CA Lisa Release Automation 4.7.1

Architecture Guide and Implementation

Best Practice Guide

Author : Keith Puzey - CA SWAT

Version: 3.3

Filename: CA Release Automation Architecture and implementation Guide 4.7.docx

Date: Monday, 01 September 2014

Contents

Document History

Version	Date	Description
2.9	June 29th 2014	Initial Document Released
3.0	July 7th 2014	Tuning Tips section Added Monitoring Section Added
3.1	July 31st 2014	Included update for new Hot Fix
3.3	September 1st 2014	Updated STAR Execution Section

CA Lisa Release Automation Components

Database Component

CA Lisa Release Automation supported Database servers:

MySQL Version 5.1.30 and Higher

Oracle Version 10g and Higher

Microsoft SQL Server Version 2008 and Higher

Management Server (NAC)

The Data Management Server is the central point of a "CA Lisa Release Automation" deployment it has several functions which include the scheduling and process control, when a process is executed within the data management server the relevant instructions are sent to the agents via the Execution Servers that are assigned to the agents. The Data Management server also includes the UI server, when connecting to the Data Management Server with a browser the landing page has launch points to three UI's:

Release Operation Center – ROC, Data manager

As the controller of the system, the Center Server drives all executions, manipulates data in the database, calculates and updates flows states. The Center Server also temporarily holds all files that are needed for currently started executions.

Designer (ASAP)

IDE for generic application deployment modelling across multiple tiers. When launching the Automation Studio interface a Java Web Start application is downloaded to the local machine (Java JRE is required). The Java web start ensures that the latest client will be installed and updated.

Delivery Dashboard

Release management and control for tracking, measuring and improving release operations (Adobe Flash Player plugin required).

Repository

The repository provides storage for artifacts to be used in Release Operations Center releases and for all actions, core and custom, to be used in Automation Studio processes. The repository server (Nexus) is installed by default as part of the Data Management server installation. The installation of the Data manager includes an option to connect to an external repository.

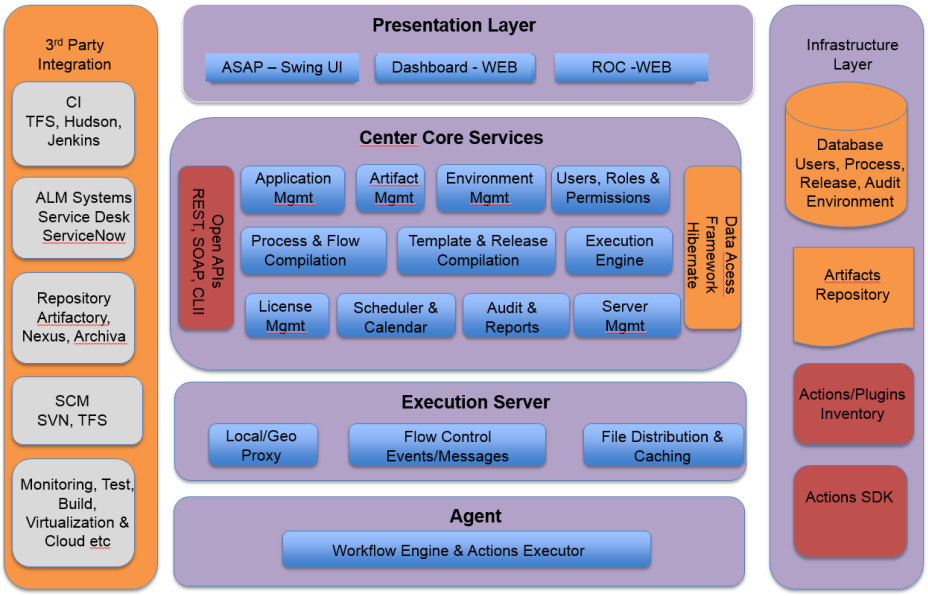
Execution Server (NES)

A CA Lisa Release Automation deployment must have at least one Execution server, The Execution Server mediates between the Center Server and the Agents. The Execution Server distributes control messages and files to the Agents and collects status data from the Agents which is retrieved later by Center Server. The Execution Server serves as a bridge when an Agent needs to Communicate with other Agents and a direct route is not available. More than one Execution Server can participate in the communication between two agents.

Agents (AGT)

All Machines that are part of the CA Lisa Release Automation environment require an Agent to be installed; agent installation can be remotely deployed from the IDE administration screen or installed manually. Agents are the main workers of the system. Each Agent is deployed on a server which it manages. The Agent gets the process flow and files from the Execution Server and sends process execution status back to the Execution Server. Agents can also be used to retrieve artifacts from the Repository.

