Remote Execution Framework for Hyper V Ecosystem

Software Design Document

TABLE OF CONTENTS

1. INTRODUCTION

1.1 PURPOSE

Today in the industry, we often come across the need of accessing virtual machines running onto a server. While several methods exist to access and execute any operation or installation on VMs, those methods require an individual to practically go to each VM and then execute the operation. Considering the scope of requirement of industry, there can exist multiple virtualization servers with multiple numbers of virtual machines running on each of them. Since there can exist large VMs on server, going to each VM and then offloading and installing a file is not feasible. Along with these reasons, file transfer over windows platforms has always been a tedious thing to implement. This project explores the file transferring alternative over MS-Windows using a smart technology called BITS (Background intelligent transfer system). BITS implementation has been carried out with powershell interface for hyper-v ecosystem. Powershell acts as a operating interface for manipulating the infrastructure.

Remote execution framework for hyper-V enables user to remotely offload and install any executable file over multiple VMs running on hyper-V Server. Thus this product provide a proper framework through which an administrator can view all VMs on a server, fetch their inventory details, and select file/files located on virtual directory for execution on the VMs. The product's main area of exploration is the offload of executables from virtual directory hosted on IIS server to the user selected VM using BITS (Background Intelligent Transfer System) download. Objective of this framework is to use capabilities of Powershell 3.0 to interface Hyper-V infrastructure, in order to carry out offloading of files remotely.

1.2 SCOPE

- Connecting to a machine and verify if this is a valid Hyper-V Installation.
- Accessing the Hyper-V Powershell Interface and getting virtual machines (VM's) inventory.
- Offloading an executable from a management box (any machine running the proposed application) to each of these VM's or VM's hosted on the connected hyper-v.
- Executing the installer silently to perform the installation and retuning the execution status.
- Initiating a UI installation of the executable, if it supports one.

- Creating a GUI which will aid in achieving the above mentioned tasks and will provide rich experience to the user.

1.3 OVERVIEW

- Document thoroughly briefs about the purpose, targeted issues and methodologies used to implement project. Along with this, it explains about technologies involved in the product which are essential for the product to fulfill the scope of project. Besides these ,other objectives of the document are to detail all the essential points like –
    - need,
    - objective
    - Exhaustive study and research regarding the proposed concept.
    - Assumptions made while designing the framework.
    - Methodologies.
    - Architecture and component designs

- Document includes supportive materials and references which were referred while proposing the issue by the product.

1.4 REFERENCE MATERIAL

- technet.microsoft.com
- OOMD (Object oriented modeling and design)

1.5 DEFINITIONS AND ACRONYMS

- **Powershell 3.0**

  Windows Powershell 3.0 is an essential management and automation tool that brings the simplicity of the command line to next generation operating systems. Inc1luded in Windows 8 and Windows Server 2012, and portable to Windows 7 and Windows Server 2008 R2, Windows

Powershell 3.0 unprecedented power and flexibility to everyone from power users to enterprise network administrators and architects.

- **Windows presentation foundation (WPF) 4.5**

  WPF is a modern graphical display system for Windows. Its a radical change from the technologies that came before it, with innovative features such as built-in hardware acceleration and resolution independence. WPF is the best toolkit to use if you want to build a rich desktop application that runs on Windows Vista, Windows 7, and Windows 8 in desktop mode (as well as the corresponding versions of Windows Server). 1

- **BITS (Background Intelligent Transfer System)**

  Background Intelligent Transfer Service (BITS) is a component of Microsoft Windows XP and later operating systems that facilitates prioritized, throttled, and asynchronous transfer of files between machines using idle network bandwidth and provides progress in-formation related to the transfers. BITS is most commonly used by Windows to download updates to your local system.

- **WinRM (Windows Remote Management)**

  WinRM is the Microsoft implementation of WS-Management Protocol, a standard Simple Object Access Protocol (SOAP)-based, firewall-friendly protocol that allows hardware and operating systems, from different vendors, to interoperate.

1.6 ASSUMPTIONS AND LIMITATIONS

- Administrator credentials are assumed to be known already for each virtual machine.
- Remote offload and installation currently not supported on Linux based virtual machines.
- All files are assumed to be stored already over a virtual directory hosted on IIS server.

2. SYSTEM OVERVIEW

Overall system is designed as a client-server architecture where each involving component has specified tasks.

Client Side system would involve-

- Interactive GUI based on
- .NETData organization
- Implementation of transport interface.

Server side system includes management box on which the product is hosted.

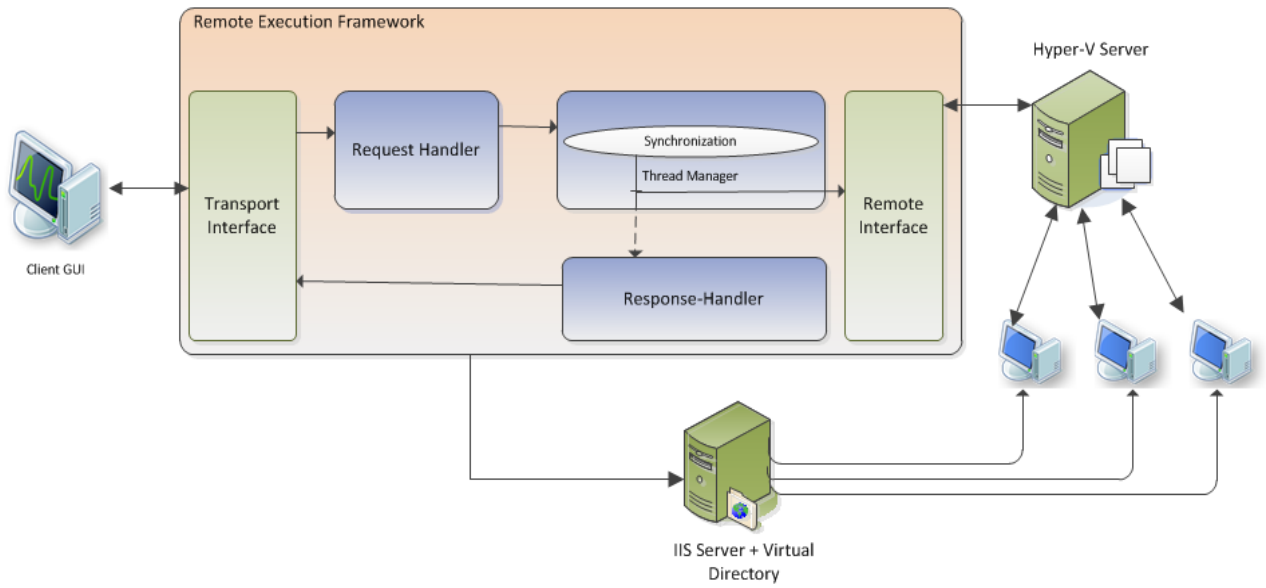
Management box simply involves-

- Implementation of transport interface to receive requests from GUI.
- Request handler, to de-serialize the objects.
- Thread Manager.
- Response Handler.
- Remoting interface to the hyper-v infrastructure.

The system also involves the IIS (Internet Information Service) Server. Server maintains a virtual directory containing all executables as well as non-executables to be offloaded onto the VMs. This IIS Server may or may not be hosted on the management box.

3. SYSTEM ARCHITECTURE

3.1 Architectural design.



Transport Interface	Flexible interface which connects user (client) machine with Management Box.
Request-Handler	Identifies type of request also remoting interface type. Responsible for creating Job Objects.
Response handler	Contains a synchronizer which would allow threads to have 'serialized' access to the component. Converts the response object into application understandable format and sends it to the transport interface.
Remote Interface	Remote interface is responsible for managing all operation executions on hyper-v server. It communicates with the hyper-v infrastructure performs operations and returns execution response to the invoking thread.
IIS Server	IIS server maintains a virtual directory which contains all executables. These executable can be downloaded by VMs using BITS download.

Hyper-V	Microsoft supported virtualization tool which allow creation of MS or non-MS based VMs.
Thread manager	Responsible for queuing incoming request objects. Also assigns thread from thread pool and forwards them to remote interface for execution.

