Software Requirements Specification

For <<Device Management >>

Version 1.0

Votary Softech Solutions Pvt. Ltd.

Plot No: 76, Lumbini layout,   
Near Euro school,   
Gachibowli-I (V), Hyderabad,  
Telangana - 500032,   
India.

**Revision History**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Version (x.y) | Date of Revision | Description of Change | Reason for Change | Affected Sections | Approved By |
| 1.0 | 25/04/2017 | Initial Draft | New Document | All |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Approval History**

|  |  |  |  |
| --- | --- | --- | --- |
| Version (x.y) | Prepared By | Reviewed By/Date | Approved By/Date |
| 1.0 | Mohan and team | Initial Draft | New Document |
|  |  |  |  |
|  |  |  |  |

**Contents**

[1 Define 4](#_Toc481142930)

[1.1 Overview 4](#_Toc481142931)

[1.1.1 Project Functions: 4](#_Toc481142932)

[1.2 Scope 5](#_Toc481142933)

[1.3 Assumptions 6](#_Toc481142934)

[1.4 Limitations 6](#_Toc481142935)

[1.5 Glossary 6](#_Toc481142936)

[3 Specify 7](#_Toc481142938)

[3.1 Design Approach 7](#_Toc481142939)

[3.2 System Features 8](#_Toc481142940)

[3.2.1 Base features 8](#_Toc481142941)

[3.2.2 Service Features 9](#_Toc481142942)

[3.2.2.1 Control Management 9](#_Toc481142943)

[3.2.2.2 Notification Management 9](#_Toc481142944)

[3.2.3 Target Features 9](#_Toc481142945)

[3.2.4 <Initialization> 9](#_Toc481142946)

[3.2.5 <Discover > 9](#_Toc481142947)

[3.2.6 <Get > 9](#_Toc481142948)

[3.2.7 <Put > 9](#_Toc481142949)

[3.2.8 <Delete > 9](#_Toc481142950)

[3.2.9 <Observe> 9](#_Toc481142951)

[3.3 Business Attributes 9](#_Toc481142952)

[3.4 Software Quality Attributes 9](#_Toc481142953)

[3.5 Other Requirements 10](#_Toc481142954)

[3.6 User Documentation and Training 10](#_Toc481142955)

[4 Architecture 10](#_Toc481142956)

[4.1 Block Diagrams 10](#_Toc481142957)

[4.2 Operating Environment 10](#_Toc481142958)

[5 Design 10](#_Toc481142959)

[5.1 Design Approach 10](#_Toc481142960)

[5.1.1 Introduction 10](#_Toc481142961)

[5.1.2 Internal Interfaces 11](#_Toc481142962)

[5.1.3 External Interfaces 12](#_Toc481142963)

[5.2 Implementation Constraints 12](#_Toc481142964)

[6 Implementation 12](#_Toc481142965)

[6.1 Functional Requirements 12](#_Toc481142966)

[6.2 Nonfunctional Requirements 16](#_Toc481142967)

[6.2.1 Performance Requirements 16](#_Toc481142968)

[6.2.2 Safety Requirements 16](#_Toc481142969)

[6.2.3 Security Requirements 17](#_Toc481142970)

[7 Validate 17](#_Toc481142971)

[8 Deploy 17](#_Toc481142972)

[9 Maintain 17](#_Toc481142973)

# Define

VITA Device management is IoT gateway firmware which enables application to discovery and engage IoT devices. DM is hardware & connectivity agnostic. This is an independent and pluggable module into any solutions.

This is part of the VITA framework, Device Management is current scope of the document.



## Overview

### Project Functions:

The current DM is developed in three phases. The functions of each version defines respective scope.

We have initially identified third party libraries to suite our VITA stack, as our main purpose is to support BLE, IP, WIFI stack.

Then wrote APIs to implement VITA DM requirements, which were developed based on IOTivity devices.

Then wrote APIs to implement non-IOTivity devices also.

The following are API’s which internal functions to achieve the frame work capabilities are.



### Why IOTivity?:

We have zeroed on the IOTvity framework after studying industry

* IoT market leaders like Intel, Samsung, Cisco defined spec and implemented.
* It supports “RestFull Architecture”, “abstracts transport layer”.
* It Supported BT,BLE,IP Transport layers

## Scope

The following is scope of work, we have done so far.