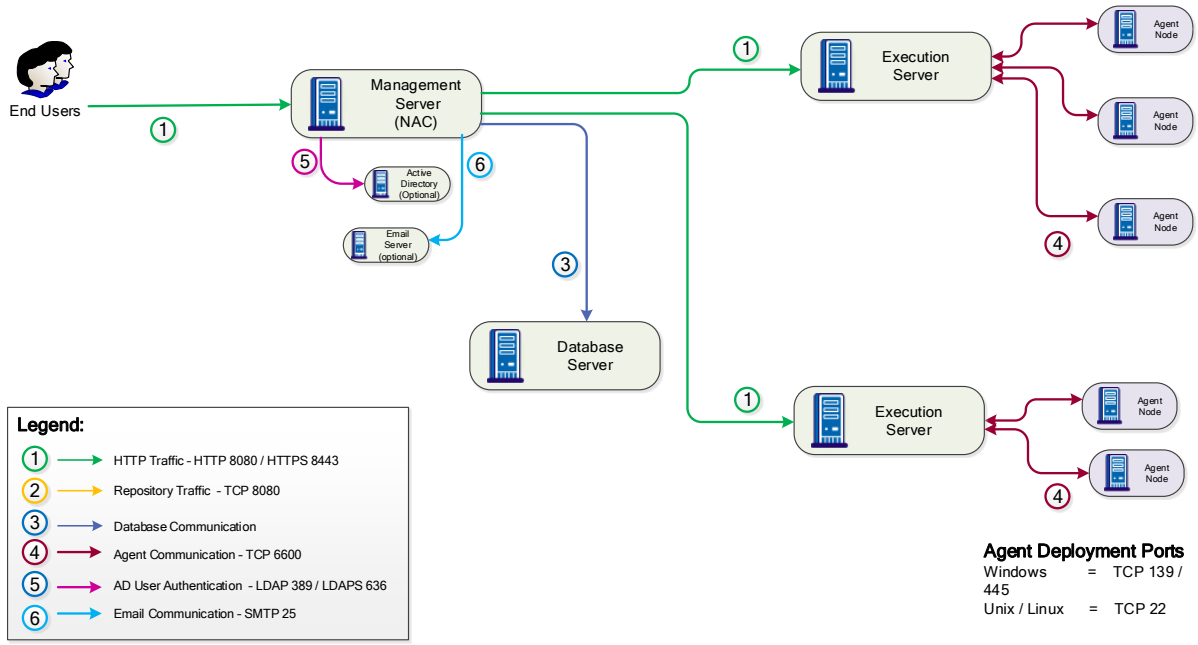
Logical Architecture



Release Automation Revision History

Release Date	Version	Description
23rd October 2013	4.7.1 b237	GA build – Build 237
28th April 2014	4.7.1 b322	Update including fix for CA Certificates – Build 322
28th April 2014	4.7.1 b343	Cumulative Fix – Build 343
11th June 2014	4.7.1 b360	Hot Fix 2 build 360 – includes build 322 and build 343
4th August 2014	4.7.1.b385	Hot Fix 3 build 385

High Level Component and Port Architecture



The schematic shows the various components that make up a typical CA Lisa Release Automation deployment.

Example Deployments

Overview	Database Server	NAC Server	NES Servers
1 NAC 12 x NES's 1300 x Agent's 23 Application's 4000 Deployments / Month	Linux RedHat 5.3 Oracle DB 11G R2 Virtual (Xen) 8vCPU 8GB Ram Disk 170 Gb	Windows 2008 Virtual (Xen) 8 vCpu's / 8GB Memory 75Gb Disk	Windows 2008 R2 64 Bit Virtual (Xen) 4Vcpu's 8Gb Memory Disk 75Gb Maximum 350 Agents
2 x NAC's 22 x NES's 1700 x Agents 25 Applications	Windows 2008 R2 SP2 Enterprise Clustered MS SQL 2008 R2 Standard Physical HP BL460 G6 8 Cores, 24gb Memory Disk – 350Gb Database 220Gb	SLES 11 Physical HP BL460 G6 8 Core, 12gb Memory	SLES 11 Physical IBM B4, 8 Core, 16Gb Memory Virtual 4vCPU, 16gb Memory, 50-300Gb Disk Maximum 350 Agents
1 x NAC 2 x NES's 218 x Agents 3 Applications 25 Deployments per week	Unix Server –Virtual (the database server is one of 6 VMS on the underlying physical server) Oracle DB 11G R2 CPU Threads: 28 Database Size 45Gb	Windows 2008 R2 Physical, Dual Processor Quad Core Hyper-threaded 8GB Memory 99Gb and 738Gb Disk	Windows 2008 R2 64 Bit Physical, Dual Processor Quad Core Hyper-threaded 8GB Memory Disk 715Gb Maximum 218 Agents

Architecture Guidelines

From an architectural perspective there is no specific limit to how many Execution Servers (and therefore how many agents) can be managed by one Release Automation Centre server. However, through working with our other customers, we have learnt that there are more practical reasons for limiting how wide a user community is that uses a single NAC.

• The main reason is that when large number of teams are sharing the same NAC, all having potentially different release cycles, it can be very hard to schedule a maintenance window for the Release Automation Server itself. So, for example, when an upgrade to the latest Release Automation version is required, it can be very hard to find a consistent time that is suitable for all users. By limiting the number of teams per NAC, each group has the ability to schedule upgrade activities around project commitments and increase the chance of planning a maintenance window.

• When a project wants to define its environments in Release Automation, the user may have to navigate through a very large list of irrelevant agents. There are features in Release Automation to assist with this, e.g. .the ability to group agents in to a custom folder structure, but having a smaller set of agents (e.g. Hundreds to thousands rather than thousands to tens of thousands) per NAC improves the general manageability.

• User and permissions management is simpler. This reduces the risk of incorrect configuration and users having access to agents and processes they should not have access to.

• Log files are easier to navigate without the additional messages created by all the additional agents. With this in mind then, we recommend planning to not have 1 single NAC for an entire estate of tens of thousands of servers, but instead plan to have a NAC per related business area. A related business area is one in which the use of a Release automation installation is on a similar / shared area, and where the users can plan required maintenance windows as discussed above.

The exact number of agents that can connect to a NES depends greatly on many factors, including:

• Number of executions performed in parallel

• The complexity of those executions

• The size of files transferred during the process executions

As such it is very difficult to accurately predict the limit. As a rule of thumb we normally advise that one NES can handle up to approximately 600 agents concurrently. Higher numbers may be possible depending upon the factors mentioned above. We typically find that a Release Automation implementation is not rolled out to all servers in one go, and that instead the agents are rolled out gradually over a period of time, team by team and environment by environment. In this way, it is possible to monitor the performance of the NES and verify workload with the demand that is being placed upon it. Should it be decided to include the Release Automation agent as part of the standard build on all servers, this is possible but we would recommend that the Release Automation service is installed in a latent state allowing phased roll out of Release Automation by simply activating the Release Automation agent services as the agents are brought in to use.

Another factor which can affect the number and location of NES's, is the network design and in particular the security and geographical aspects. If the NAC is not physically located with the target servers (e.g.it is in a separate datacentre), it is a good idea to place an NES near to those physical servers (see the Network Latency considerations section).

This allows the NAC to talk in an efficient manner with the more remote NES, and the NES can then have a higher volume of local communication with the many target servers. In terms of network security, a single NES can manage multiple environments including pre---production and production, but it may be necessary to consider separating some environments, e.g. Production, from the others---either from a pure network security perspective (maybe there is not one location that can see both pre---production and production) or from a risk and compliance viewpoint.

When using a repository / utility agent for artefact collection multiple agents should be deployed when supporting multiple environments

Execution Servers should be placed electronically close to agent machines where ever possible.

Deployments will generally include a mixture of single NES's and Super NES's, a super NES should be used where resilience is required such as production environment where as a test or QA environment may only require a single NES.

The effect of network latency varies depending upon which Release Automation components experience the latency. Between the NAC and NES, and between NES and NAG, latency can have the effect of increasing the overall execution time of a deployment process. This would be particularly noticeable if the process is transferring very large files across the affected connections. For this reason some customers have separated the file distribution aspects of a deployment process into one Release Automation process, and the actual deployment activities into a second process. The distribution process can then be executed at a convenient time before the deployment is actually required. If the latency is between the Release Automation client UI and the NAC, it can have different detrimental effects on the user, e.g. the time to load the client UI (as the UI has to load all the relevant applications, components, environment information etc.), and the time it takes to perform all actions (such as adding an action, deleting an action, changing a parameter value etc.). Latency can be reduced by taking into consideration the location of users and NACs and providing appropriate network connectivity between these locations.

