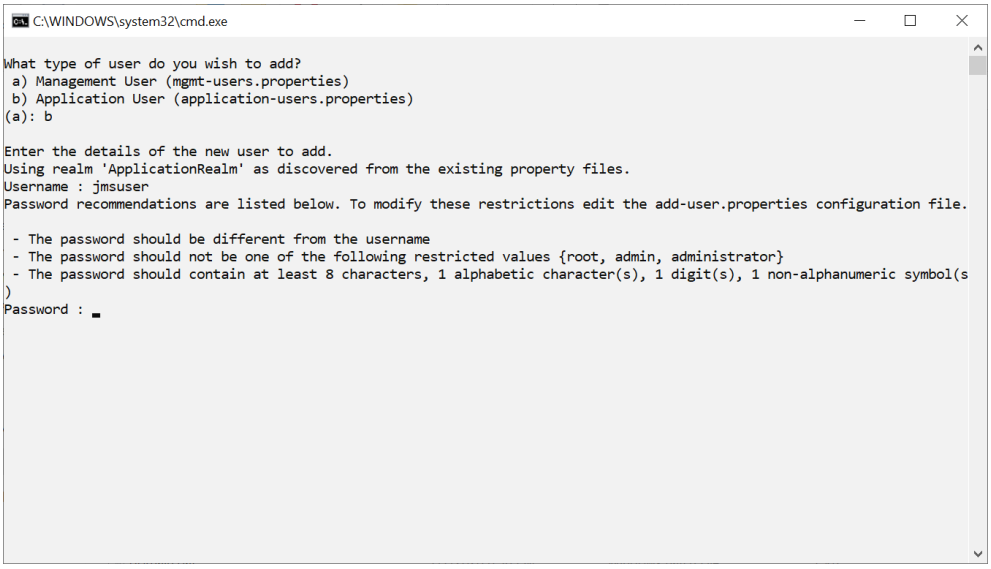


e Specify the **Username** and **Password** when prompted. For example:

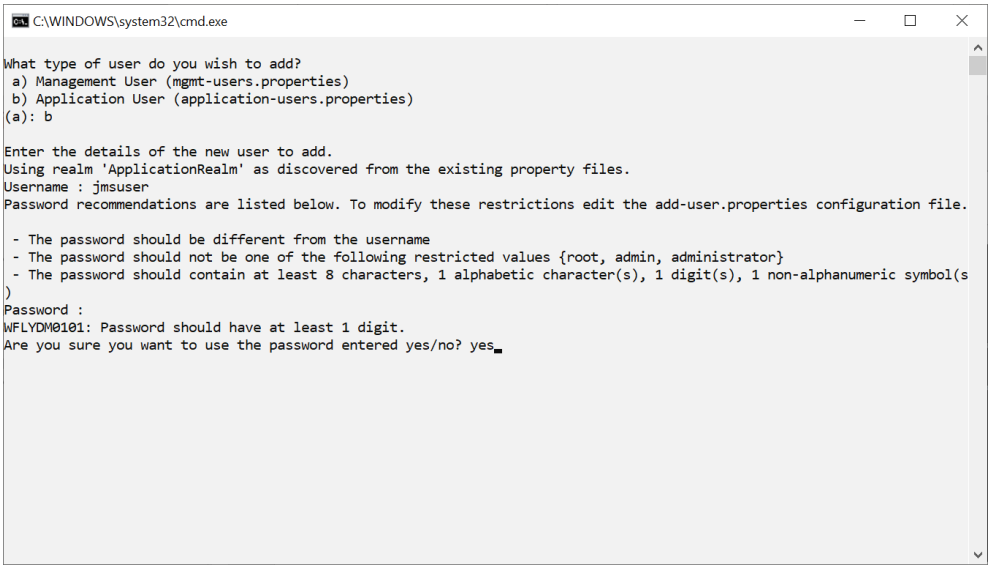
Username: **`jmsuser`**

Password: **`jmspassword`**

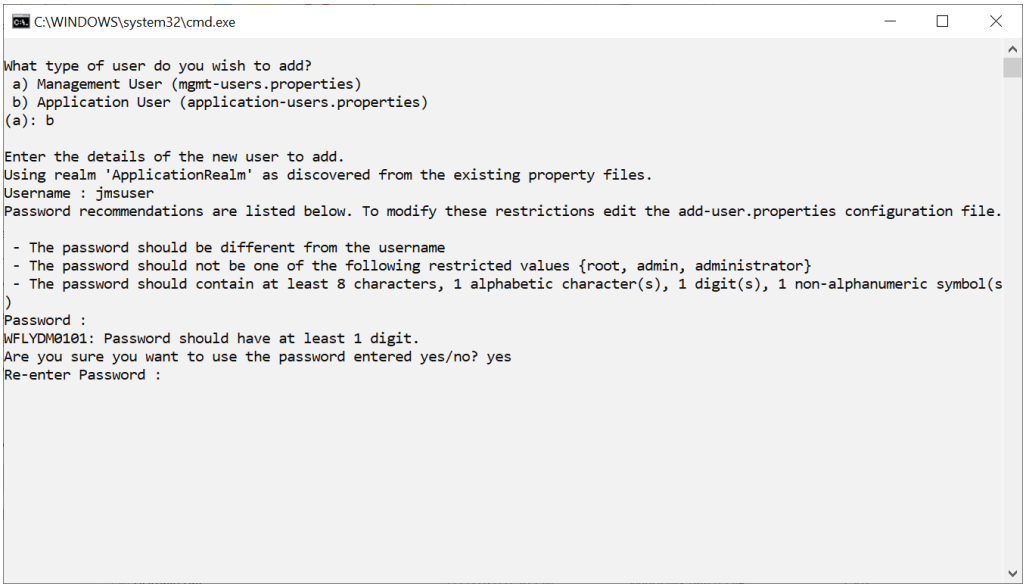
Note: The password is not visible to user while typing.



