Log File Management Tool Deployment and User's Guide

Installing Workbench Agent (Mass Deployment) for LFMT 8.5.1

4/15/2025

Contents

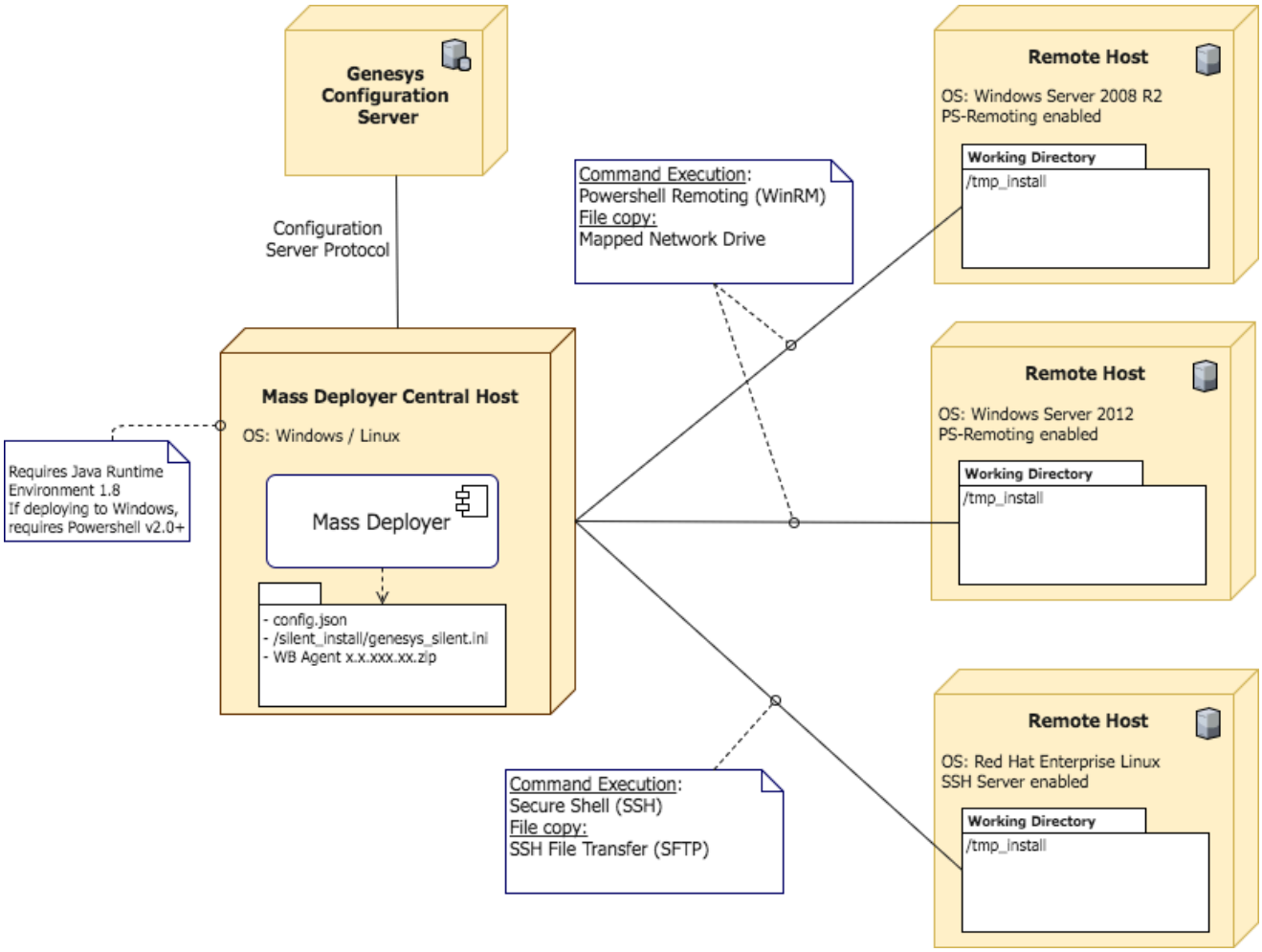
Installing Workbench Agent (Mass Deployment) for LFMT 8.5.1

The Mass Deployer allows the installation of the Workbench Agent to multiple hosts. It includes the following elements:

Mass Deployer Central Host The server where the mass deployer will be executed. It connects to the Remote hosts via different protocols depending on the Remote Host operating system. It requires a configuration file that provides credentials and connection details about the remote hosts.