The database is used to store all the data relating to your use of the product, including:

a) The processes themselves
b) The process parameters and their values
c) The environment definitions
d) Process execution results
e) Administration settings such as user accounts, LDAP configuration, permissions settings etc.

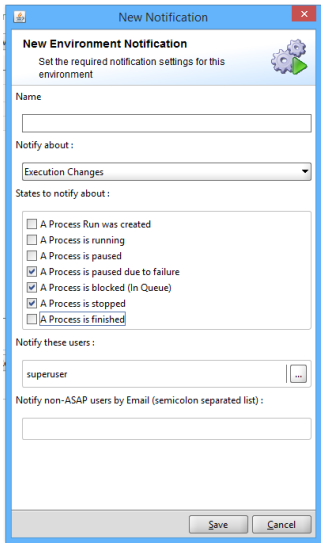
As such the amount of data created is heavily dependent upon a number of factors including the size and complexity of the processes, the number of executions performed in a given time period etc. As such we are not able to accurately predict the database growth until we can gather real usage metrics.

Monitoring Health and Availability

CA Recommends that to ensure that Ca Release Automation is functioning at its optimum the operating systems hosting the application servers are monitored using a system monitoring tool, the table below shows the recommended monitoring for each application server type.

Application Server	Metric	Description
Database	CPU / Memory	Monitor for High CPU or Memory usage
	Disk Capacity	Monitor the volume where the Release Automation Database is located to ensure that plenty of disk capacity is available.
NAC	CPU / Memory	Monitor for High CPU or Memory usage
	Disk Capacity	Monitor the volume where Release Automation and the Temp location volume is located to ensure that plenty of disk capacity is available.
	Windows Service	Ensure that the following windows service is running: ServiceName = NolioServer20
	Log file	nolio_dm_all.log - Monitor for a string “error “
Repository	CPU / Memory	Monitor for High CPU or Memory usage
	Disk Capacity	Monitor the volume where Release Automation and the Temp location volume is located to ensure that plenty of disk capacity is available.
	Windows Service	Ensure that the following windows service is running: ServiceName = RepoService
NES	CPU / Memory	Monitor for High CPU or Memory usage
	Disk Capacity	Monitor the volume where Release Automation and the Temp location volume is located to ensure that plenty of disk capacity is available.
	Windows Service	Ensure that the following windows service is running: ServiceName = NolioServer20
	Log file	nolio_exec_all.log - Monitor for a string “error “

As well as monitoring the operating system it is important for the CA Release Automation administer to be notified if any issues are being experienced with jobs/ processes. The simplest way to be notified of these events is to configure the SMTP server settings in the ASAP "System Settings" and to create Notification policies for each environment. The Status's that an administrator would find useful to be notified of are:



"A Process is Paused due to failure" – Diagnose Paused Process and identify the failing part of the process.

"A Process is blocked (In Queue)" - A Process has been Queued due to an agents which is required not being available. This is normally caused by another process being paused or failed which is blocking the agent.

"A Process is stopped"

Backup and Recovery Recommendations

When creating a Backup procedure for CA Release Automation the database is the most important component that required regular backups. It is recommended to complete a full Database backup weekly and incremental backup between these full back ups. If the CA Release Automation Nexus Repository is being used to store artifacts this should be backed up in the same way as the Database. The NAC / Repository and NES installation folders should be backed up after any major upgrades

Tuning tips

Data Manager (NAC) Tuning

For large environments it is recommended to add the following parameter "exection_server_request_time_out" to the systems setting and setting a value of 10

Parameter name	Parameter Value
Audit Design Changes	false
Audit Design interval	1
Audit Report Page size	100
GRACE_PERIOD_TIME	24
MAX_PROCESS_TAGS	5
Monitoring data: clean every (in minutes)	5
Monitoring data: clean servers every (in minutes)	5
Monitoring data: life length (in minutes)	1440
Monitoring data: life length of servers (in minutes)	2880
SMTP Email	lonmi06@ca.com
SMTP Password	*****
SMTP Port	25
SMTP Requires Logon	false
SMTP Server	mail.ca.com
SMTP User	
Show Deprecated Actions	false
Unix Agent Default Installation Dir	/usr/local
WAKE_UP_PERSISTENCY_SCHEDULE	1
Windows Agent Default Installation Dir	C:\Program Files\Nolio\NolioAgent
execution_server_request_time_out	10

Execution Server Tuning

Communication between execution servers and agents is via the protocol nimi which is configured in the \CA\LISAReleaseAutomationServer\conf\nimi_config.xml. Two important settings are:

1. The Capacity element setting is used to configure the maximum number of agents that can be registered to the Execution Server and the default value is 200. When 200 agents are registered to the execution server any further registration requests will be rejected.
2. The Warn-capacity.element setting should always be lower than the Capacity element setting. When the number of agent registered with the execution server exceeds this value the new connecting nodes will be asked to seek another supernode

The following thread settings can be changed when the execution server is handling a large number of agents.