3.2 Decomposition Overview–

➢ Thread Manager

1) Object queue -

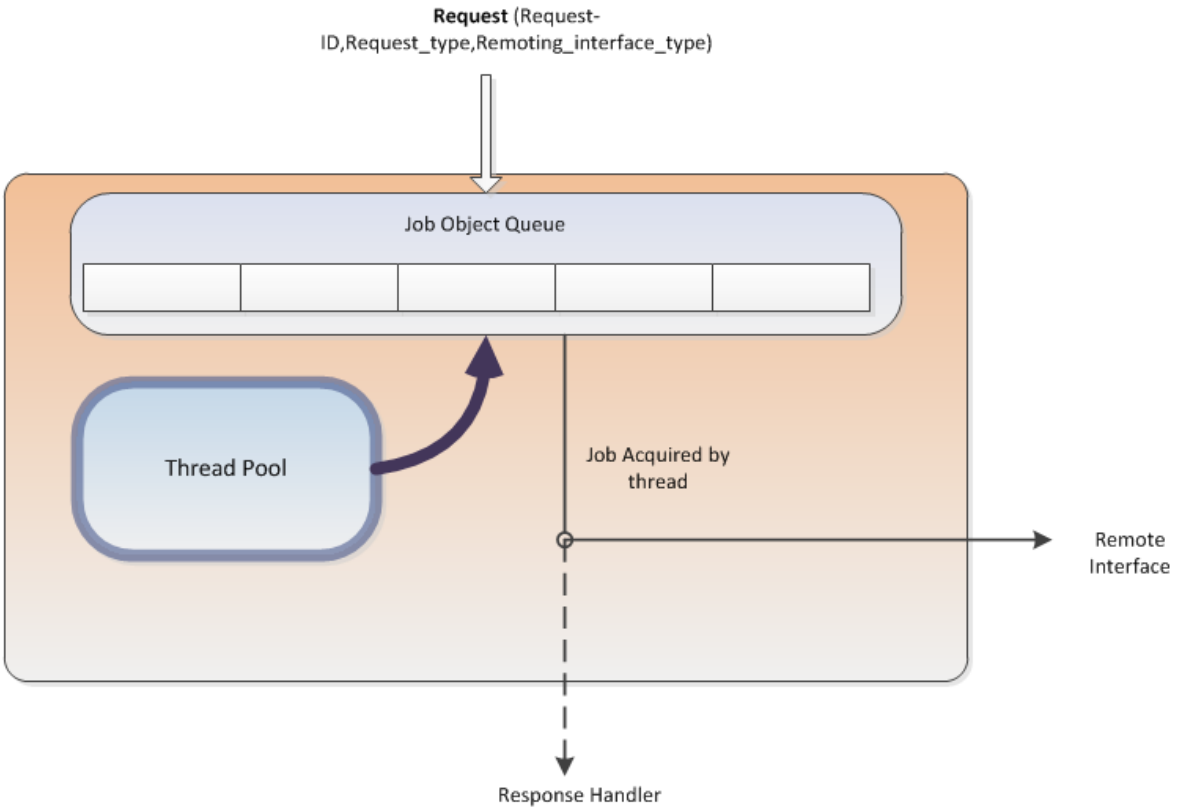
Thread pool is capable of assigning limited threads per instance. This object queue ensures that job objects sent by request handler would be first put up in queue , from where the 'idle state' thread would pick up a job object and execute it.

Job objects would be of various types such as offload , install ,authentication, inventory fetch.

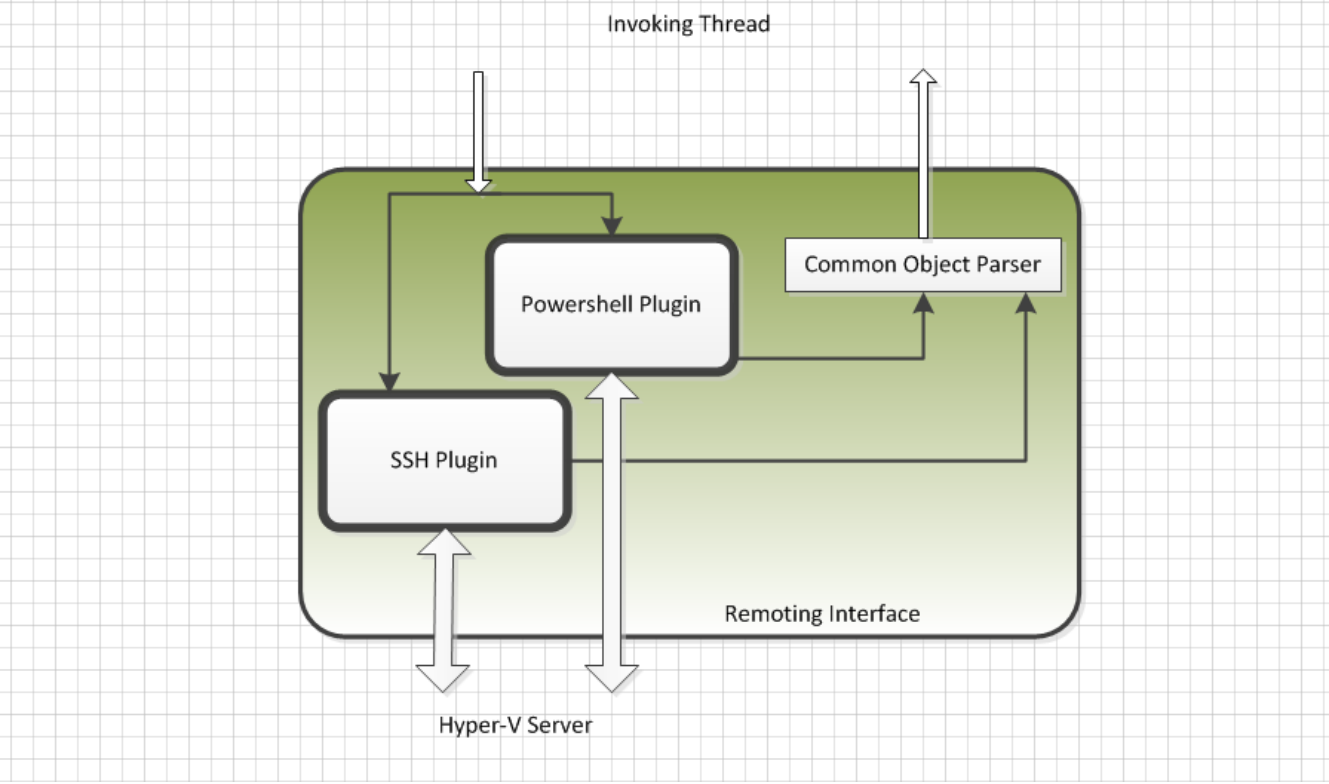
2) Thread Pool -

Accepts the incoming request and accordingly assigns the idle thread for particular thread pool. Multithreading is the core concept which has to be implemented in order to facilitate the multiple requests sent by GUI component. These threads run independently from each other existence, and invoke a particular remote interface operation as shown in fig.2.

After receiving response from the remote interface, thread forwards its response to the response handler.


Request (Request-ID,Request_type,Remoting_interface_type)
Job Object Queue
Thread Pool
Job Acquired by thread
Remote Interface
Response Handler

➢ Remoting Interface



- **SSH interface -**

  This plugin interface is required if the product is required to expand its execution capabilities over to the Linux based VMs. This interface would then interact with the VM hosted on hyper-v server.

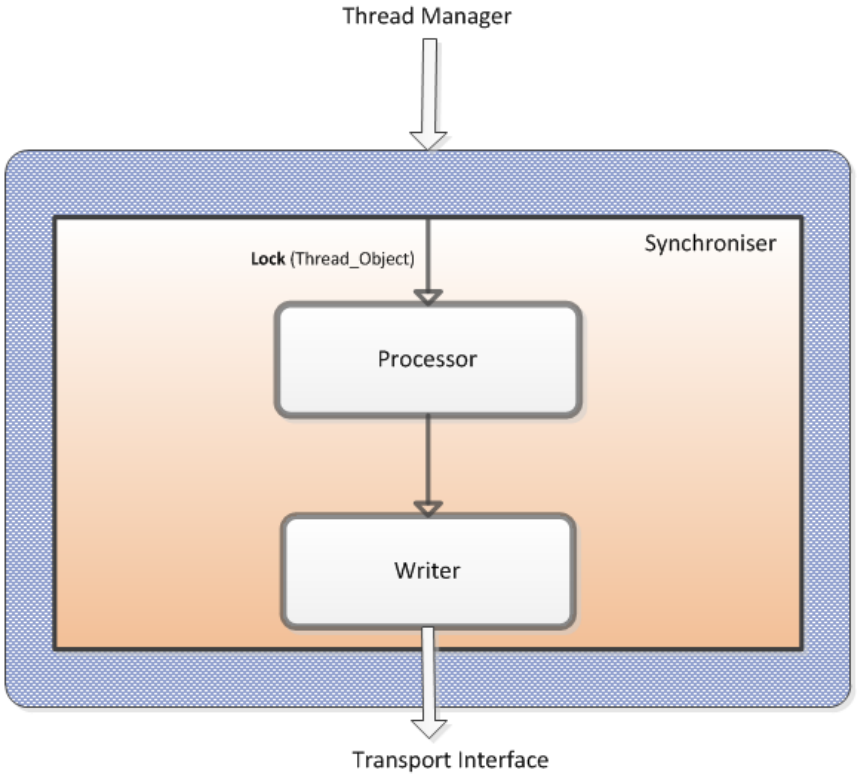
- **Powershell Interface**

  Required interface to manage and operate tasks on hyper-v as well as VMs (MS Windows supported.) based on hyper-v Sever.

- **Common object parser**

  This sub-component returns response to the invoking thread object in a particular format consisting of just required details. Information may consist of PSObject consisting inventory info, Status information detailing about the task completion or failure details.

➢ Response Handler



- **Synchronizer -** Response handler should allow access to one thread at a time. When a thread with a response arrives at response handler, the synchronizer mechanism makes sure only one thread accesses the component at a time, until it has been converted in application understandable format by **Processor** which then forwards the object to **writer.**

  Writer then accepts input object and transfers it to transport interface. After the thread has been forwarded, the synchronizer then releases the locked thread object, and accepts the next incoming response.

3.3 Design Rationale

- **Client- Server Architecture.**

Remote Execution framework for Hyper-V dominantly focuses on providing functionalities to the user. User (client machine) where GUI is responsible for conveying requests, may or may not be located at a same place as that of management box. A management box however, needs to a central entity or component which would execute tasks and send response to the client machine. Along with this client-server architecture would ensure scalability and adaptability, as any new component/s can be added or updated without altering the client-side GUI.