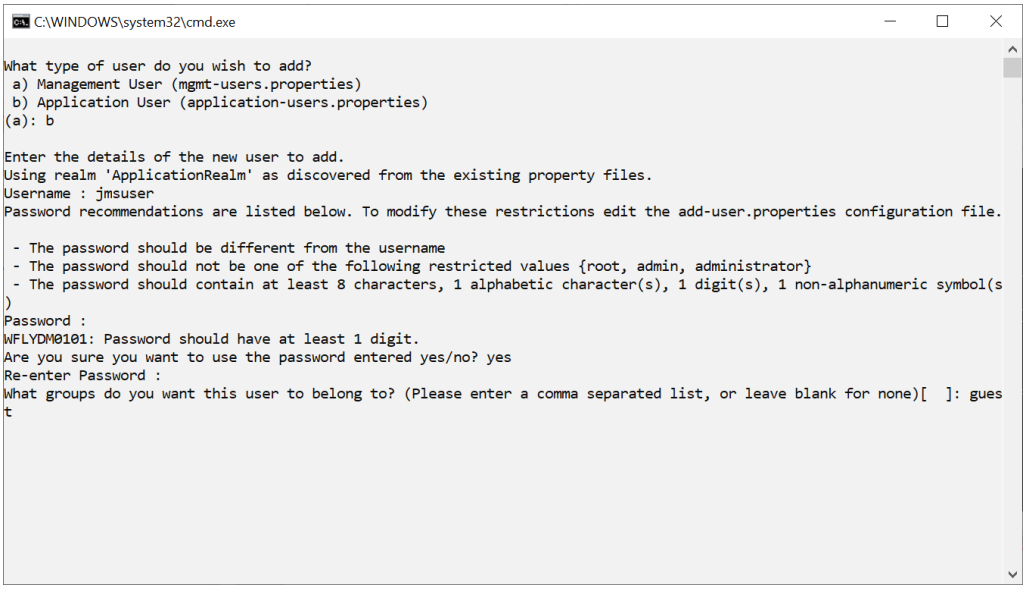
f This prompt will display: **Are you sure you want to use the password entered yes/no?** Specify **yes**.



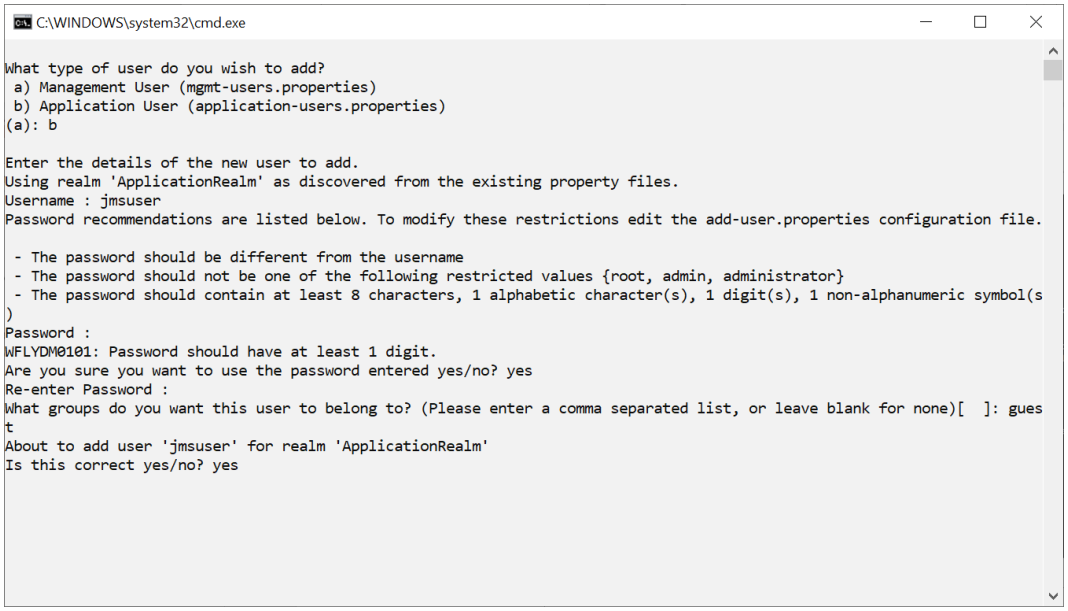
g Specify the password when prompted to re-enter password.