DM module is capable of handling IOTivity and non-IOTivity devices

IOTivity devices follows complete stack of IOTivity (It follows CoAP and UDP, Iotivity base [CSDK] )

**Phase 1 :**

In phase1 VITA DM supports Linux on Raspberry and X86

* **Schedule 1:** Third party lib identification and integration
  1. In order implement “identification and operation” devices, we have to utilize third part libraries. This is will help us to enhance our developmental lifecycle.
  2. We have chosen “IOTvity version 1.2 ” as baseline and third party library. This are well suited for our VITA architecture. This baseline was required, as it supports BLE, IP , WIFI based devices detection (Wifi is not in current phase)
* **Schedule 2**: Discovery, Connect, Engage –BLE devices
  1. We have implemented the discovery, Connect, Engage
  2. Here , implementation is based on IOTivity stack defined in **Schedule 1**
* **Schedule3**: Discovery, Connect, Engage –Third party stack BLE/IP devices
  1. We have implemented the discovery, Connect, Engage
  2. Here , implementation is based on IOTivity an non IOTvity

**Phase 2 : Android support**

Android is not in present scope

## Assumptions

* DM APIs are called through SA with help of TL (Transport Layer)
* DM pushes data to SM using TL (Transport Layer) APIs
* DM is responsible for detecting different devices (BLE , IP, Wifi). DM converts this device data to JSON format. JSON format is default data in VITA DM.

## Limitations

* DM can handle one application at a time.
* BLE range is 30 feet
* Wifi range is 60 meters
* Security feature is not implemented in Phase1
* Current phase does not support “Android”
* Current phases does not support “DeleteResource”

## Glossary

| Term | Definition |
| --- | --- |
| VITA | Votarytech IOT Architecture |
| BLE | Blue-tooth Low energy devices (Ex: Beacons) |
| TCP/IP | Transmission Control protocol |
| IP | Internet protocol |
| DM | Data Management |
| OCF | Open Connectivity Forum |
| GPS | Global positioning system |
| RFID | Radio Frequency Identification |
| Zigbee | ZigBee is a specification for a suite of high level communication protocols used to create personal area networks built from small, low-power digital radios. |
| NFC | Near field communication |
| IoTivity | Open source project implemented based on OCF specification |
| QT | Tool to write/Debug the code |
| GCC | Compiler to build C & Cpp files |
| SCON | Software Construction tool |
| RaspberryPI (ARM Arch) | It is a small single board computer which have ARM processor |
| Desktop (x86 ) | Any computer which have x86 processor |
| TL | Transport Layer |

# 

# Specify

This document is the DM SRS document which specifies

* The Framework
* Core architecture and Interfaces to SA, SL, SM. This is shown in below diagram