Considering the above mentioned scenarios, client-server architecture would eventually provide flexibility and scalability to the product framework.

- **Multithreaded design approach.**

The product is supposed to provide a multi-utility GUI which would **not** restrict user to perform single operation at a time and wait until the completion of that operation. In order to provide this functionality , main thread needs to be relieved of tasks such as authentication, offload, install, fetch inventory, hence a separate thread pool is required which would create independent threads for each of various functionalities.

Thus multithreading would ensure GUI would work and issue various request regardless of completion of previous request.

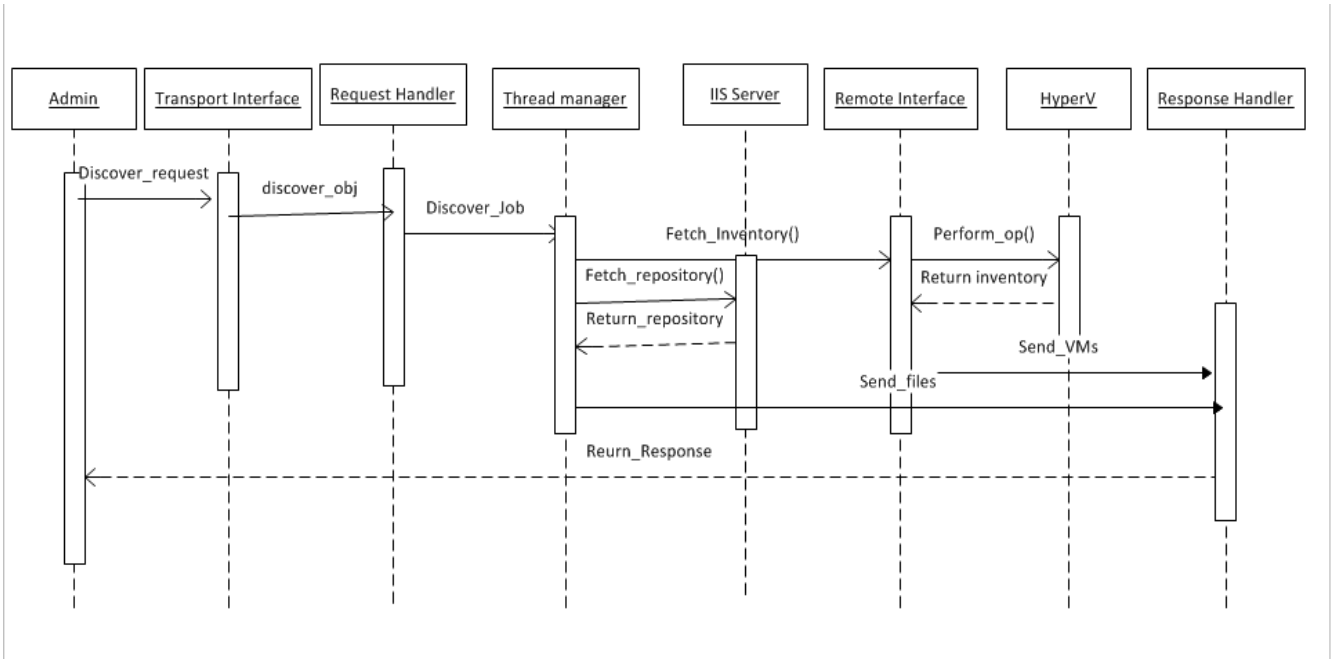
- **WPF based GUI.**

The WPF (Windows Presentation Foundation) is a powerful, reliable and an elegant solution for providing platform to create GUI. It not only would provide attractive GUI but also would ensure the run-time events that need to be handled dynamically.

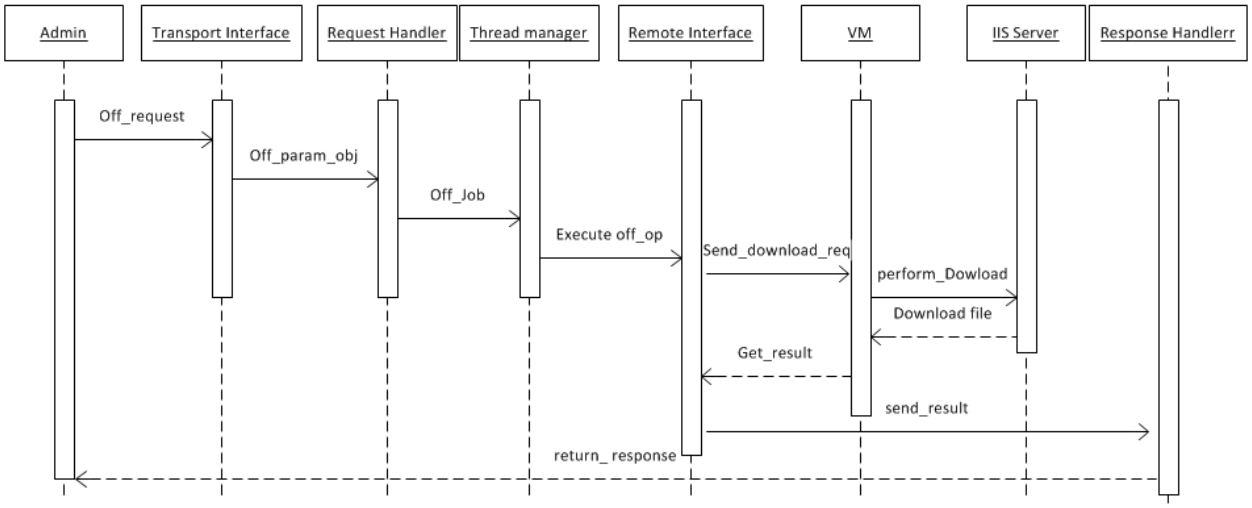
4. DATA DESIGN

- **Workflow/ Sequence Diagrams.**

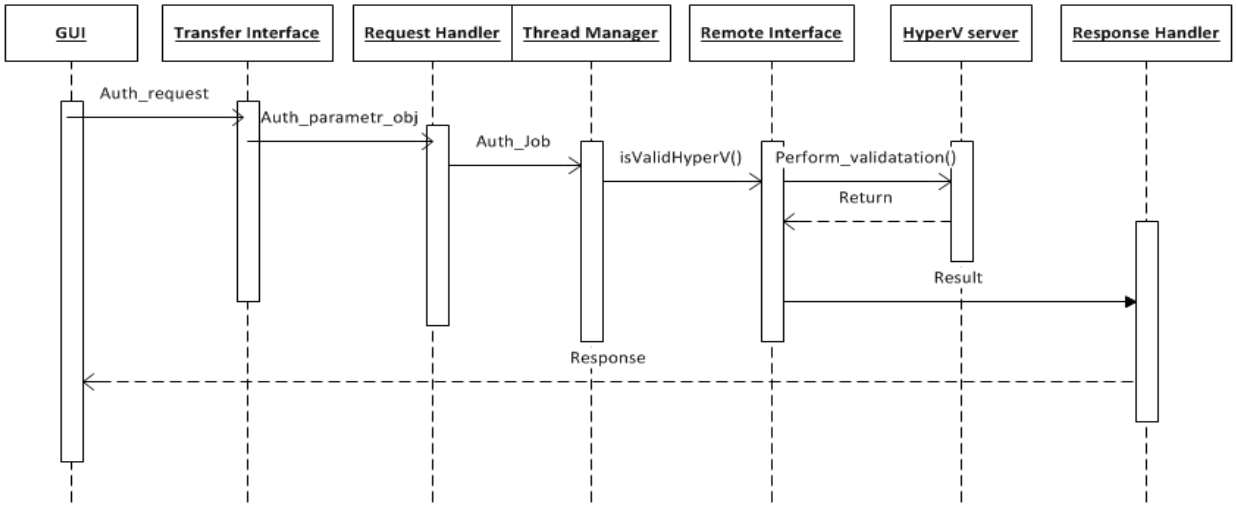
  - Discover inventory and IIS Files.



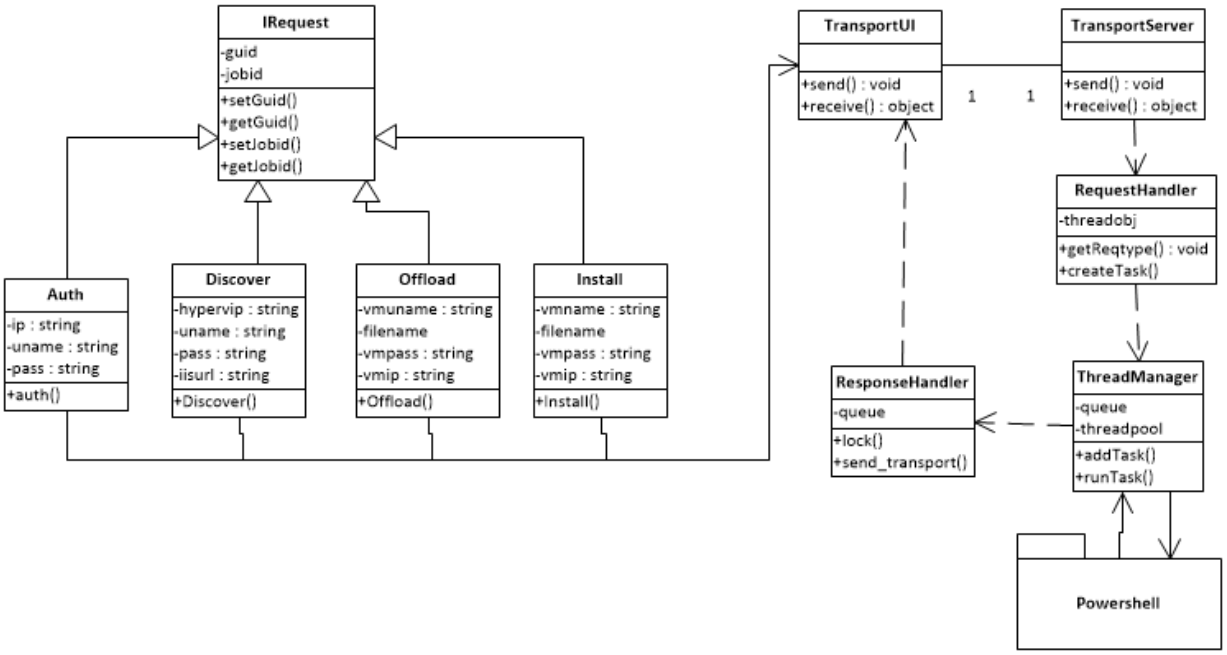
  - Offload Files on VMs.