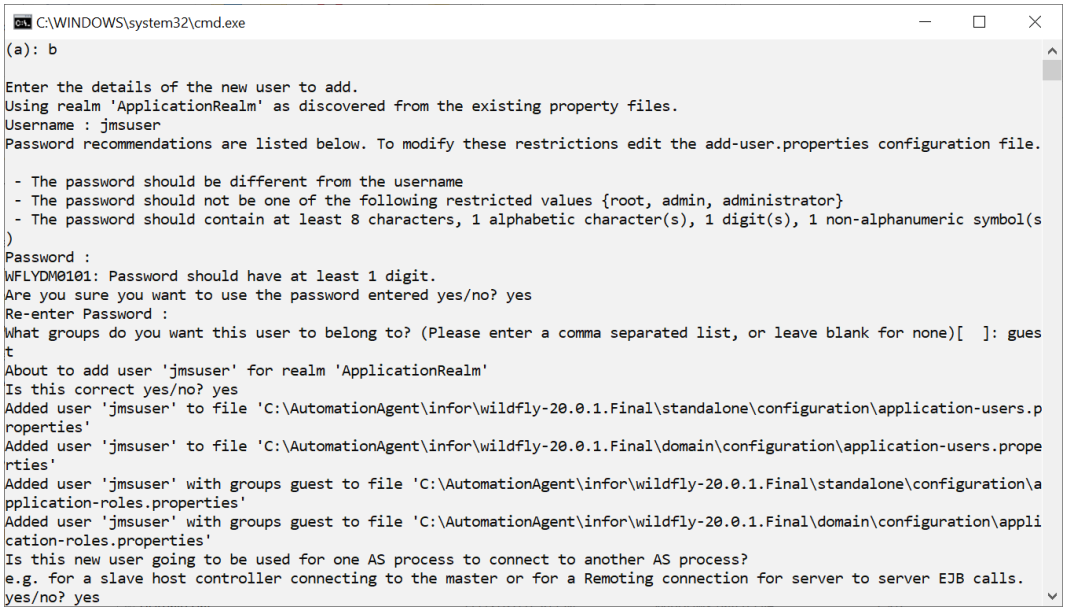


h This prompt will display: **`What groups do you want this user to belong to`**? Specify **`guest`** as role and click **Enter**.

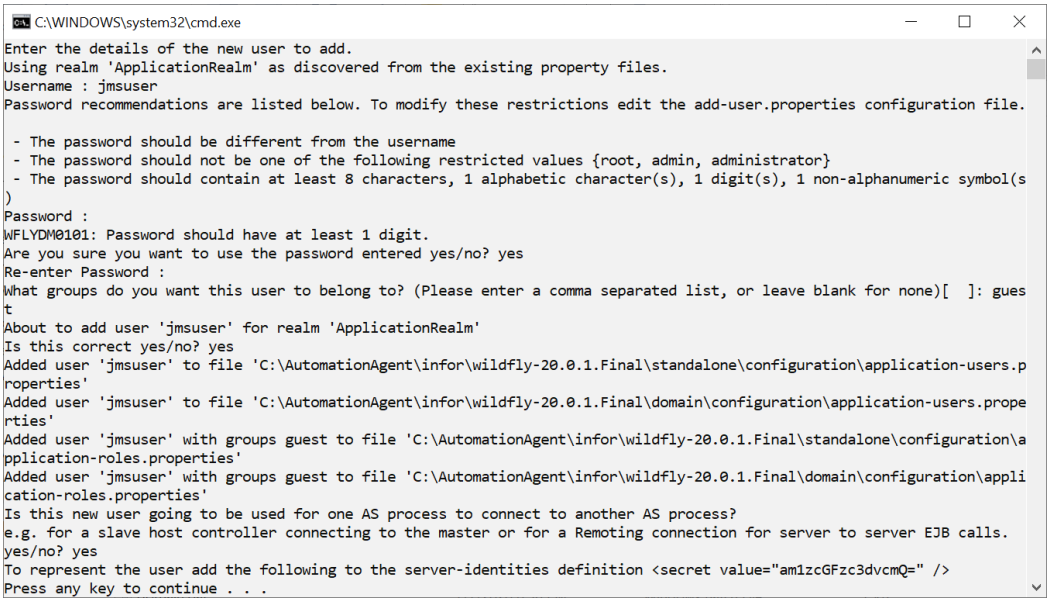


i This prompt will display: **`About to add user 'jmsuser' for realm 'ApplicationRealm' Is this correct yes/no?`** Specify **`yes`** twice and press **Enter**.