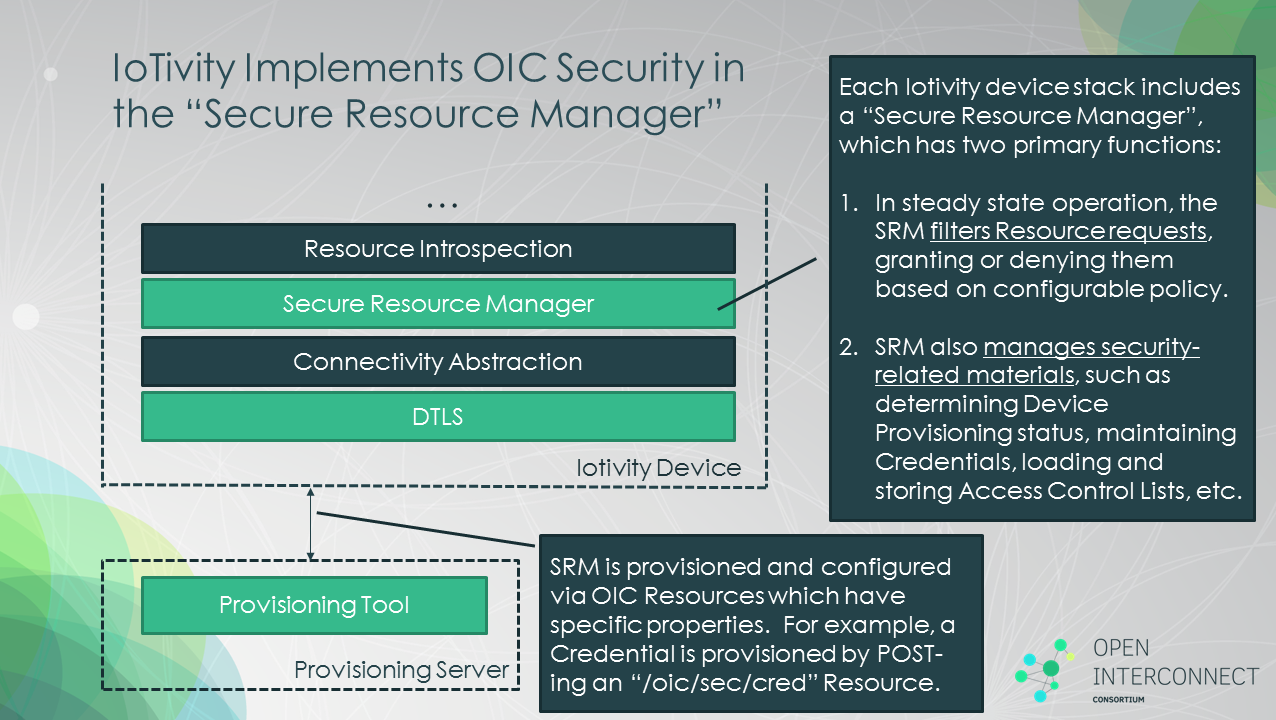
## Design Approach

The architecture is based on the Resource Oriented Architecture design principles. The architecture enables resource based interactions among IoT devices independent of how they connected (BLE,WiFi,TCP/IP,etc)

The architecture leverages existing industry standards, technologies and open source projects and provides solutions for establishing wired/wireless connections. Architecture manages the flow of information among IoT devices, regardless of hardware platform, operating systems, Transport layer on which they connected.

Key features:

* + - Based on RESTFull architecture (Simple Interfaces Discover/Get/ Delete API ( Delete API is work in progress)
    - A connectivity and interoperability for major domains including Health, Consumer, Enterprise, Industrial, Automotive
    - Common communication mechanism for discovery and connectivity
    - Security ( Current version does not support ]
    - A scalability
      * Scalable means , we can do modularize and support independent activities [ Wifi only, IP Only, BLE only, IP, BLE, Wifi permutation and combinations or all inclusive ]
      * Security ( Current version does not support ]
      * Security can be elaborated as follows



## System Features

System features are classified into Base, Service and Target.

In summary, the “System APIs” are derived from this features. They are described as below.

### Base features

Discovery: Discovery of the devices in the network and connect to the devices.

### Service Features

#### Control Management

* **Read/Get** from devices
* **Delete** the device
* **Write/Put** to the devices

#### Notification Management

* **Observe** the device

### Target Features

Linux and Android

### <Initialization>

In initialization process client is registering to the network.

### <Discover >

In discovery process, all the resources connected to the network is identified and database is maintained.

### <Get >

In this process, we get the state of particular “resource”.

### <Put >

In this process, we get the state of particular “resource”.

### <Delete >

In this process, user to delete the resource listed by the discovery API.

### <Observe>

In this process, user to identify the change in the resource state

## Business Attributes

|  |  |
| --- | --- |
| Schedule | 6 weeks |
| Effort | 5 Engineers + 1 Manager |

## Software Quality Attributes

Services: Resource API – Interface between Project and user.

VITA– DM is pluggable

VITA - DM supports Restful architecture

VITA – DM connects via “Transport Layer”

VITA – DM supports any type “transport layer “

VITA – DM can be configurable from remote areas.

## Other Requirements

|  |  |
| --- | --- |
| Database Requirements | Not Applicable |
| Internationalization requirements | Not applicable |
| Legal requirements | Not applicable |
| Reuse objectives for the project, | Provides a development standard for VotaryTech.  Every VT project can make use of VITA – DM framework to create a new product line |

## User Documentation and Training

1. Developer Guide - A comprehensive manual on how to use APIs provided by the framework.
2. Training Sessions – Training will be given to client based on the distribution license and pricing.

# Architecture

## Block Diagrams

Mentioned in 5.11 section

## Operating Environment

Preferable OS – Ubuntu 14x.

This is the most widely used Linux OS for any IOT product.