Remote Hosts The hosts where the Workbench agent will be deployed. The agent will be extracted in the working directory specified in the Mass Deployer configuration file.

Configuration Server The properties of Remote hosts are retrieved from this server, including their IP address in order to connect to them. Every Workbench Agent deployed in Remote Hosts is provisioned in this Configuration Server.



Prerequisites

Software Requirements The Central and Remote hosts should have one of the following operating systems:

Windows

- Windows Server 2008 R2
- Windows Server 2012

Linux

- Red Hat Enterprise Linux (RHEL) 7 or later

> **Important**
>
> - If the central host is a Linux host it will only be able to mass deploy the agent to Linux Remote hosts.
> - If the central host is a Windows host, it can mass deploy to other Windows and Linux Remote hosts.
> - If a Workbench Agent is already deployed to a remote host, the agent must be uninstalled prior to running the Mass Deployer. This is a limitation that will be addressed in a future release.

Before using the Mass Deployer, the following software must be installed in the central and remote hosts:

- Java™ Platform Standard Edition Runtime Environment 8, 64-bit (JRE™ 8)
- If using a Windows Server, make sure that Powershell is enabled. The Mass Deployer requires Powershell v2.0 or a later version.

Communication Requirements

Linux Mass Deployer Central Host:

- The central host should be able to reach the Linux remote hosts using SSH.

Linux Remote hosts:

- The remote hosts where the agent will be deployed should be able to receive SSH connections from the Mass Deployer Host. An SSH Server (e.g. OpenSSH) should be enabled. Commands and files will be sent using SSH and SFTP respectively.

Windows Mass Deployer Central Host:

- Powershell should be enabled. The Mass Deployer requires v2.0 or a later version.
- The remote hosts should be included in the Trusted Hosts List of the central host in order to correctly connect to them using Powershell Remoting (Windows Remote Management – WinRM).
  - The Mass Deployer will try to temporarily modify the trusted host list in order to connect to the remote hosts. To do this, the Mass Deployer should be executed from a command prompt with administrative privileges. Otherwise, the Mass Deployer will continue to execute, but the trusted host list should be modified by the user before starting the Mass Deployer.
  - You can use the following command replacing the “*” with the list of remote host IP addresses. The “*” would indicate that all hosts are trusted.

Set-Item WSMan:\localhost\Client\TrustedHosts -Value "*" *–Force*

Windows Remote Hosts:

- Powershell should be enabled, v2.0 or later is required.

- The remote hosts should be able to receive remote Powershell commands. The WinRM service should be enabled to allow this remote access. A way of enabling it is running the following command from Powershell with administrative privileges:
  - Enable-PSRemoting –Force
  - This will start or restart the WinRM service and create the listener to accept requests from any IP address.
- The Mass Deployer will copy the Installation Package and extract it in a folder called “tmp_install” inside the provided working directory. If it doesn’t exist, the Mass Deployer will try to create it.

Execute the following command in Powershell on the central host to ensure that WinRM remoting is properly configured in a remote host and is accepting requests:

Test-WSMan -ComputerName <Remote Host IP address>

It should print an output similar to:

- wsmid: http://schemas.dmtf.org/wbem/wsman/identity/1/wsmanidentity.xsd
- ProtocolVersion: http://schemas.dmtf.org/wbem/wsman/1/wsman.xsd
- ProductVendor: Microsoft Corporation
- ProductVersion: OS: 0.0.0 SP: 0.0 Stack: 3.0

The “Stack” version will change depending on the remote host WinRM service.

Installing the Mass Deployer

The Mass Deployment software is available in the **Utilities** folder of the **Collector** installation directory, post installation of the Collector.