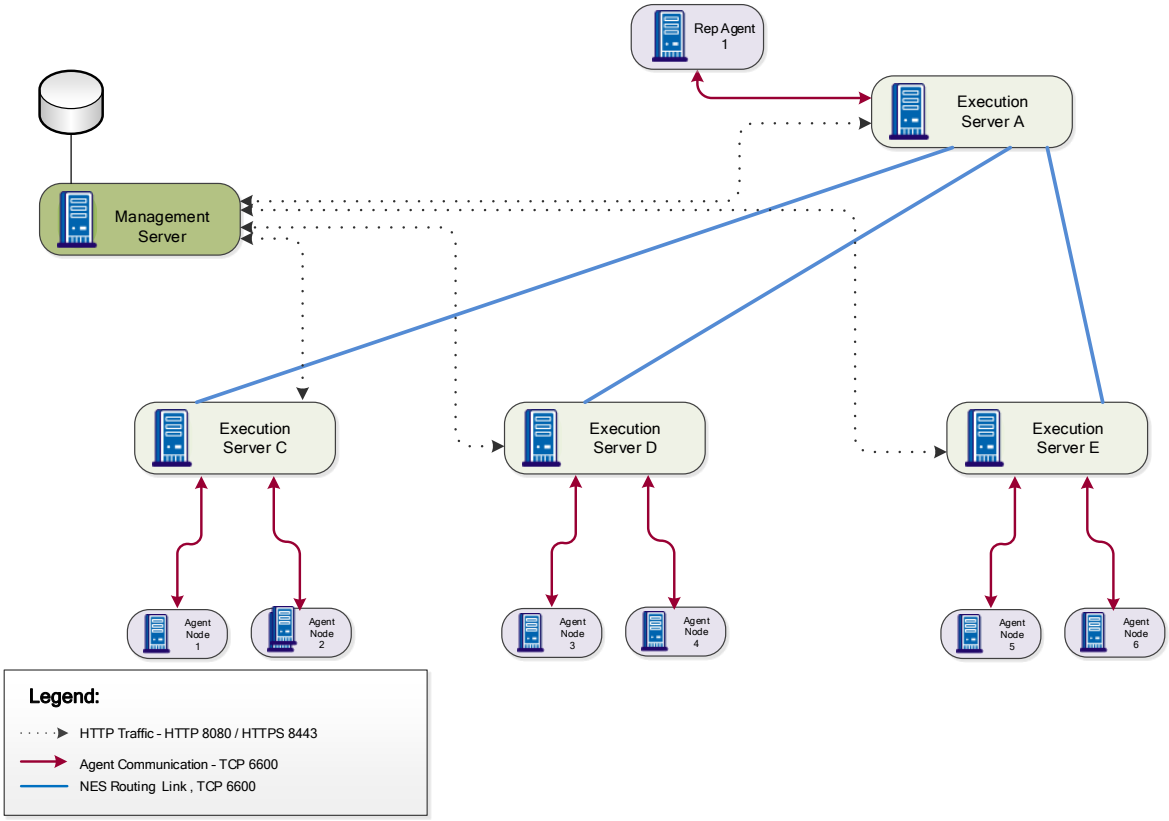
Parameter	**Default Setting**	**New Setting**
appmessages > threadpool	Size = 30	60
network > threadpool > client	Size = 2	4
network > threadpool > server	Size = 2	4
network > threadpool > outbound	Size = 30	60
network > threadpool > reverse	Size = 30	60
network > routing > threadpool	Size = 30	60
network > keepalive > threadpool	Size = 30	60
network > files > threadpool	Size = 30	60

Execution Server Logical Architectures

Execution servers communicate directly with agents and should be placed electronically close to the agents being managed. As well as the physical placement of Execution servers there is also a logical connectivity of the Execution Servers which is used for routing messages and files between agents. The following section contains examples of different NES architectures.

STAR Execution Server Architecture

In an architecture involving multiple datacentre's or environments the execution servers are installed within the datacentre's or separated environments, in this architecture a process can only copy files between agents attached to the same execution server. If the processes require access to other agents within Release Automation the execution servers can be connected in a STAR configuration as seen in the following schematic. Connecting the Execution servers in this logical arrangement allows any agent to copy files to any other agent. As an example if "Agent Node 1" is required to copy a file to "Agent Node 6", "Agent Node 1" will send a request to "Execution Server C" to look up the location of "Agent Node 6" as the Execution server does not have a connection to "Agent Node 6" the request will be passed to "Execution Server A", this execution server also does not have a connection to "Agent Node 6" and the request will be passed to "Execution Server E" as "Agent Node 6" is connected to "Execution Server E" this information is passed to "Agent Node1" and a route is defined between the two agents.

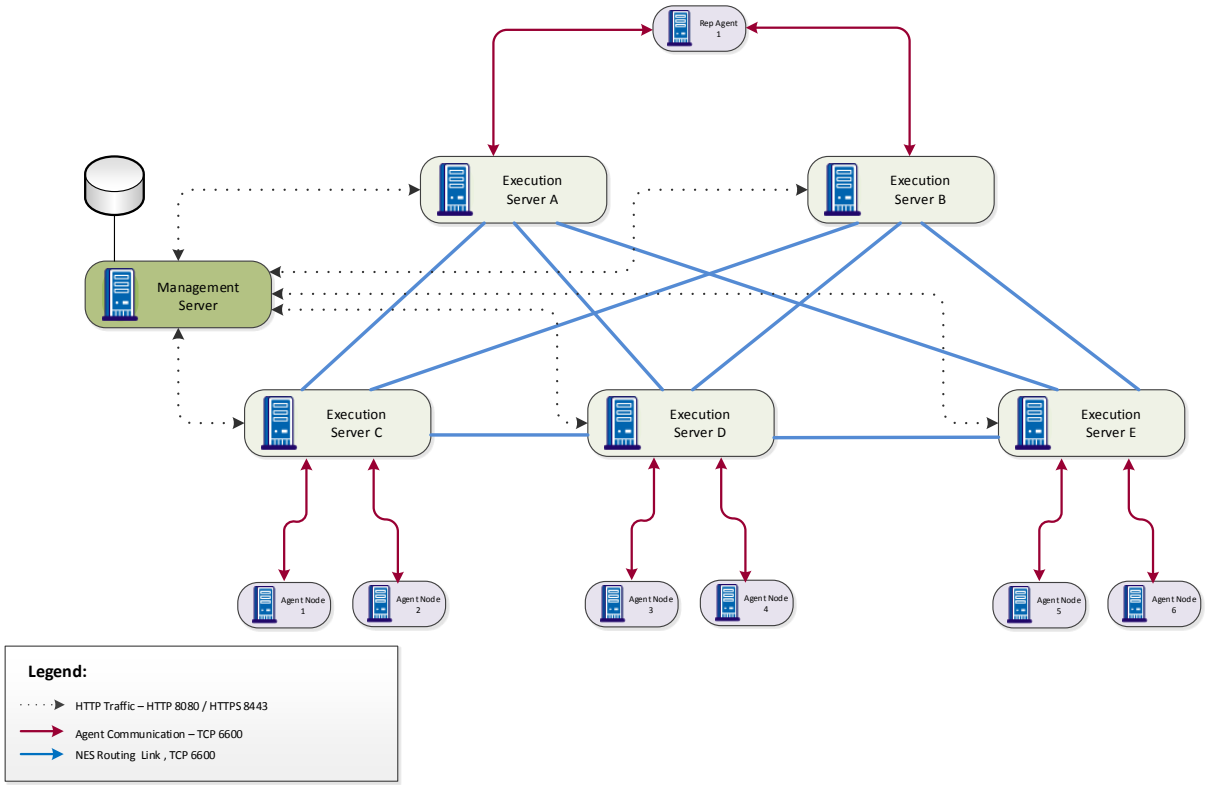


Note: Before configuring a STAR architecture ensure that the environment is upgraded to 4.7.1 b385 or later

STAR Execution Server with High Availability Architecture

The STAR Execution Server with High Availability architecture is currently being tested internally and should not be recommended to customers at this point

A single STAR architecture introduces a single point of failure if “Execution Server A” is unavailable this will stop any inter NES file transfers in this instance a STAR Execution Server with High Availability architecture can be used as per the following schematic, this architecture includes a secondary route between execution servers.