Android is future support

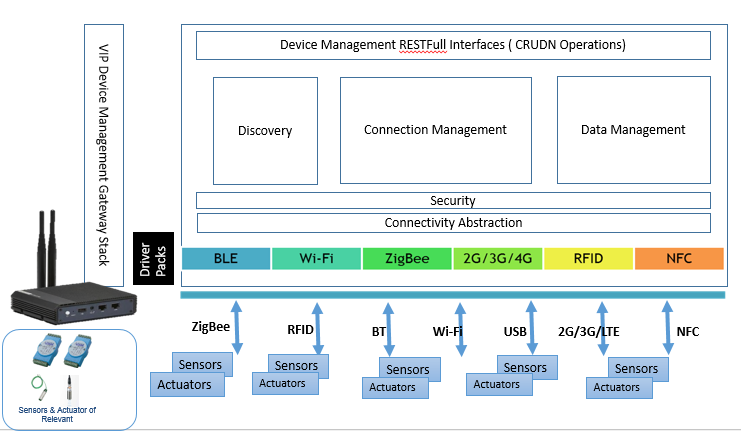
# Design

## Design Approach

### Introduction

The architecture is based on the Resource Oriented Architecture design principles. The architecture enables resource based interactions among IoT devices independent of how they connected (BLE,WiFi,TCP/IP,etc)

The architecture leverages existing industry standards, technologies and open source projects and provides solutions for establishing wired/wireless connections. Architecture manages the flow of information among IoT devices, regardless of hardware platform, operating systems, Transport layer on which they connected.



* Three Independent vertical design as the diagram
  + Detection Managements
  + Connection Management
  + Data Source Management
* Interoperable
* Easily configurable
* Easily pluggable other solutions
* Reliable and Scalable
* Highly secure
* Designed to support a wide variety of IoT solutions across industries/domains.

Note

* In above diagram “Security is not implemented: in current phase
* In above diagram, Data management module stores the data in DM: In current phase. In future phase, DM should able to send data to SM or other devices

### Internal Interfaces

**Detection Management:**

* **Initiate:** Initialization of the Device Management module
* **Discovery:** The job of this API is to find the resources available on the network and given to the user as a list.

**Connection Management:**

* **Read:** The job of this API is read action at resource.
* Write**:** The job of this API is write action at resource.
* **Observe:** The job of this API is observe the change of action at resource.
* **Delete Resource:** The job of this API is delete resource from DM module.
* **Data Send/Send Data:** The job of this API is send data to resource from DM module.

This specifies the Framework and core architecture and Interfaces to external world [external interface mentioned in 5.1.3]

**Requirements**

* Initialization
* Discover
* Get
* Put
* Delete
* Observe

### External Interfaces

* DM may get the configuration logic from SL (Smart logic)
* DM will send the data to SM ( Storage management)
* DM will get the commands from SA (Smart App)

## Implementation Constraints

If the customer chooses to deploy the product on VotaryTech Environment, maintenance cost will be borne by the customer in accordance to the distribution policies framed by the Sales team. VITA – DM product as a standalone framework. If the client wants the product to be installed on VotaryTech Environment, the cost of maintaining the server instances will be billed to the client.

# Implementation

<Implementation details of the Project. References to Coding Standards. Review guidelines. Supplement with Resource and procurement requests and Development Test environment setup and collaboration tools. >

TBD

## Functional Requirements

< Itemize the detailed functional requirements associated with this feature. These are the software capabilities that must be present in order for the user to carry out the services provided by the feature, or to execute the use case. Include how the product should respond to anticipated error conditions or invalid inputs. Requirements should be concise, complete, unambiguous, verifiable, and necessary. Use “TBD” as a placeholder to indicate when necessary information is not yet available.

Each requirement should be uniquely identified with a sequence number or a meaningful tag. The following is a typical identification method for requirements. >