- Authentication of valid hyperv installation.

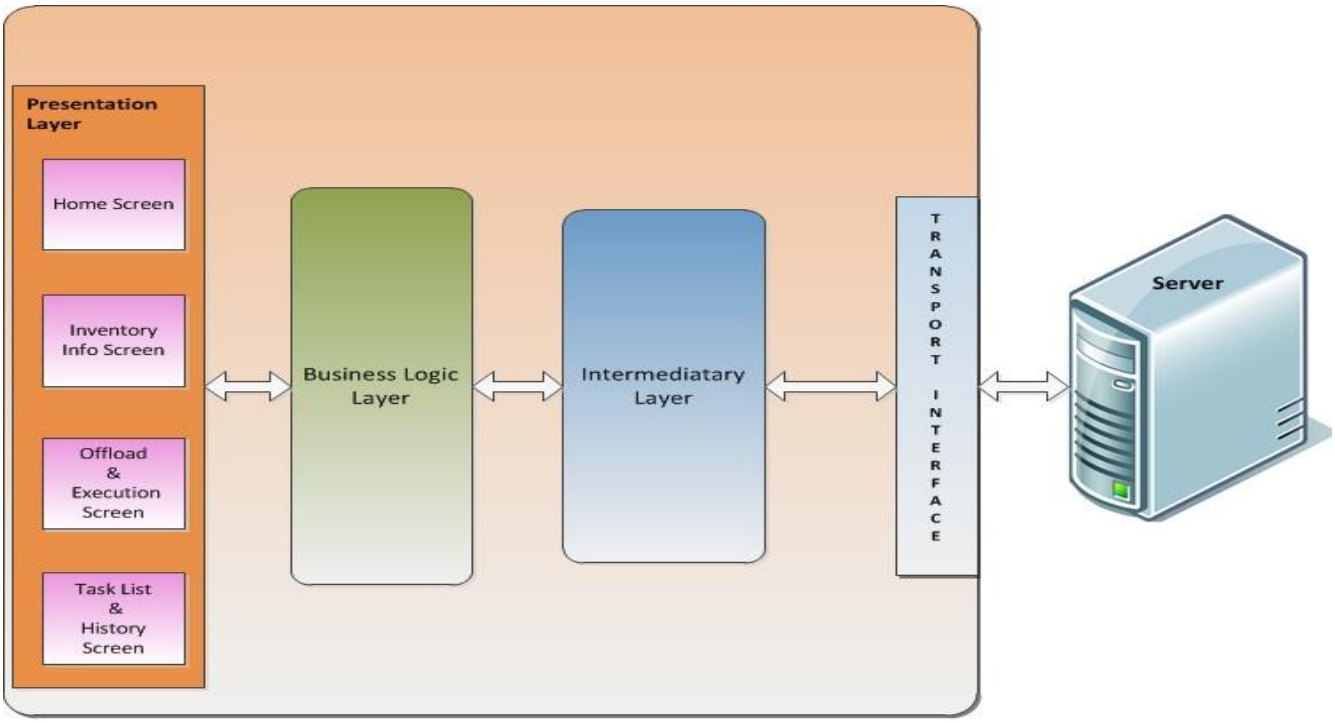


5. COMPONENT DESIGN



6. HUMAN COMPUTER INTERFACE DESIGN

6.1 OVERVIEW OF USER INTERFACE



The UI architecture consists of 4 layers:

1. Presentation Layer:
   It contains the actual presentable matter.

2. Business Logic Layer:
   This layer deals the actual logic behind the UI elements.

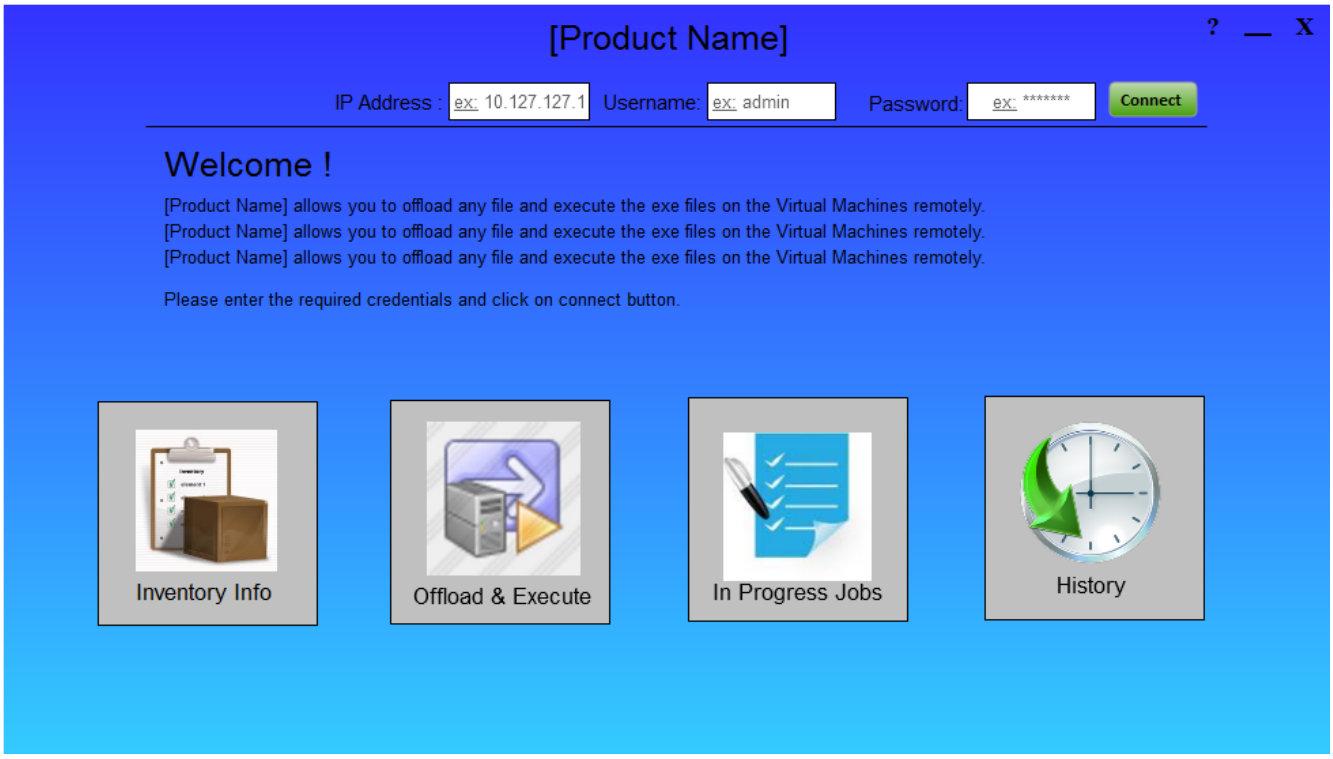
3. Intermediatory Layer:
   This layer will the convert UI understandable code to Server understandable code.