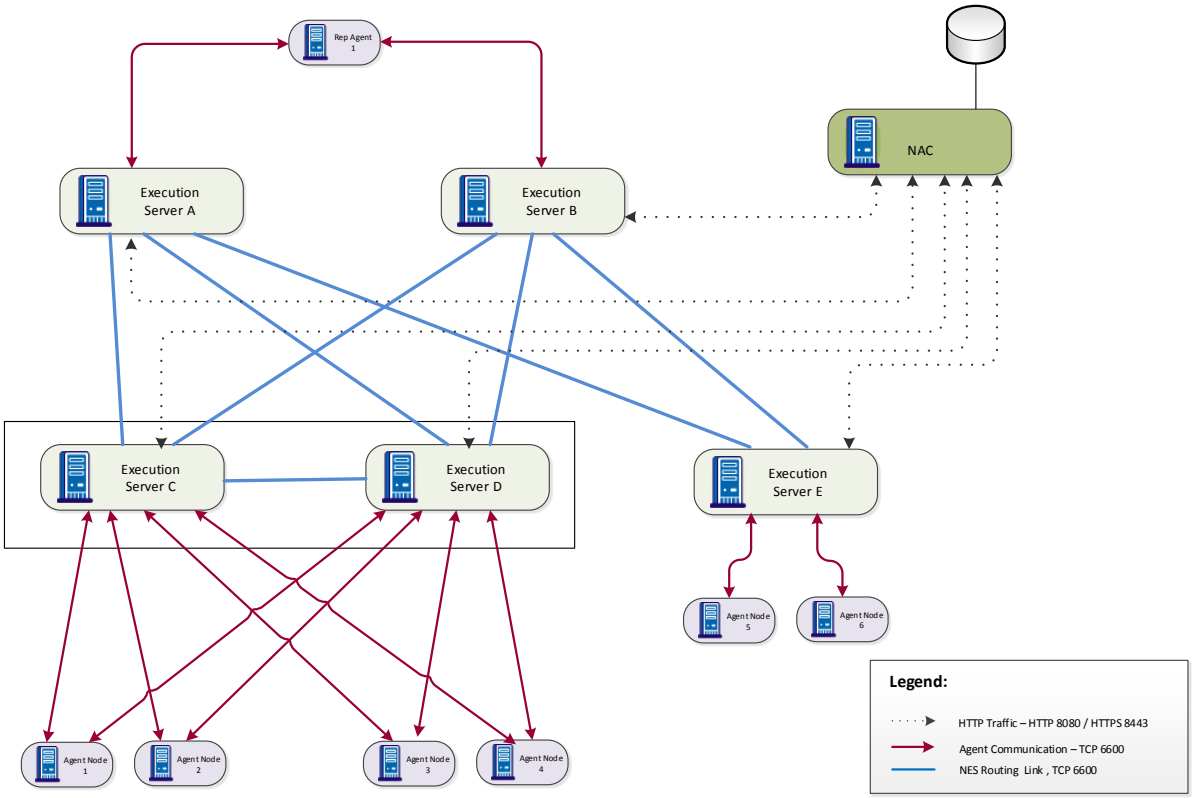


Note: Before configuring a STAR architecture ensure that the environment is upgraded to release 4.7.1 b385 or later

STAR Execution Server with High Availability Architecture and Super NES Layer

The STAR Execution Server with High Availability architecture is currently being tested internally and should not be recommended to customers at this point

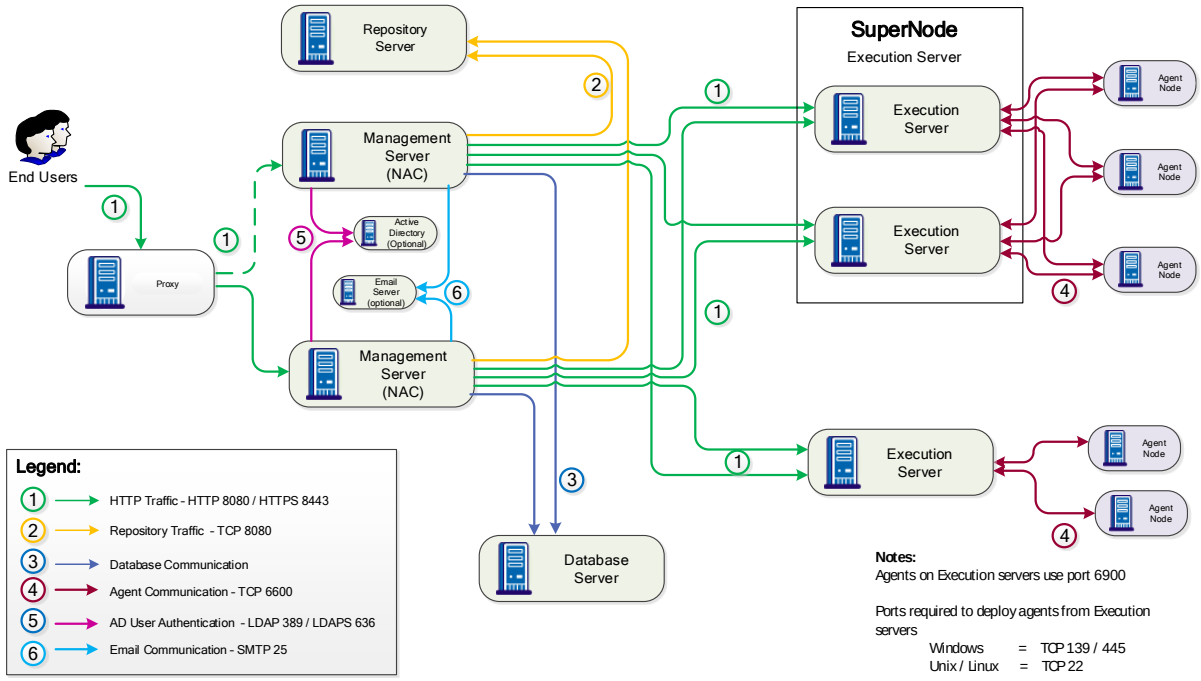
For complete fault tolerance Execution servers can be paired into what is called a “Super NES” and the agents can be configured to communicate with both execution servers, in the event of an execution failing within a “Super NES” the NAC will communicate with the agent via the remaining Execution Server.