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SL # | Requirements | Description | Identification Method | Identification method or API |
| 1 | REQ – 001 | DM shall be transport independent | DM should abstract Transport Technology  (How IoT device connected ) to calling Application | None , generic feature |
| 2 | REQ – 002 | DM shall be OS independent | 1. DM should run on Linux 2. DM should run on Android (Phase 2) | None , generic feature  (Note: Current version support only Linux . Android in next phase ) |
| 3 | REQ – 003 | DM shall be hardware agnostic | 1. *DM should run on RaspberryPI (ARM Arch)* 2. *DM should run on Desktop (x86 )* 3. *DM should run on Android Mobile* | None , generic feature  (Note: Current version support Raspberry and X86 . Android in next phase) |
| 4 | REQ – 004 | Supporting simultaneous access from multiple clients not allowed |  | None , generic feature  (Note: How do verify this ) |
| 5 | REQ – 005 | Communication between Application to DM should be secure |  | None , generic feature  (Currently security feature is not implemented in current phase ) |
| 6 | REQ – 006 | Communication between DM and IoT devices should be secure |  | None , generic feature  (Currently security feature is not implemented in current phase ) |
| 7 | REQ – 007 | DM shall provide RestFull APIs | Discover, Get, Put, Observe, Delete | DM\_Discovery, DM\_Get, DM\_Put, DM\_observe, DM\_DeleteResource |
| 8 | REQ – 008 | DM shall communicate/engage BLE-IoT devices | Discover, Get, Put, Observe, Delete | DM\_Discovery, DM\_Get, DM\_Put, DM\_observe, DM\_DeleteResource |
| 9 | REQ – 009 | DM shall communicate/engage **IoTivity** BLE-IoT devices | Discover, Get, Put, Observe, Delete | DM\_Discovery, DM\_Get, DM\_Put, DM\_observe, DM\_DeleteResource |
| 10 | REQ – 010 | DM shall communicate/engage IP (Ethernet & WiFi) **IoTivity** Devices | Discover, Get, Put, Observe, Delete | DM\_Discovery, DM\_Get, DM\_Put, DM\_observe, DM\_DeleteResource |
| 11 | REQ – 011 | DM shall take all configuration parameters from application | 1. *VITA Filter information* 2. *SM Server IP* | This needs to be elaborated , on configuration of parameters us |
| 12 | REQ – 012 | DM shall be able to Discover IoT devices around it | 1. *DM shall be able to Discover BLE-IoT devices within 30Feet* 2. *DM shall be able to Discover IoTivity-IP devices* 3. *DM shall be able to Discover IoTivity-BLE devices within 30Feet* | DM\_Discovery |
| 13 | REQ – 013 | DM shall be able to provide interface to Read device Attributes in key-value pair |  | DM\_Get |
| 14 | REQ – 014 | DM shall be able to provide interface to Write device Attributes in key-value pair |  | DM\_Put |
| 15 | REQ – 015 | DM shall be able to provide interface to Observe any/few Attributes |  | DM\_observe |
| 16 | REQ – 016 | DM shall out put its results to SM | 1. Discovery Result to be stored in SM 2. Get Result to be stored in SM 3. Observe Result to be Stored in SM | DM\_SentData ?? ( it is part of TL, butnot part DM) |
| 17 | REQ – 017 | If Application running remotely, in this case its communicates to DM through TL-MQTT |  | No APi will be available, it is generic functionality |
| 18 | REQ – 018 | Application and DM may be running in same System (Hardware), in this case its directly call DM APIs through TL |  | No APi will be available, it is generic functionality |
| 19 | REQ – 019 | DM shall be able to monitor/observe (notify if any change in device attributes change) IoT device | 1. DM shall be able to monitor continuously to any IoT device 2. DM shall provide interface to cancel observe 3. DM shall monitor multiple IoT devices 4. DM shall monitor multiple attributes in IoT Device | DM\_observe |
| 20 | REQ – 020 | DM shall be modular | Discovery alone shall be built independently | This is not clear |

## Nonfunctional Requirements

### Performance Requirements

< If there are performance requirements for the product under various circumstances, state them here and explain their rationale, to help the developers understand the intent and make suitable design choices. Specify the timing relationships for real time systems. Make such requirements as specific as possible. You may need to state performance requirements for individual functional requirements or features.

Ex: Response time for a transaction (average, maximum)

Throughput, for example, transactions per second

Capacity, for example, the number of customers or transactions the system can accommodate >

### Safety Requirements

< Specify those requirements that are concerned with possible loss, damage, or harm that could result from the use of the product. Define any safeguards or actions that must be taken, as well as actions that must be prevented. >

### Security Requirements

<Specify any requirements regarding security or privacy issues surrounding use of the product or protection of the data used or created by the product. Define any user identity authentication requirements. >

# Validate

<References to Checklists, VGates Tracker for SRS, Review defect logs and updates in Project Master List of Deliverables, Approval Emails, Traceability with Use Case Document >

# Deploy

<Commits to SVN of Baselined SRS after approval. Share the approved SRS with team and stakeholders. You can add references to SVN and document and emails.>

# Maintain

<Changes to Plan with appropriate references>