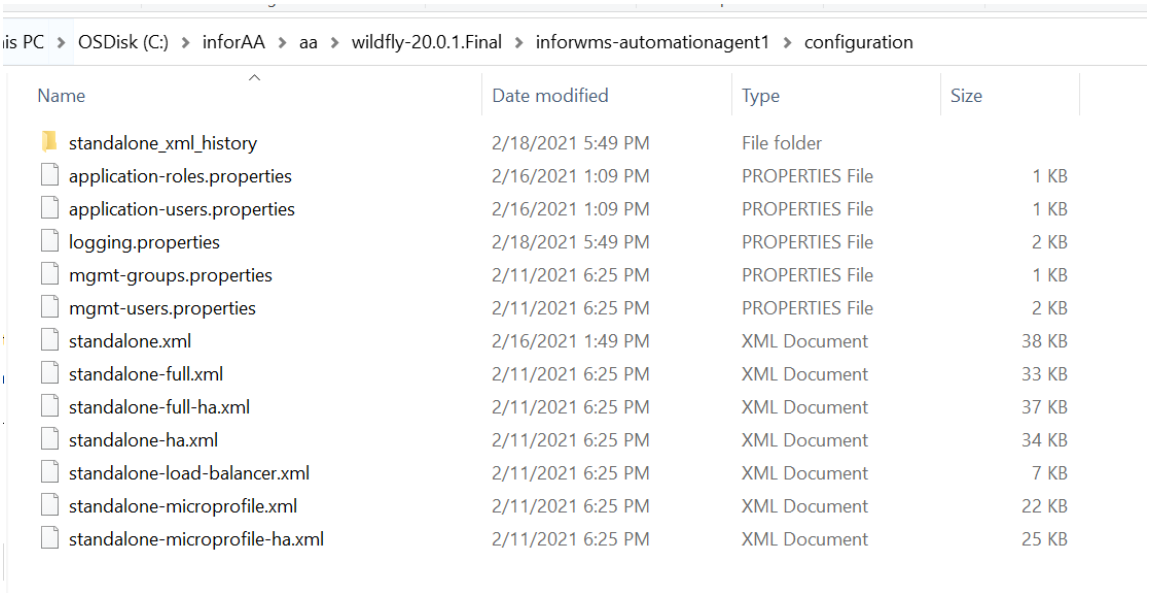


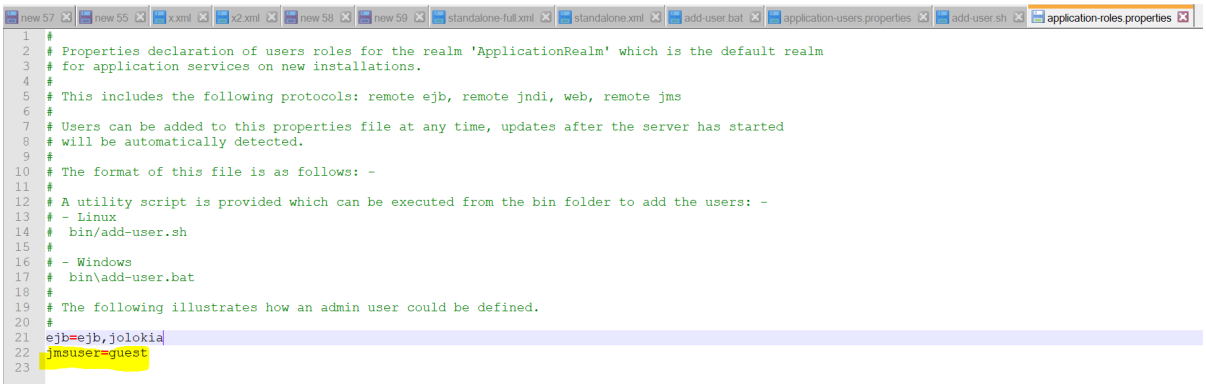


j Press any key to continue and exit from the batch file.



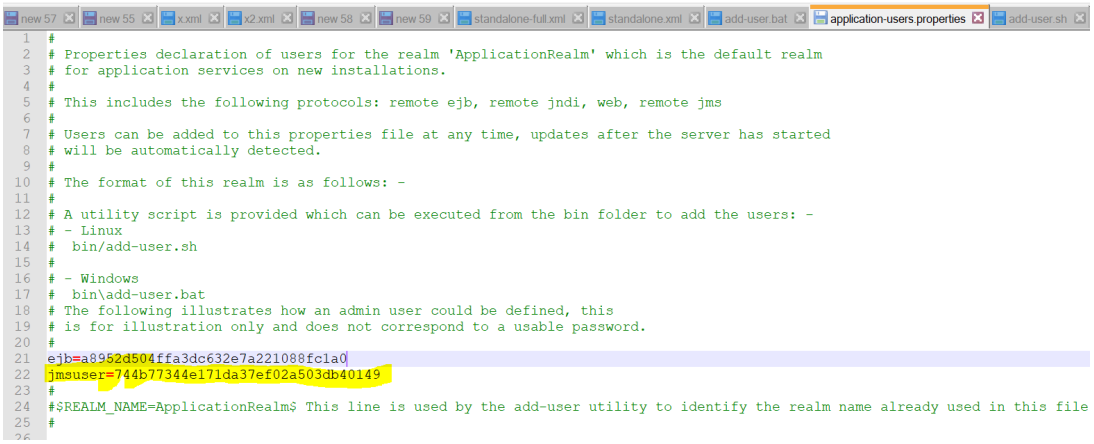
5 Open `application-roles.properties` and `application-users.properties` files under Automation Agent configuration folder and confirm the user with guest role has been added.





```
#
# Properties declaration of users roles for the realm 'ApplicationRealm' which is the default realm
# for application services on new installations.
#
# This includes the following protocols: remote ejb, remote jndi, web, remote jms
#
# Users can be added to this properties file at any time, updates after the server has started
# will be automatically detected.
#
# The format of this file is as follows: -
#
# A utility script is provided which can be executed from the bin folder to add the users: -
# - Linux
#  bin/add-user.sh
#
# - Windows
#  bin\add-user.bat
#
# The following illustrates how an admin user could be defined.
#
ejb=ejb,jolokia
jmsuser=guest