For Example: C:\Program Files\GCTI\collector\LFMT_Collector_8.5.100.05_Alt\utilities\agent_mass_deployer

The working directory of the Mass Deployer in the central host should have the following structure:

```
/<Mass Deployer working directory>
     Mass Deployer-x.x.xxx.xx.jar
       /deployments  Stores the silent install configuration files for each remote host where
the agent is deployed
      /silent install
             genesys silent.ini  A template for the silent install configuration file.
             config.json The Mass Deployer configuration file. See “Configuring the Mass
Deployer” for details about this file.
             WB Agentg x.x.xxxx.xx.zip The agent ZIP distribution
       /lib Directory with the Mass Deployer runtime dependencies
```

Configuring the Mass Deployer

The configuration of the Mass Deployer is stored in a JSON file used by the tool at runtime.

The configuration file has the following sections:

{ “config_server” : CfgServer,

“app_template_name” : <string>,

“app_name_prefix” : <string>,

“app_parent_folder” : <string>,

“deployments”:Deployments,

“agent” : <string>,

“linux_ip_path” : <string>,

“windows_ip_path” : <string>,

“wb_server_app” : <string>,

“global_listening_ports” : GlobalListeningPorts,

“global_options” : GlobalOptions,

“windows_global_options” : WindowsGlobalOptions,

“linux_global_options” : LinuxGlobalOptions }

CfgServer section:

Includes the connection parameters required to interact with the Configuration Server.

Required: Yes

{ "ip_addr": <string>

"port" : <number>,

"username" : <string>,

"password" : <string>,

"app_client_name" : <string> }

- ip_addr - The IP address of the Configuration Server
- port - The port where the Configuration Server is listening
- username -The username used to connect to Configuration Server
- password -The password of the associated username
- app_client_name - The Application object used to connect to Configuration Server

app_template_name The name of the Workbench Agent Application Template. This template will be used to provision all agents. Required: If provisioning to Configuration Server.

app_name_prefix The prefix that will be used to name the new Workbench Agent Applications that will be provisioned. The name of the Application will be <app_name_prefix><Remote Host name> Required: If provisioning to Configuration Server.

app_parent_folder The folder where the application object will be stored in Configuration Server. If this folder doesn't exist, the Mass Deployer will try to create it. If not provided the application will be created in the root of the Applications section. Required: No

Deployments section:

Includes the properties of the hosts where the Agent will be deployed. It is a list with one object per deployment.

Required: Yes

[{ "cfg_server_host_name": <string>

"username" : <string>,

"password" : <string>,

" working_dir" : <string>

}, . . .]

- **cfg_server_host_name** - The name of the remote host as it appears in Configuration Server
- **username** - The username used to connect to the remote host using SSH or a Powershell Remote session.
- **password** - The password of the username
- **working_dir** - The directory where the agent will be extracted and from where it will be executed

> **Important**
>
> - Please ensure that each CME Host in Configuration Server has an Operating System in the "OS Version" property; this is used by the Mass Deployer to decide how to connect to the remote host.

agent

Path to the Workbench Agent distribution. If deploying from Windows, the path should have double backslashes (\\). For example: /home/genesys/WBAgent_8.5.000.31.zip.

Required: If deploying to remote hosts.

linux_ip_path

Relative path to the Linux installation package directory inside the agent package. For example: /IPs/WBAgent_UNIX_8.5.000.31/linux/b1/ip.

Required: If deploying to remote hosts.

windows_ip_path

Relative path to the Windows installation package directory inside the agent package. It requires double blackslashes. For example: \\IPs\\WBAgent_Windows_8.5.000.31\\windows\\b1\\ip.

Required: If deploying to remote hosts.

wb_server_app

Name of the Workbench Server application that will interact with the Agent being deployed/provisioned, as it appears in Configuration Server. This is used during the installation of the Workbench Agent.

Required: If deploying to remote hosts.

GlobalListeningPorts section:

The list of listening ports that will be used when provisioning the Workbench Agents in Configuration Server. Each port is a JSON Object with an ID and a value.