4. Transport Interface:
   This layer will be responsible for the data transfer between server and the UI.

6.2 SCREEN IMAGES

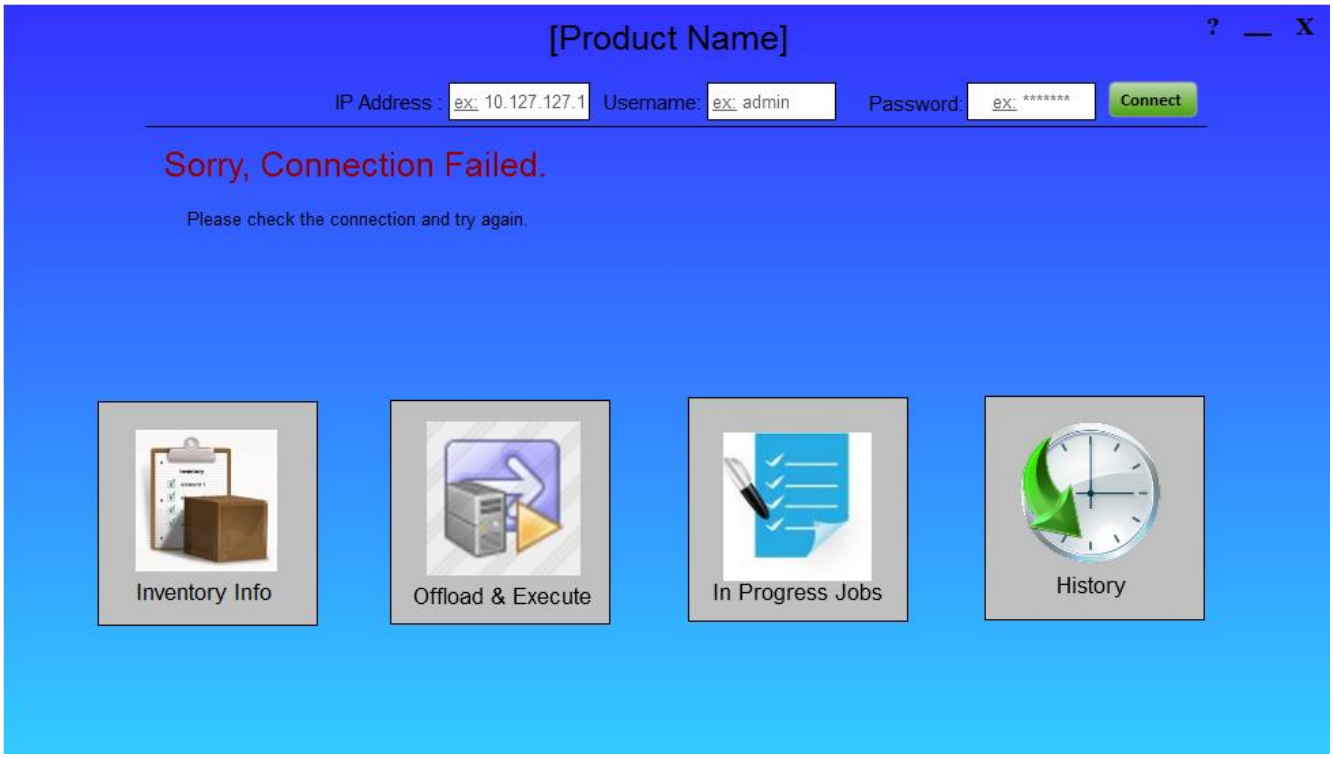
- **Login Page:**

  This screen accepts the credentials for authentication. At the same time it displays the basic functionalities of the product. The tabs at the bottom half are disabled.



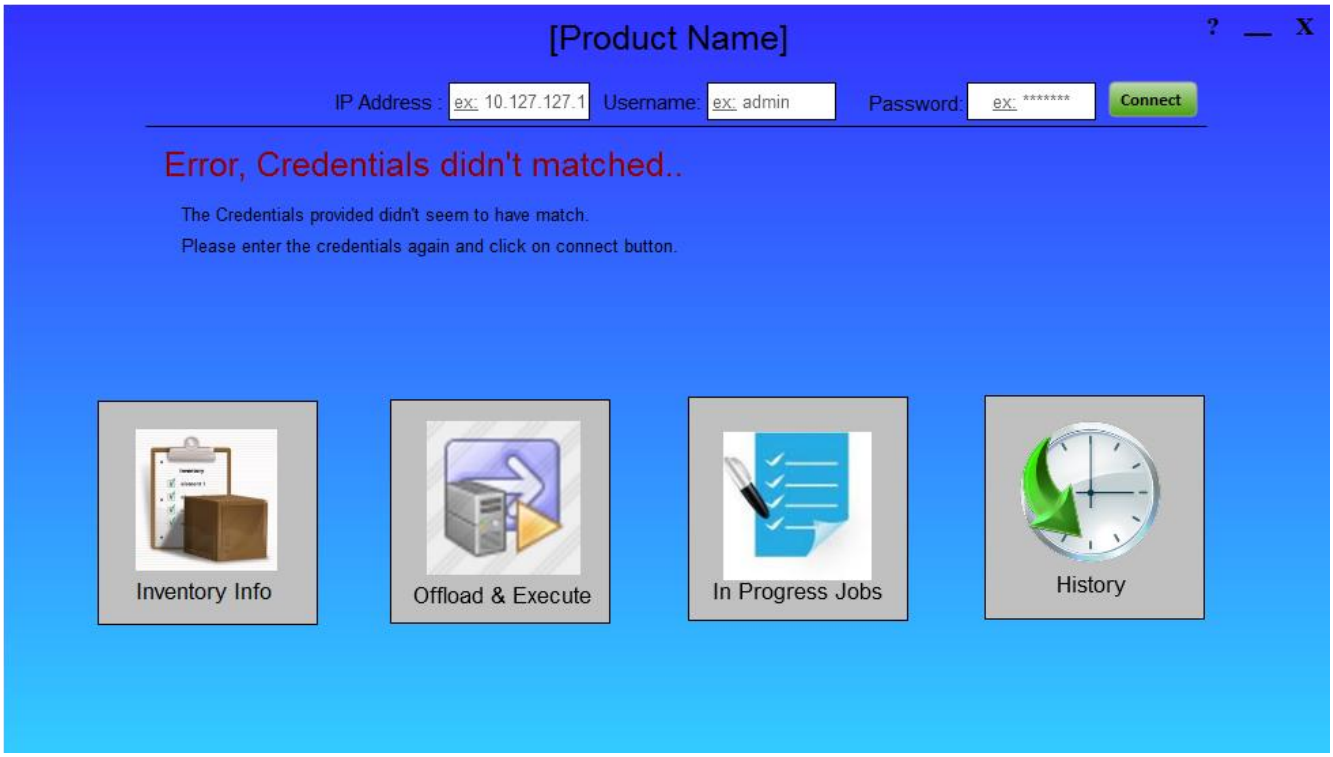
- **Login - Connection Failed Error :**

  This screen displays the error message, if the connection fails. The textboxes gets reset.



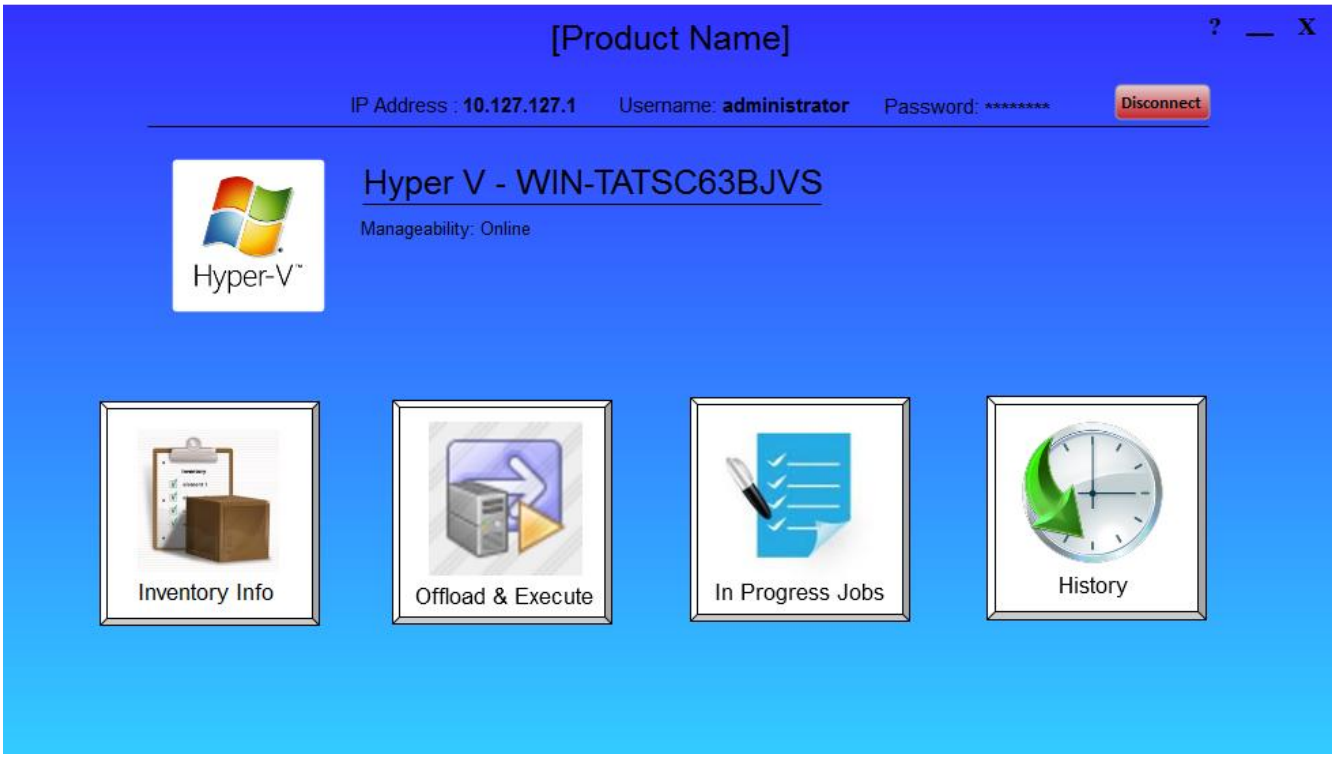
- **Login - Wrong Credentials Error :**

  This screen displays an error message, if the credentials are incorrect. The textboxes gets reset.