```



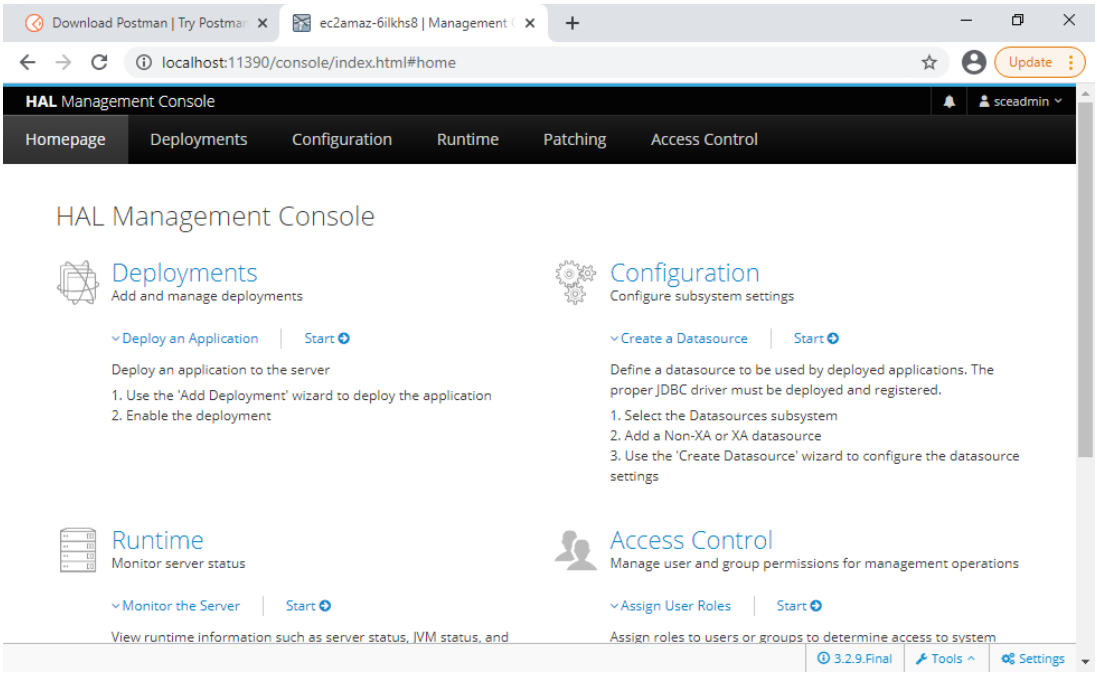
```
#
# Properties declaration of users for the realm 'ApplicationRealm' which is the default realm
# for application services on new installations.
#
# This includes the following protocols: remote ejb, remote jndi, web, remote jms
#
# Users can be added to this properties file at any time, updates after the server has started
# will be automatically detected.
#
# The format of this realm is as follows: -
#
# A utility script is provided which can be executed from the bin folder to add the users: -
# - Linux
#  bin/add-user.sh
#
# - Windows
#  bin\add-user.bat
# The following illustrates how an admin user could be defined, this
# is for illustration only and does not correspond to a usable password.
#
ejb=a8952d504ffa3dc632e7a221088fc1a0
jmsuser=744b77344e171da37ef02a503db40149
#
#$REALM_NAME=ApplicationRealm$ This line is used by the add-user utility to identify the realm name already used in this file
#

