**vmiDC Orchestration Agent Functional Specifications**

The purpose of this document is to provide high level overview of vmiDC orchestration agent. We have chosen this name, i.e. Orchestration Agent, to differentiate with other vmiDC agents such as Data Path Agent, etc. This is a first draft and will evolve as more information becomes available. Using term “agent” in this document implies “vmiDC Orchestration Agent” unless otherwise stated.

**Functionalities**:

We achieve the following functions by having our agent running on the vNSP appliance:

1. Extract necessary network/management information by parsing the NSX generated content
2. Configure networking parameters of the appliance to enable outgoing communications. We manage states to avoid redundant configuration.
3. Send the registration request along with required arguments to the vmiDC server. Among the arguments is the Virtual System ID, which indicates to vmiDC server the vCenter to which this distributed appliance instance belongs. vmiDC server will generate a unique, intuitive and immutable name and will then send a device add request to NSM. It will then send a response back to the agent which includes manager ip, device name and shared secret key.
4. Once the agent receives the response to its registration request from vmiDC server, it will authenticate the appliance with NSM by running the idscli provided commands. We, again, manage states to avoid redundant authentication
5. The agent is capable of responding to vmiDC server’s inquiries about its status.

Below diagram depicts steps to have a paired up vmiDC agent:

4: Add Device

Acknowledgement

vmiDC Agent

1: Configure agent’s networking parameters

vmiDC Server

2: Registration Request

5: Response

NSM

3: Add Device

6: Pair UP

**Appliance Registration(with vmiDC server)/Authentication(with NSM):**

As explained in the previous section of this document, our virtual appliance must register itself with the vmiDC server when it is up the first time. To begin this process we need to fill in the REST request data structure which contains the following fields:

*Appliance IP (extracted from mountetd XML)*

*Virtual System ID(extracted from mounted XML)*

*Appliance Name (which will be null the very first time)*

*vmiDC agent Version*

A REST call is used to communicate with the vmiDC server. Basic authentication on top of HTTPS (TLS based) is used to ensure security of our communication channel.

Upon receiving the registration request at the vmiDC server, a check is done against the database and the Appliance Name field of the coming request to see if it is the first time registration attempt. If so, a unique and immutable name will be generated. Also the server will contact the NSM to add the device after which (if successful) a response including the following data will be sent back to the registration request originator:

*Appliance Name*

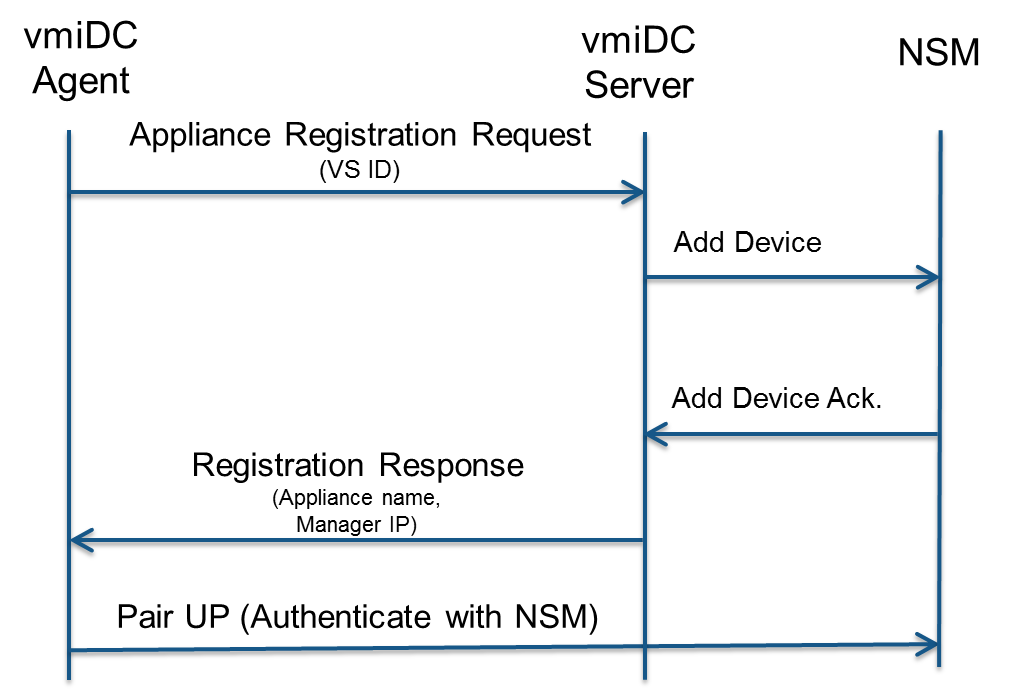
*Manager IP*

*Shared Secret Key*

Again the communication will be based on Basic Authentication and TLS.

At this point the appliance has all the necessary information to authenticate itself with the manager (NSM). This will happen only if a prior authentication is not valid anymore due to changes in some of the information (manager IP, etc.). A property file is used to manage states to avoid unnecessary authentications.

The following sequence diagram depicts the aforementioned steps to perform registration/authentication.



Note: the vmiDC agent has logic of re-registration, i.e. the whole registration process happens periodically (currently every 5 minutes) by our scheduler to accommodate any failure due to unexpected situations such as network outage, etc.

**REST APIs:**

The vmiDC Agent facilitates remote set and get to achieve some important tasks by exposing REST APIs. Below some of these APIs have been introduced:

1. Agent Upgrade: The main purpose of this API is to enable agent self-upgrade:

*Request: The URL from which the new jar file is to be downloaded from*

*Response: Upgrade status*

1. Triggering Registration: Bu exposing this API , we give the flexibility of on-demand registration triggering to our vmiDC server.
2. vmiDC Agent Status: To obtain the status of our agent running on an appliance, vmiDC server will use this API.

*Response: Version, current agent time, appliance IP, appliance name, manager IP, and pid (process ID of the running JVM)*

1. vmiDC Update IP: This API is needed to propagate the new IP address of the vmiDC server (in case of any change) to our agents:

*Request: The new IP address of the vmiDC server*

1. vmiDC Update Password: Similar to the IP change any password change needs to be propagated to our agents as well or else any future communication from vmiDC agent to vmiDC server will fail due to invalid credentials*:*

*Request: The new IP address of the vmiDC server*

**Security Specifications:**

All communications between vmiDC agents and our vmiDC server occur through a secure HTTPS channel which is based on Transport Layer Security Protocol. Also for authentication, the BASIC authentication approach is used on top of this secure channel.

Another aspect of our security involves confidentiality of our sensitive data such as passwords and Shared Secret Key which is ensured by exploiting an Encryption/Decryption algorithm. We actually use NSM encryption/decryption tool which is based on Md5 digest with “DESede/CBC/PKCS5Padding” cipher.

We do not just ensure encrypted password/SSK on the wire but also for the persistence of any of these data. i.e. if needed they are only stored in files and our database encrypted.