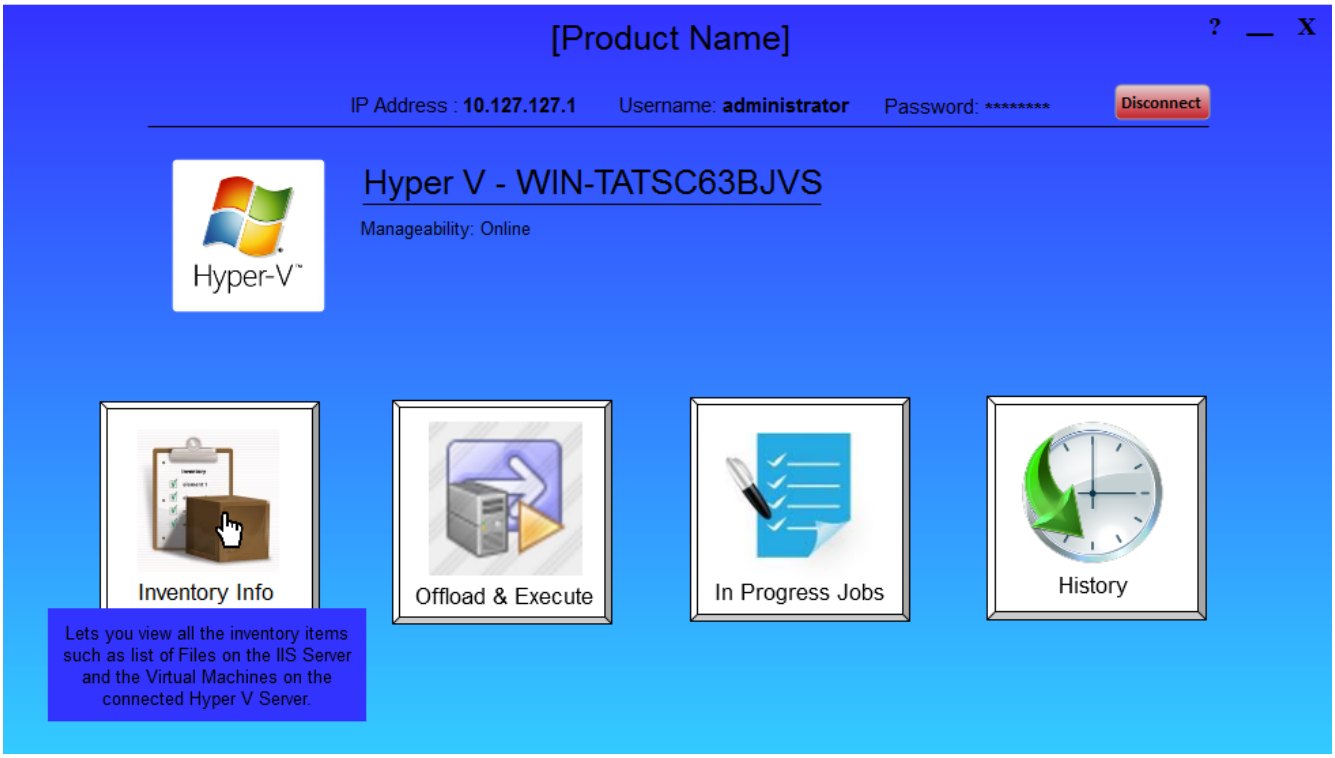
- **Login Successful - Landing Page :**

  This screen displays the HyperV name and related information after the authentication is successful. At the same time, the tabs gets enabled.



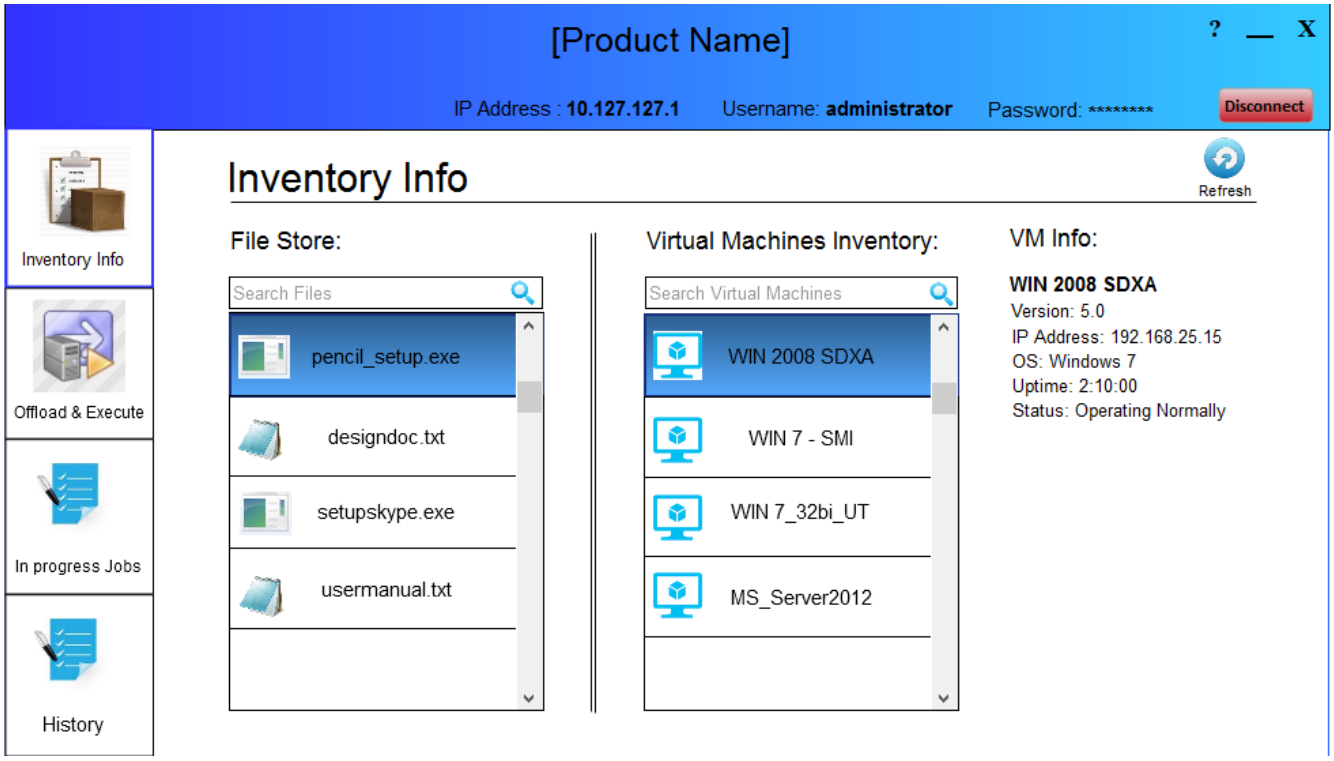
- **Landing Page - Mouse Hover on Tabs :**

  This screen shows the Mouse Hover effect on the tabs. The information related to the particular tab is displayed.



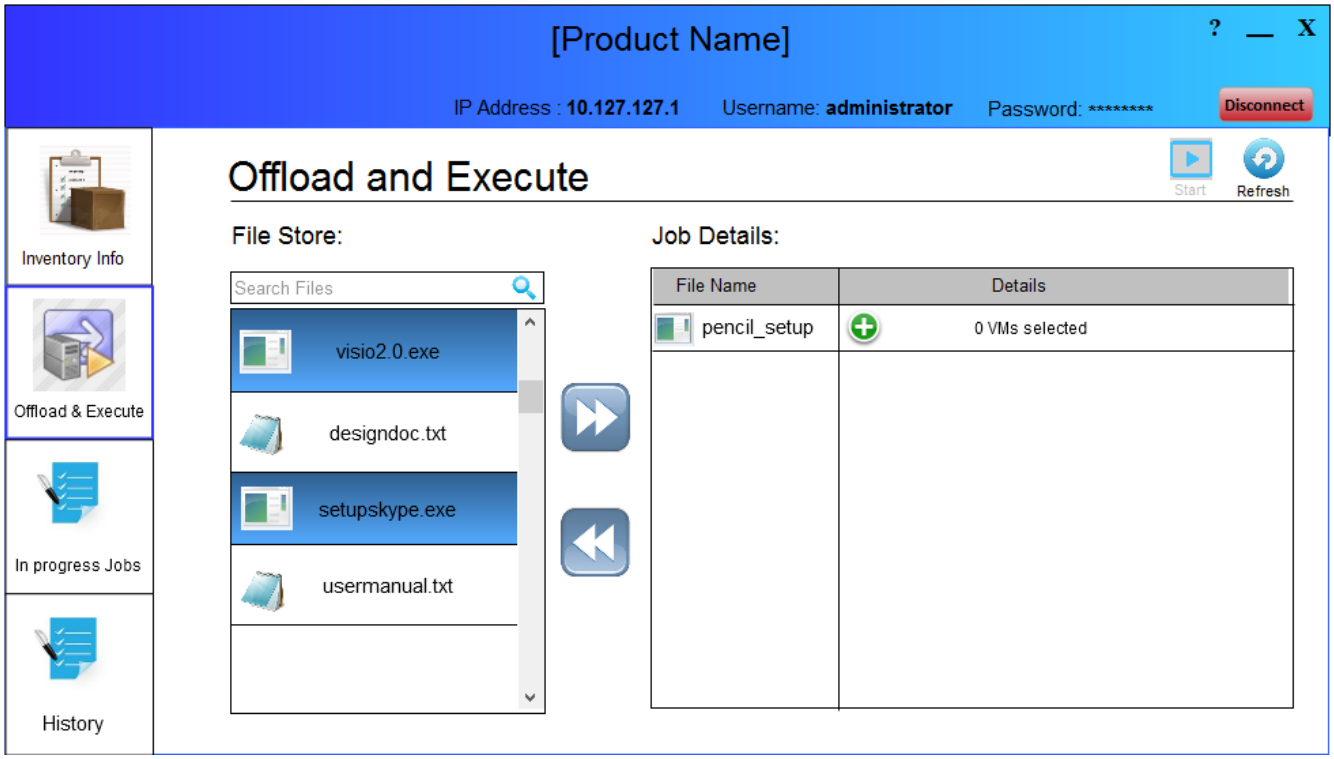
- **Inventory Info Page:**

  The Inventory info page displays the information regarding the Files Store and the VMs present on the connected HyperV. Also, the information regarding a particular VM is displayed on extreme right.