```

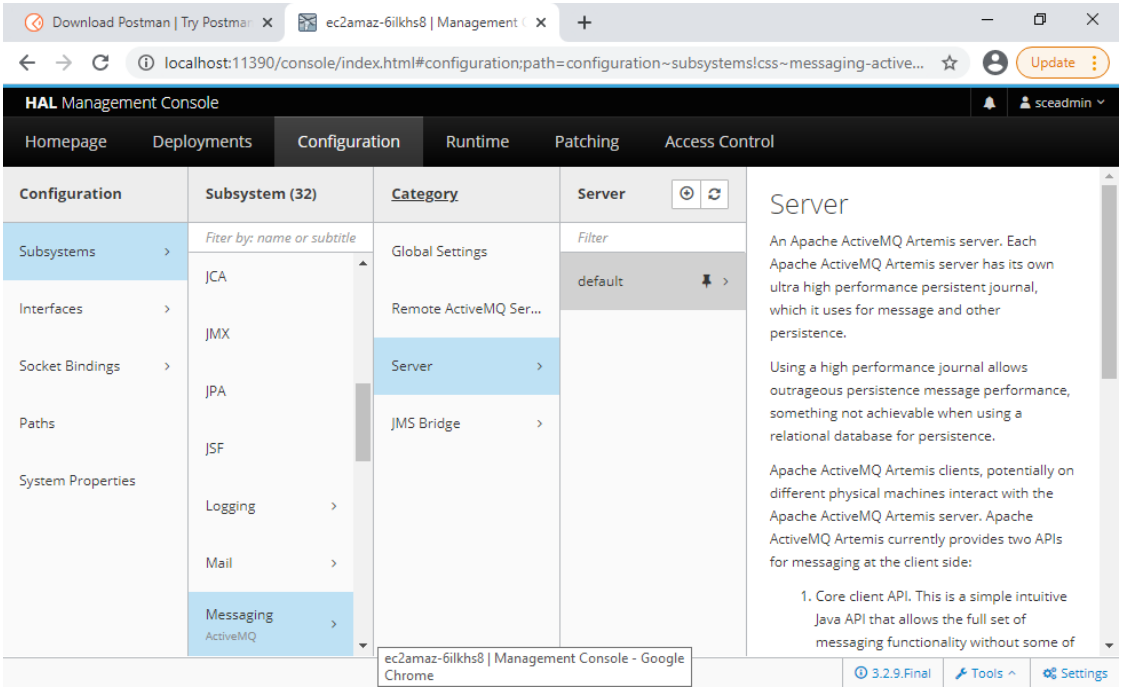
6 Start the Automation Agent bat file and specify this URL in a browser: **`http://localhost:9990/`**

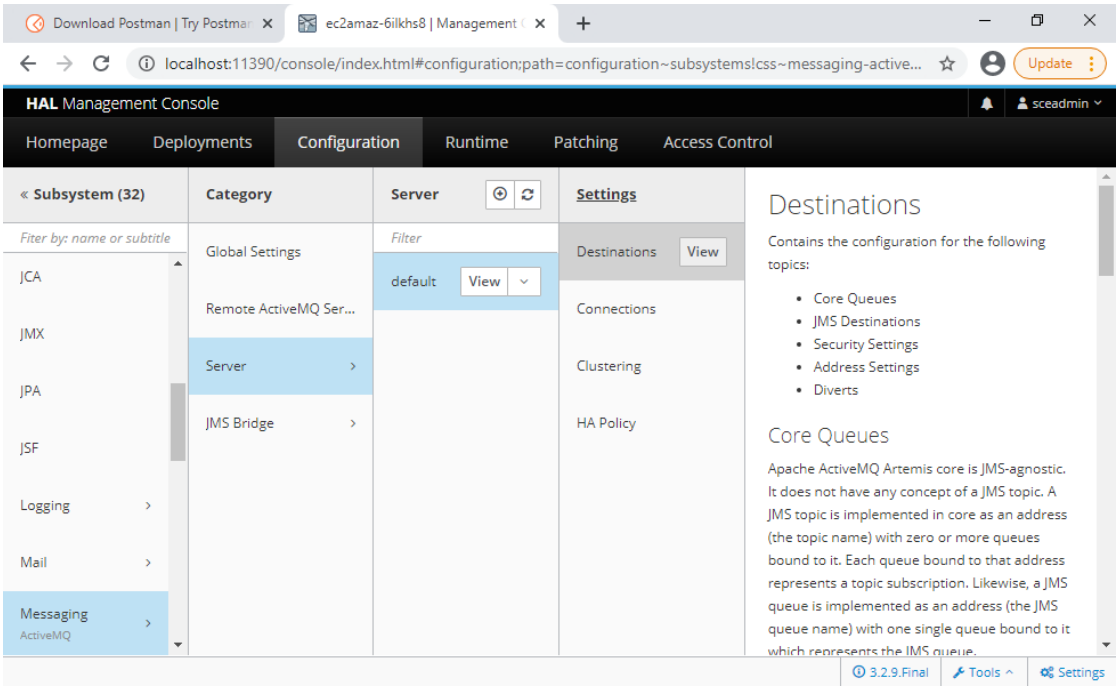
7 When prompted for **username** and **password,** specify **`sceadmin`** and the password configured for **`sceadmin`**. You can locate this password in the Automation Agent installation folder `C:\infor\aa\AAPRD in the AA_install.html file`.

The Automation Agent Admin page is displayed.