Required: If provisioning to Configuration Server.

[{ "ID": <string>

"value" : <number>

}]

- ID - A unique identification for each port
- value - The listening port number

GlobalOptions section:

A JSON object with the options that will be assigned to the Workbench Agents being provisioned. Options are grouped by sections and all values should be strings.

Required: If provisioning to Configuration Server.

{ <section_name_1> :

{ <option_name_1>: <option_value_1>,

<option_name_2>: <option_value_2>,

... <option_name_n>: <option_value_n>

}, <section_name_2> :

{ <option_name_1>: <option_value_1>,

... <option_name_n>: <option_value_n>

}, ..., <section_name_n> :

{ <option_name_1>: <option_value_1>,

... <option_name_n>: <option_value_n>

} }

For example, these options:

- Option: **log/all** – Value: C:\logs\workbench_logs
- Option: **log/expire** – Value: 20
- Option: **general/customer_name** – Value: default

Would be configured as follows:

{ “log” :

{ “all”: “C:\\logs\\workbench_logs”,

“expire”: “20” }, “general” : { “customer_name”: “default” } }

WindowsGlobalOptions section:

A JSON object with the options that will be assigned to the Workbench Agents if being provisioned to a Windows host. The structure of this JSON object is the same as the one described in the GlobalOptions section. If the same property is set in both the GlobalOptions and the WindowsGlobalOptions section, the latter will be used.

Required: No.

LinuxGlobalOptions section:

A JSON object with the options that will be assigned to the Workbench Agents if being provisioned to a Linux host. The structure of this JSON object is the same as the one described in the GlobalOptions section. If the same property is set in both the GlobalOptions and the LinuxGlobalOptions section, the latter will be used.

Required: No.

Executing the Mass Deployer

Run the Mass Deployer by executing the following command from the directory where the executable was placed:

java -jar <MassDeployer-x.x.xxx.xx.jar> , followed by these arguments:

Short argument: -c

Long argument: --cfgFile

Mandatory: Yes

Valid Values: A path to a valid configuration JSON file

Description: Path to the Mass Deployment Configuration file

Short argument: -m

Long argument: --mode

Mandatory: Yes

Valid Values: provision, deploy, provision_deploy

Description: Mass Deployment mode.

*provision: Connects to configuration server and provisions the Agent apps in the configuration file.

*deploy: Copies the agent to the remote hosts and installs it.

*provision_deploy: Provisions the agent apps in Configuration server and then installs the agents on the remote hosts.

Short argument: -e

Long argument: --failOnError

Mandatory: No

Description: Cancels the execution if one of the deployments fail. If not included, the mass deployer will continue deploying to other hosts even if one of the previous deployments failed.

Short argument: -o

Long argument: --overwriteApps

Mandatory: No

Description: Overwrites the Config Server Apps if they are already provisioned. If not included, the mass deployer will fail if an application previously existed.

Examples:
java –jar MassDeployer-x.x.xxx.xx.jar –c /home/genesys/MD/config.json –m provision_deploy
java –jar MassDeployer-x.x.xxx.xx.jar –c C:\Users\genesys\MD\config.json –m provision –e –o
java –jar MassDeployer-x.x.xxx.xx.jar –-cfgFile /home/genesys/MD/config.json –m provision_deploy --failOnError

Important

- The Mass Deployer executes the installer of the Workbench Agent in **silent mode**.
- This version of the Mass Deployer will not monitor the outcome of the silent installation; this is accomplished by checking the **genesys_install_result.log** file that is generated in the working directory of the Agent Application.

Important

- As a known issue, the Mass Deployer might not work properly if there are disconnected Network drives in the central host.
  - Try removing these network drives before running the Mass Deployer.
  - If you see an error message similar to: “Attempting to perform the InitializeDefaultDrives operation on the 'FileSystem' provider failed” when opening a new Powershell terminal, then the Mass Deployer will not function appropriately.

- It has also been observed in some occasions that Powershell processes will remain active after the Mass Deployer has finished executing.
  - Please check the running applications after finishing deployment and manually clean-up any open Powershell processes