Note: Before configuring a STAR architecture ensure that the environment is upgraded to release 4.7.1 b385 or later

Highly Available Architecture

The following section will explain how the CA Lisa Release Automation components can be deployed in a highly available environment and the steps required for configuring each of the components.



Data Management Center (NAC)

The Data Management Center component can be installed in a Highly Available (Active / Standby) configuration. The pre-requisites for installing the Active / Standby configuration are as follows:

1. Management Servers and proxy time must be synchronized to the second
2. Management Servers must be running on the same Operating System.
3. Management Servers and reverse proxy must be on the same LAN.

The steps to build this environment are as follows:

1. Install a Database server remotely from the Data Management Center and follow the steps in the administration guide to ensure the database is available for the installation of CA Lisa Release Automation.
2. Install a remote Nexus repository in a highly available configuration
3. To Install the Primary Data Management Center use the CA Lisa Release Automation custom installation do not install an Execution server, configure the Data Management Center with the details for the database server created in step 1 and connect to the highly available repository that was defined in step 2
4. On the Secondary Data Management Center use the CA Lisa Release Automation custom installation do not install an Execution server, configure the Data Management Center with the details for the same database server as the Primary node and connect to the highly available repository that was defined in step 2
5. A network Load Balancer is required to redirect network traffic to the Data Manager Master node the Load Balancer functionality is not part of the CA solution. The Load balancer must be configured using a weighted priority algorithm with the Preferred NAC having a priority of 1

Database Server

The Database Server should be highly resilient (Clustered) and be installed remotely from the Data management Server

Repository Server

When configuring the Data Management Center servers as highly available the repository server should be installed remote from the NAC and in a highly available configuration. Details on configuring the Nexus repository in an active passive configuration can be found on the Sonatype website https://support.sonatype.com/entries/21451383-How-to-Set-up-Active-Standby-Failover

Execution Server

Execution server routing should be configured to allow for redundant routing and Execution servers should be deployed to create “Super Nodes” in this configuration agents are configured with a primary and secondary Execution server if the agent cannot communicate with the primary execution server it will switch to the secondary execution server.

During the agent installation the Primary execution server details are stored in the <Nolio Agent root install directory>/conf/nimi_config..xml under the supernodes section. In order to add another Execution Server you should edit nimi_config.xml and add another value to the supernodes section, the value will include the IP/Hostname of the secondary Execution Server and the port that the Execution Server needs to bind to communicate with the Agent. Below you can see an example how of the Agent is configured to work with two NES

<supernodes>

<supernode>**Execution-SRV1:6600**</supernode>

<supernode>**Execution-SRV2:6600**</supernode>

</supernodes>

An alternative method to configure the execution servers that an agent is registered with is to select the agent / agents in the ASAP UI administration tab / Agent Management and Right Click on the agents selecting the option “Change Execution Server of the selected agents” all execution servers will be displayed if you select the execution servers that the agent should be registered with and click OK the agents will be reconfigured

Firewall Architecture Scenarios

The following section includes some examples of how Release Automation can be configured in a firewalled environment.

Scenario One – Agent outside firewall and only outbound traffic allowed

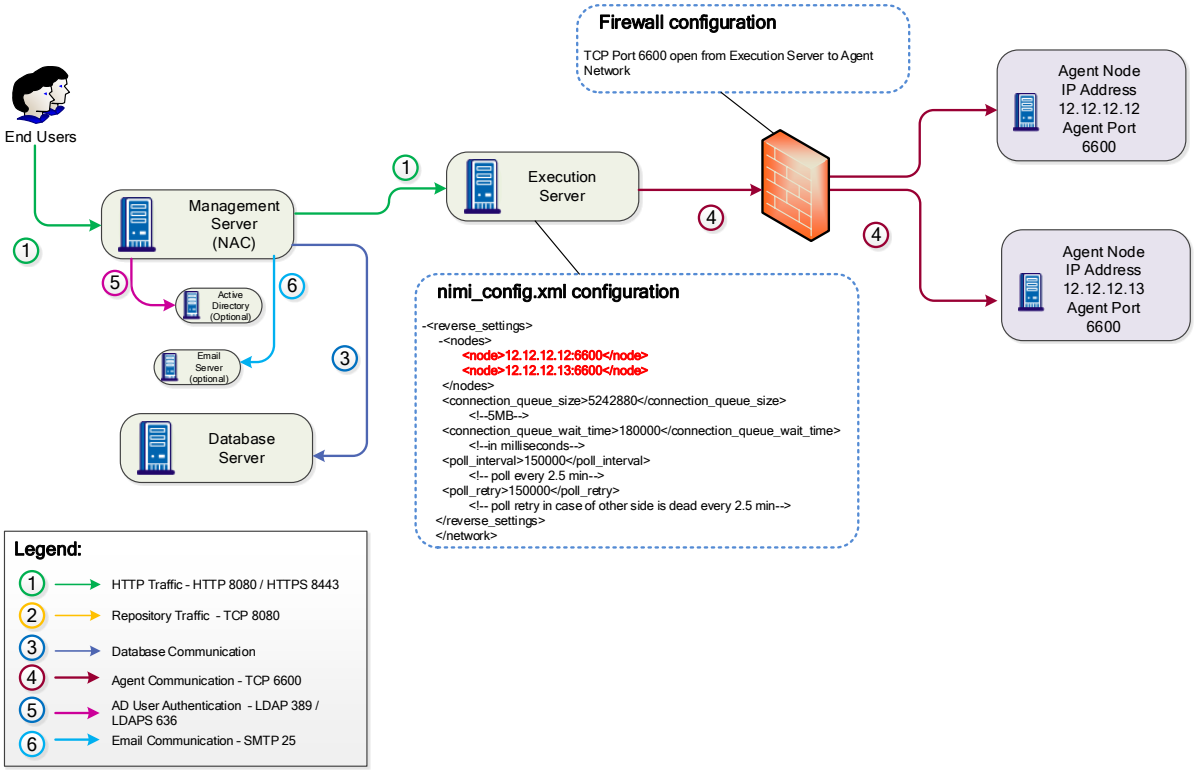
In the first scenario the servers to be managed with Release Automation are located outside of a firewall in a DMZ and network traffic is only allowed to travel from the internal network to the DMZ. Normally a Release Automation agent will connect to the NES but in this scenario the connection must be initiated from the NES these are the high level deployment steps:

1 The agent should be installed on the servers in the DMZ using the silent installer. The silent installer includes a switch that will override the default setting to validate the connection from the agent to the NES, For further details on this please refer to the CA SWAT guide – Zero touch deployment

2 Configure the firewall to allow traffic from the internal network to the DMZ using TCP port 6600

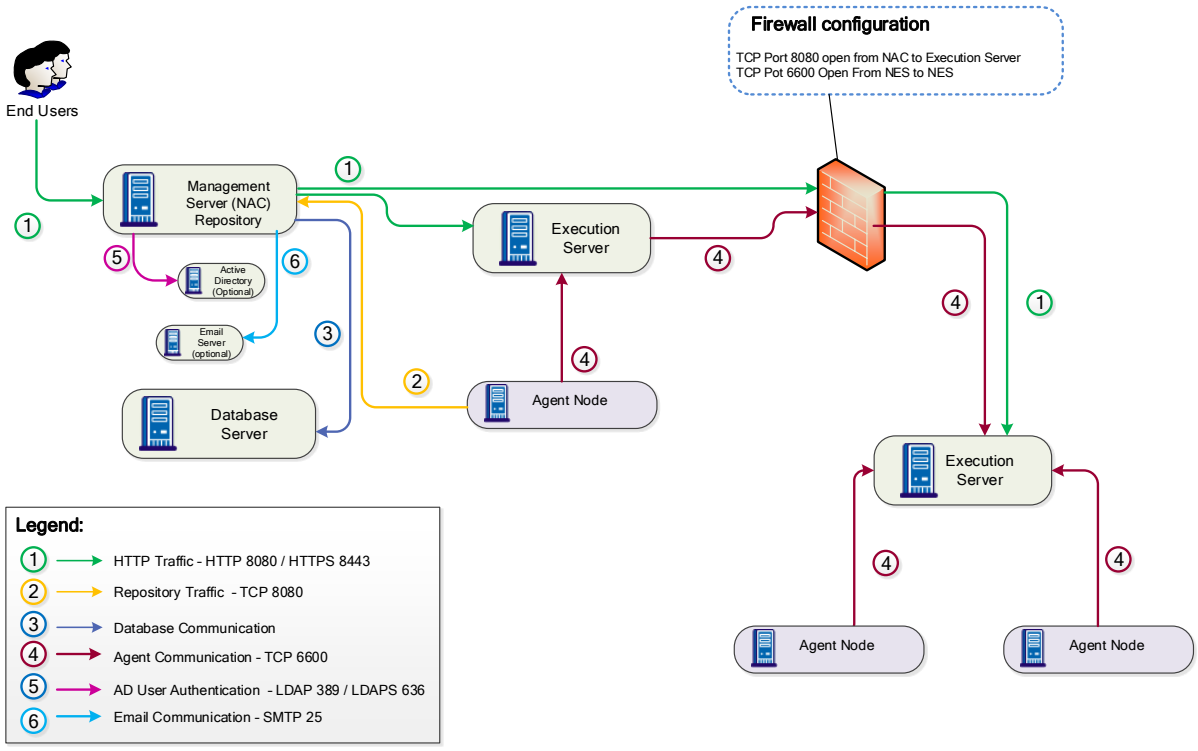
3 Configure the reverse settings section of the nimi_config.xml on the NES with the IP addresses of the agent outside the firewall.

4 Restart the NES service, the NES will periodically (2.5 Minute Default) attempt to connect to the agent and if the agent responds the session will be kept open through the firewall.



Scenario Two – Execution Server placed outside Firewall

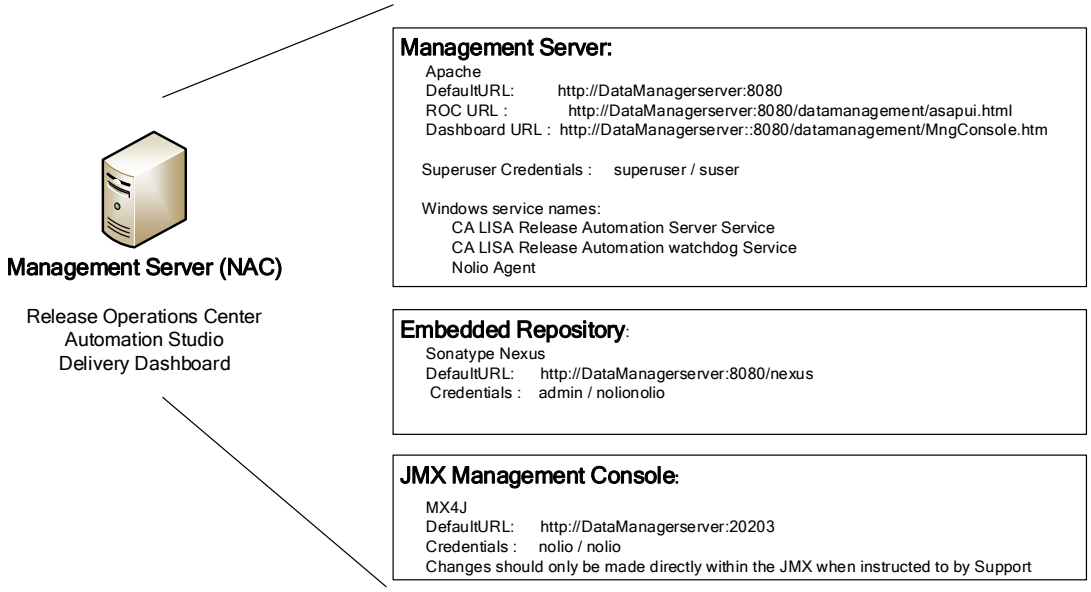
In the second scenario the servers to be managed with Release Automation are located outside of a firewall in a DMZ and network traffic, the agent inside the firewall is acting as a repository agent and is connected to a NES inside the firewall. NES routing has been defined between the two NES's



Appendix A

Data Manager Server - URL's / Default Credentials and Logs

The following graphic shows the various internal component's that make up a Data Management Server with the corresponding connection details.