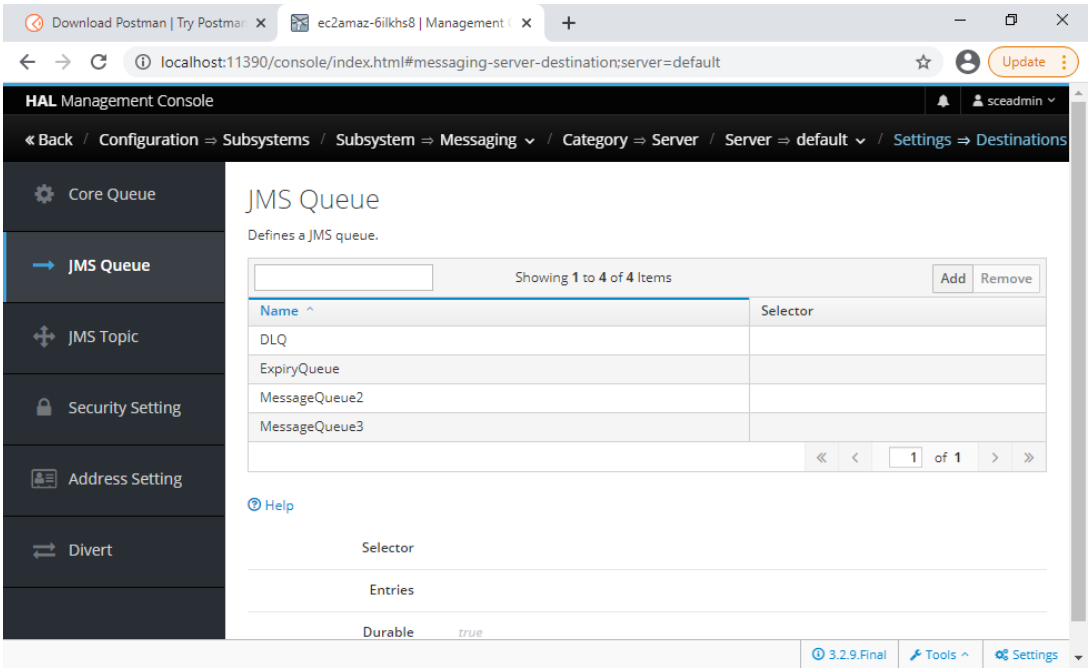


8 Navigate to **Configuration > Subsystems > Messaging-ActiveMQ > Server > default > Destinations**. Select **View**.





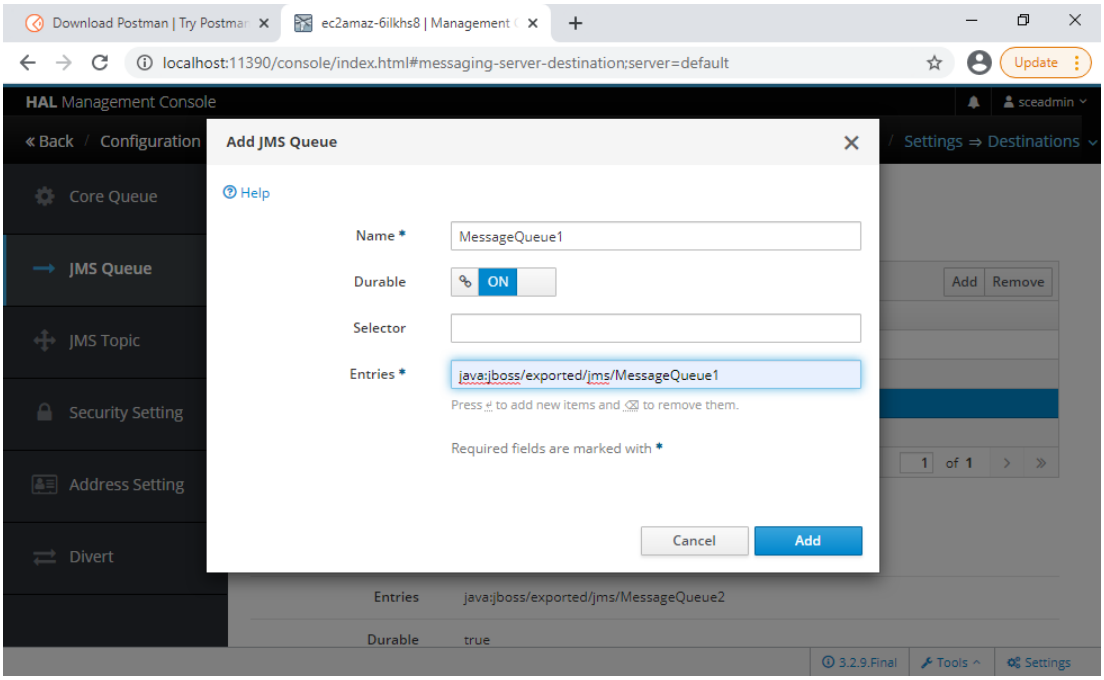
9 When the **Destinations** window is displayed, select **JMS Queue > JMS Topic** and click **Add** to add new queue/topic.