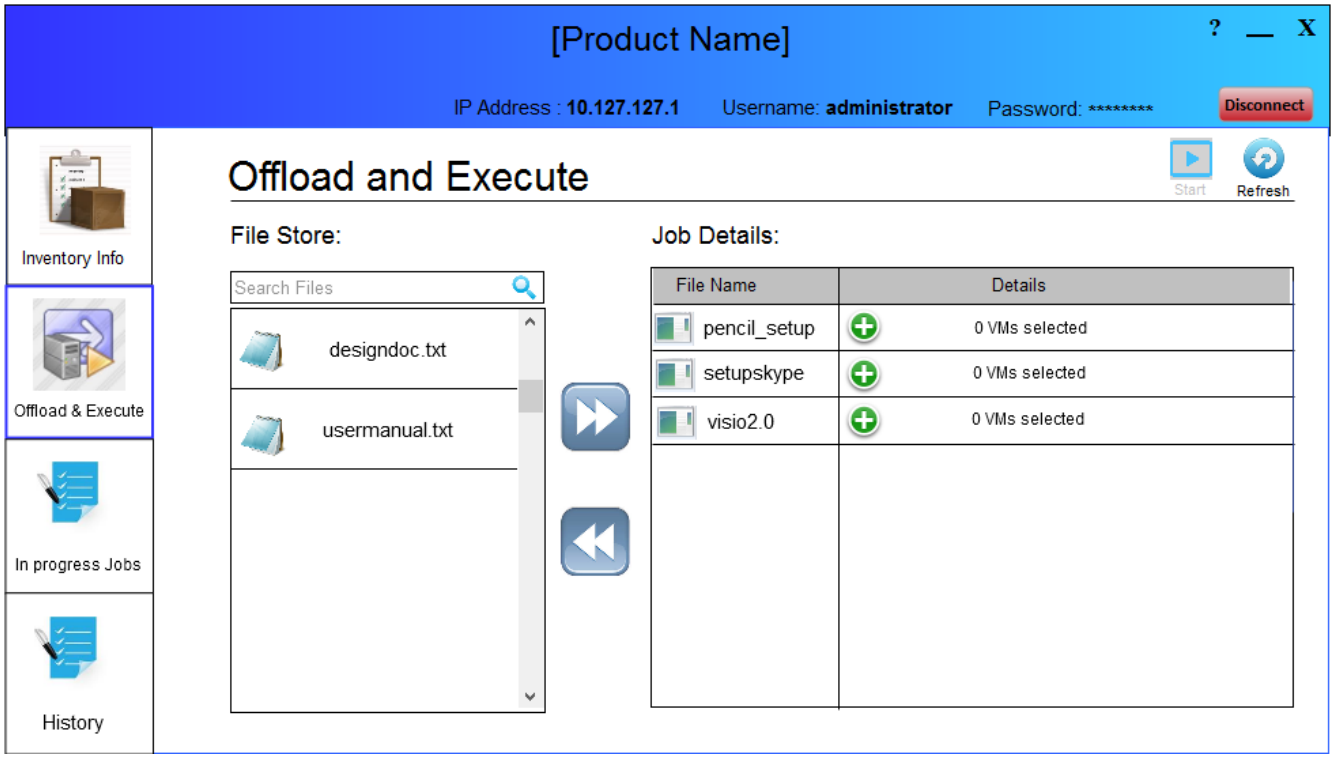


- **Offload and Execute Page :**

The Offload and Execute Page displays the files on the File Server. The files from the file store list can dragged and dropped in the Job details grid

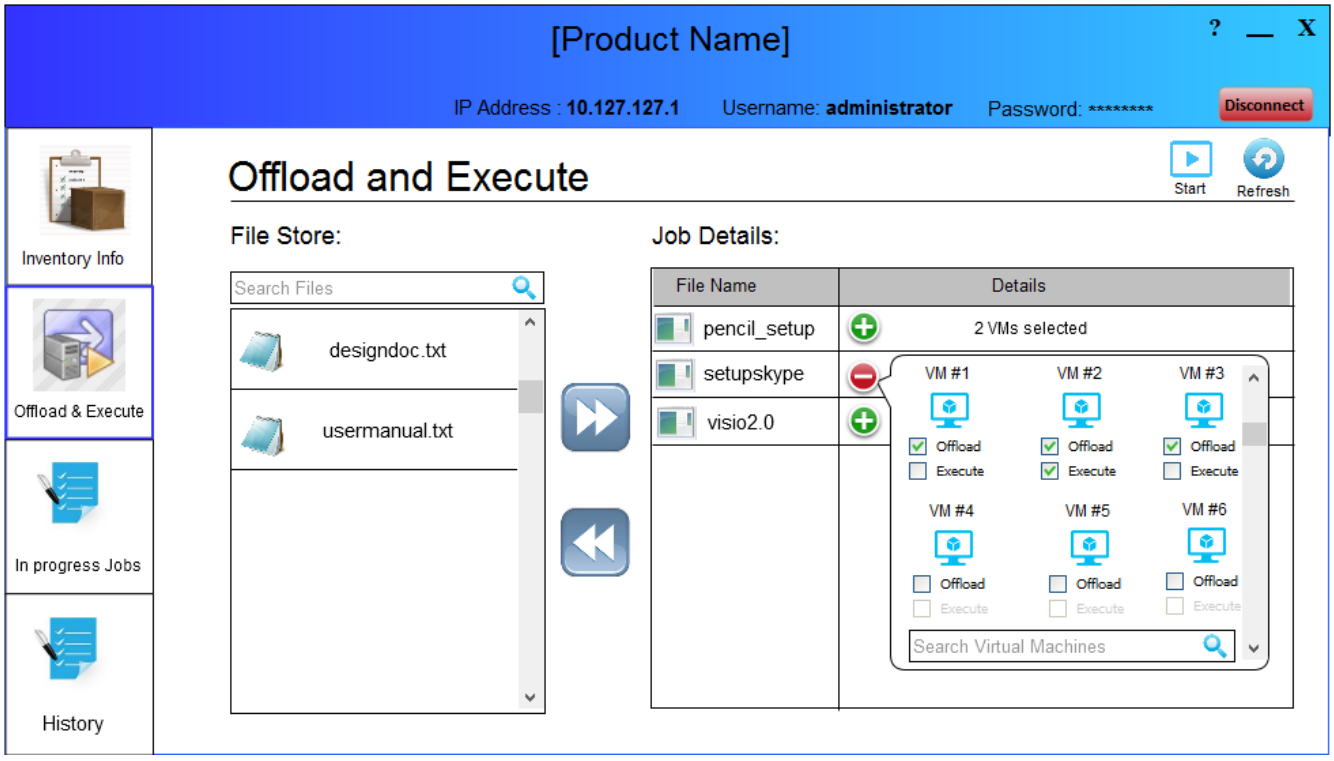


- **Offload and Execute Page - Adding more files:**
  More than one files can be added to the Job details grid.