Data Manager Log files overview:

1. nolio_dm_all.log
    a. Log file contains the following information:
        i. NAC start-up sequence.
        ii. DB connectivity
        iii. Amount of agents that connected to each NES and unreachable agents.
        iv. Status of processes execution
        v. Logged in users
        vi. Details about design and publish activities
    b. File Location = <Install dir>/logs/nolio_dm_all.log
2. nolio_document.log
    a. Contains information about processes that exported to xml document
    b. File Location = <Install dir>/logs/nolio_document.log
3. nolio_export.log
    a. Contains information about components/applications that imported/exported to/from the system
    b. LogFile Location = <Install dir>/logs/nolio_export.log
4. nolio_auditing.log
    a. Contains all design and administration changes (Note that audit report need to be enable)
    b. Log file Location = <Install dir>/logs/nolio_auditing.log
5. installation.log
    a. Contains a summary of system installation
    b. Log file Location = <Install dir>/.install4j/installation.log
6. installation.log.*

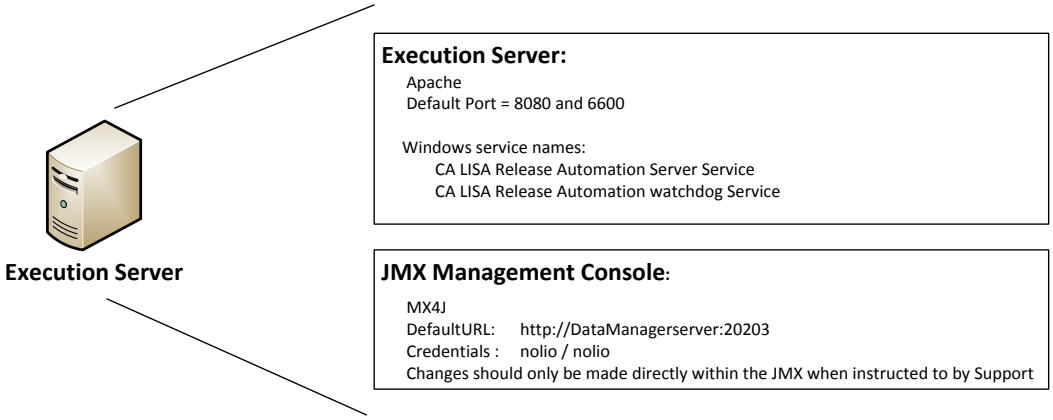
    a. Contains a summary of system upgrade from previous version
    b. <Install dir>/.install4j/installation.log.*

7. Agent_upgrade.log
    a. Contains a summary of agents upgrade
    b. Log file Location = <Install dir>/logs/Agent_upgrade.log
8. Installation log can be found in %temp% folder

Execution Server (NES) Internals - URL's / Default Credentials and Logs

The execution server is the link between the NAC / Data Manager server and the Agent Nodes, requests from the NAC / Data Manager to agents are sent via the Execution server. Additional execution servers can be added when working with complex networks, remote data centers or to scale out the solution. An execution server by default is configured to up to 200 Agents in normal operation, this figure could be higher if the workload for the agents is particularly low.

The following graphic shows the various internal component's that make up an Execution Server with the corresponding connection details.



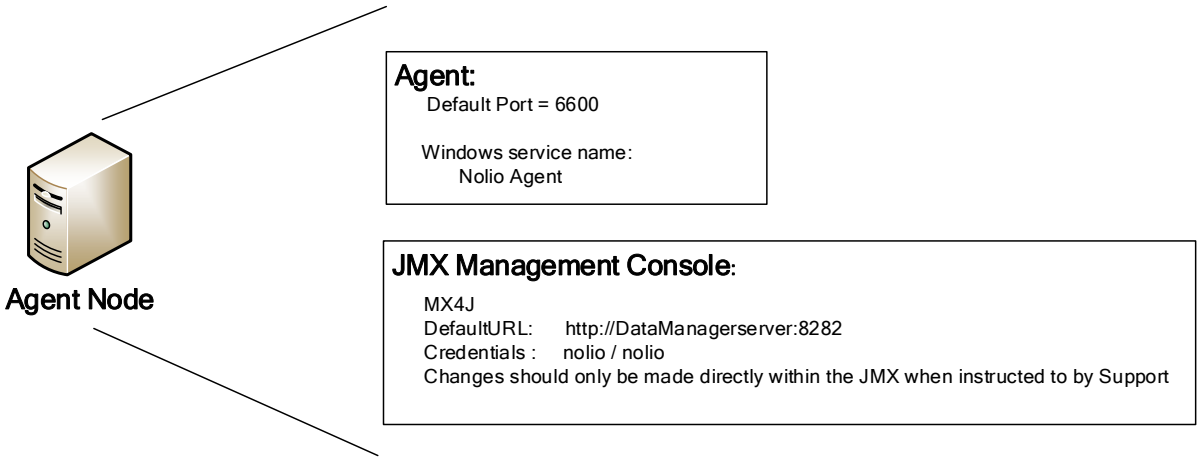
Execution Server Log files description:

1. Nimi.log
    a. Contains Information regarding communication between NAGs and NES such as handshake activity
    b. Contains Network topology (NAG and NES versions, ID's IP's etc.)
    c. Contains Information regarding parameters values and files that transfers between NAGs.
    d. Log file location = <Install dir>/logs/nimi.log
2. Nolio_exec_all.log and execution.log
    a. Contains Information regarding execution events and parameters that transfers between NAG's NES and NAC.
    b. Contains Remote agent installations logging
    c. Log file Locations = <Install dir>/logs/Nolio_exec_all.log, <Install dir>/logs/execution.log
3. installation.log
    a. Contains a summary of system installation
    b. Log file Location = <Install dir>/.install4j/installation.log
4. installation.log.*
    a. Contains a summary of system upgrade from previous version
    b. <Install dir>/.install4j/installation.log.*
5. Installation log can be found in %temp% folder

Agent Node (AGT) – Default Ports and Logs

The agent node communicates with the execution server and utilises a proprietary communication protocol called nimi.

The following graphic shows the core component's that make up an Agent node



Agent node Log files description:

1. Nimi.log
    a. Contains Information regarding communication between agent and NES
    b. Contains Information regarding parameters values and files that transfers
    c. Log file location = <Install dir>/logs/nimi.log
2. Nolio_all.log
    a. All NAG activity except the network layer (stored in nimi.log)
    b. Log file Locations = <Install dir>/logs/Nolio_all.log
3. Nolio_action_exe.log
    a. Contains specific information about actions executions and their results.
    b. Log file Locations = <Install dir>/logs/
4. installation.log
    a. Contains a summary of system installation
    b. Log file Location = <Install dir>/install4j/installation.log
5. Installation log can be found in %temp% folder