10 Specify **Name** and **Entries** and click **Save**.

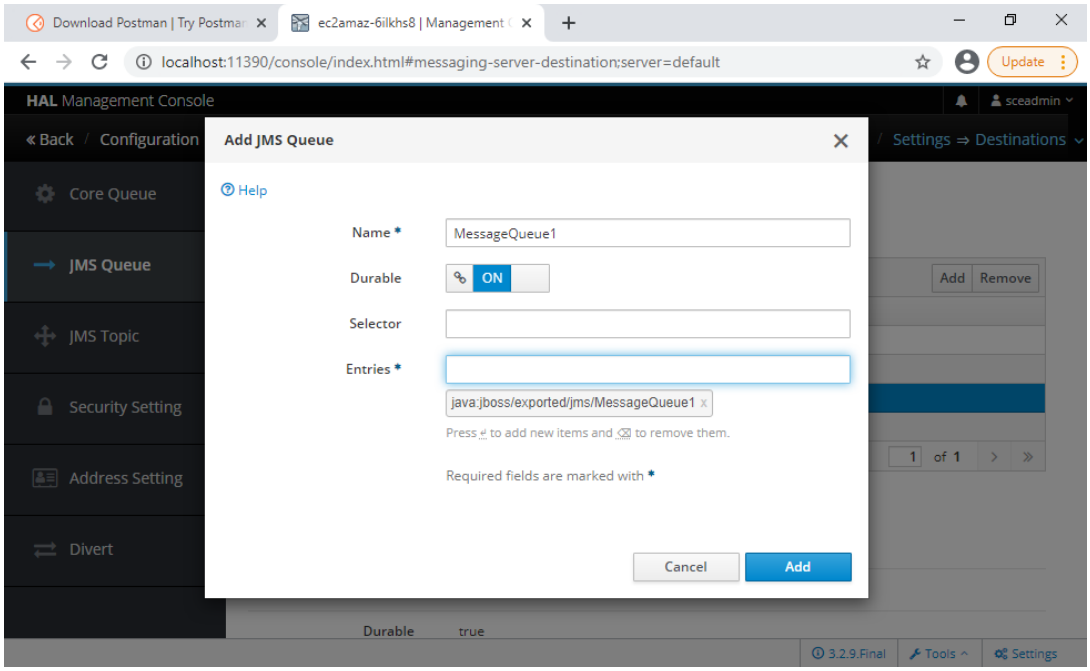
For example:

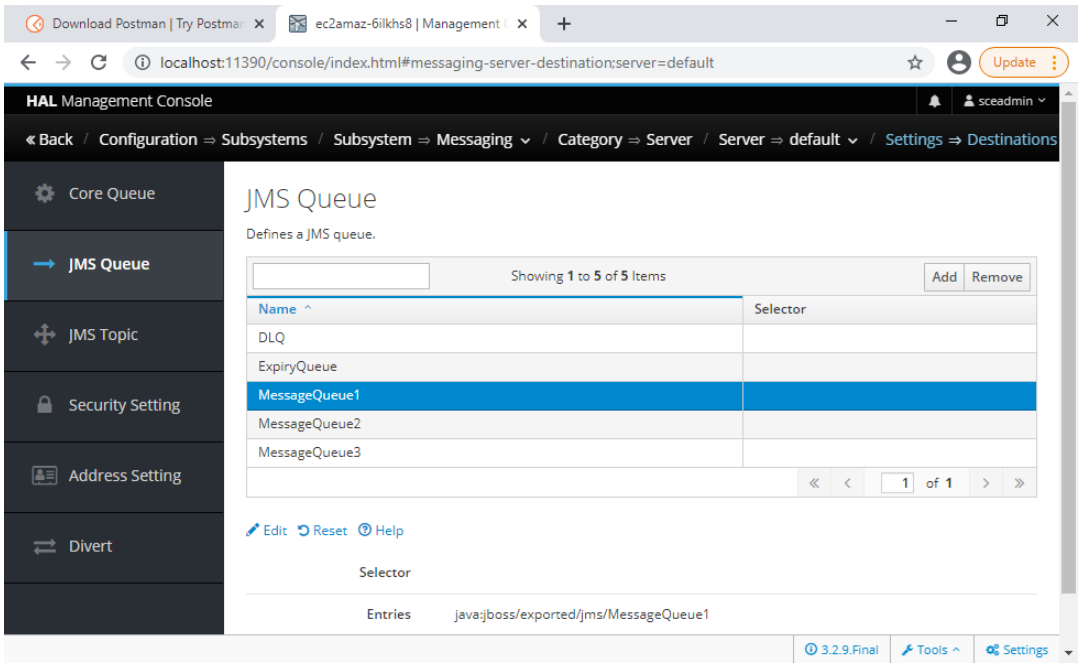
- **Name**: `MessageQueue1`
- **Entries**: `java:jboss/exported/jms/MessageQueue1`



11 Press **Enter** to add the entries and then click **Add**.

The new queue is added.





The JMS queue has been configured in Automation Agent.