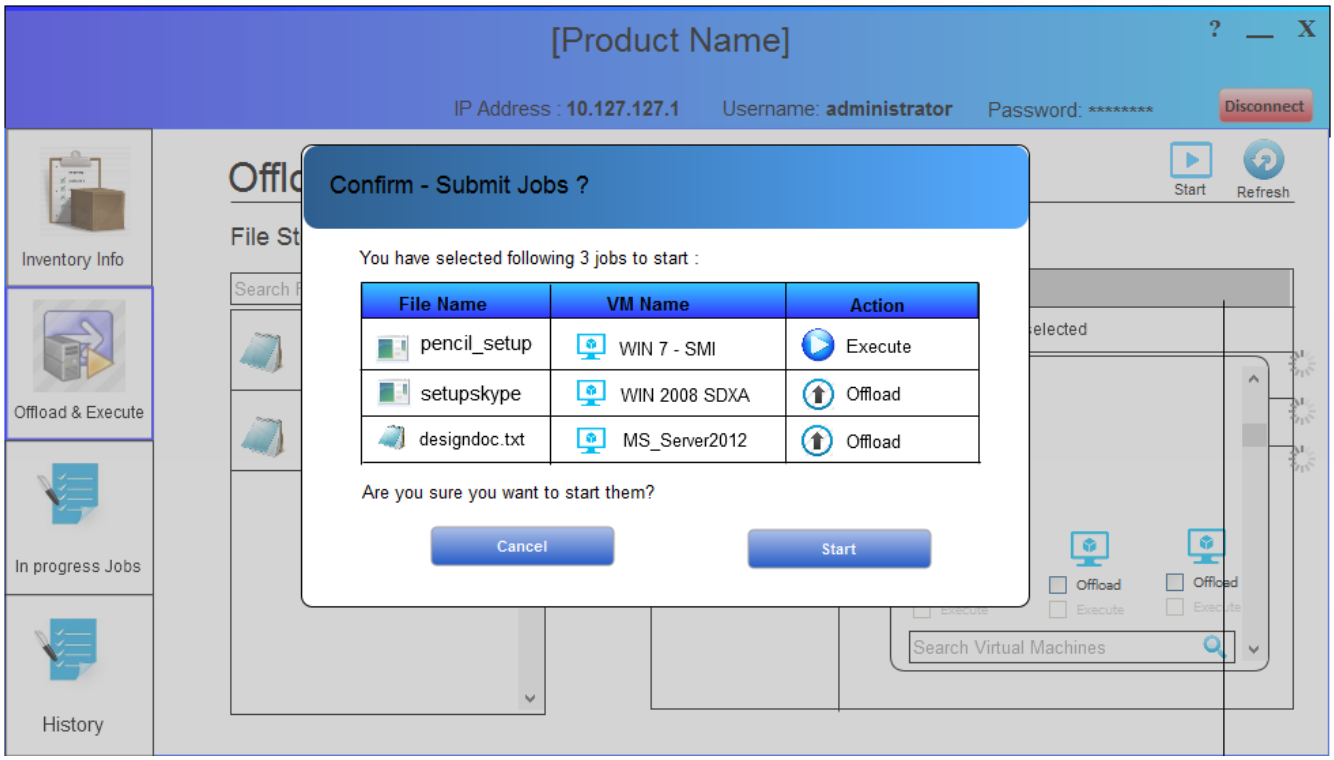
- **Offload and Execute Page - Selecting the VMs :**

Once the required files are added to the Job details grid, the plus button adjoining the file can be clicked to select VMs on which the particular file is to be offloaded and installed.



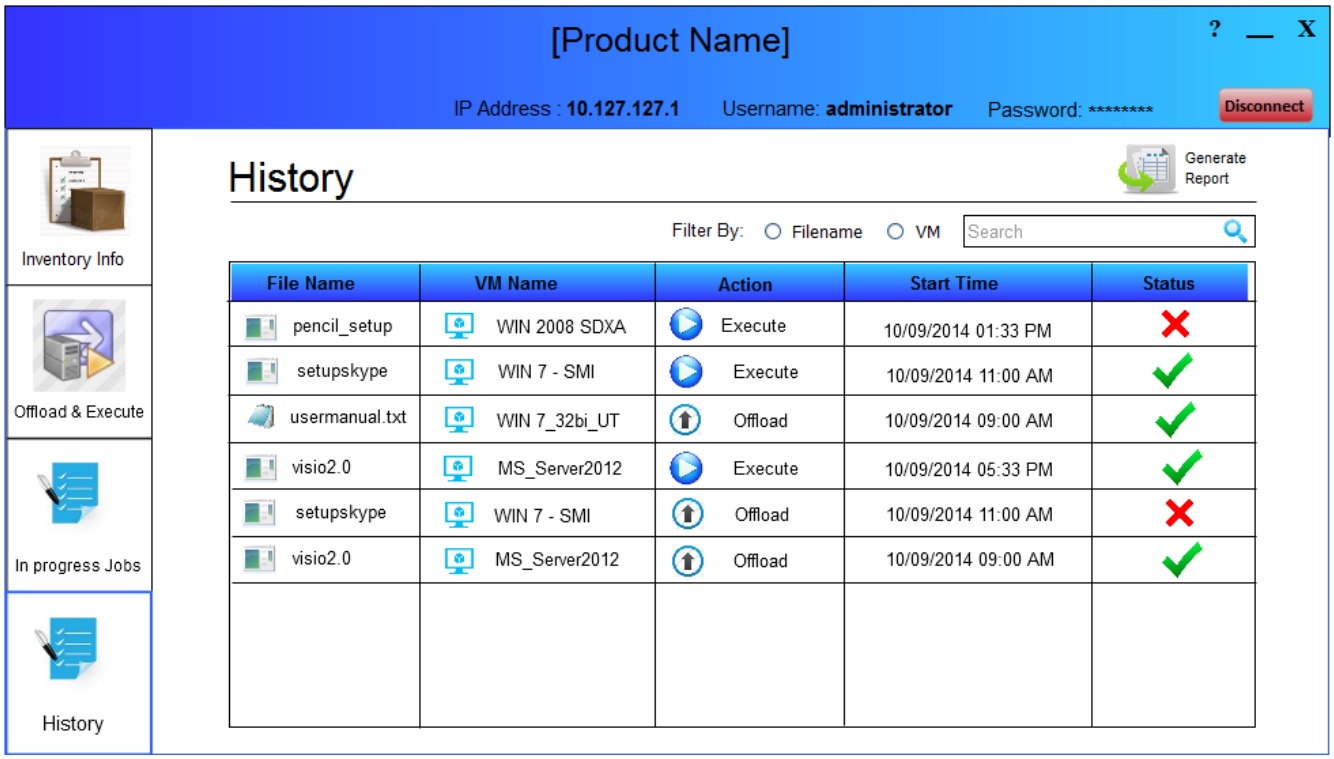
- **Submit Jobs - Confirmation :**

  Once the job details are specified and start button is clicked, this windows pops up for confirmation. Once the jobs are confirmed, the jobs are executed.

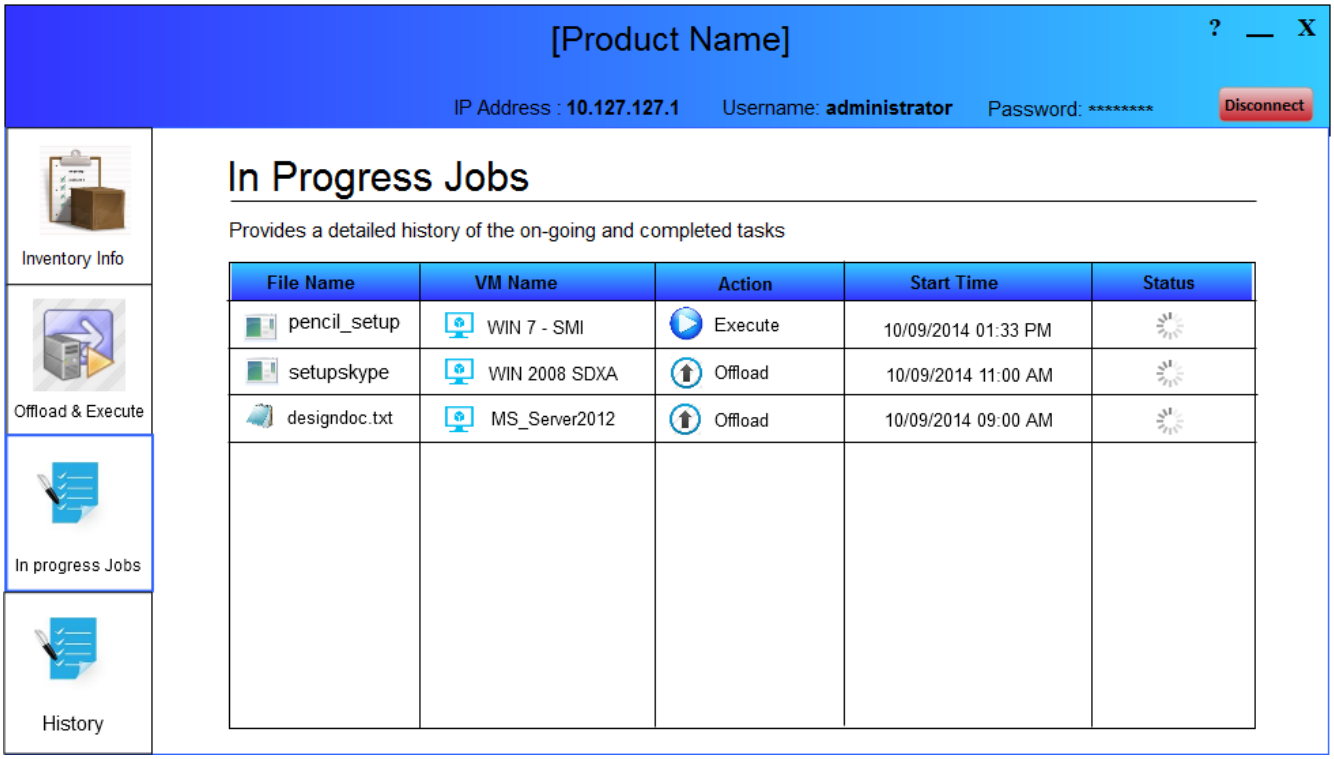


- **History :**

  The history screen displays all the completed jobs along with other details such as filename, VM name and the action to be performed. It also displays whether the job was successfully completed or not.



- **In Progress Jobs :**
  The In Progress Jobs screen displays the on going jobs.



7. REFERENCES

[1] "Experimental Study of Remote Job Submission and Execution on LRM through Grid Computing Mechanisms," Advanced Computing & Communication Technologies (ACCT), 2014 Fourth International Conference on , vol., no., pp.335,341, 8-9 Feb. 2014.

[2] www.technet.microsoft.com

[3] www.techblogs.com