ST Electronics (Info-Software Systems) Pte Ltd

(*Regn No: 198601030N*)

Tech Factor Challenge 2016
IoT Platform
Usage Guidelines

This document contains a total of 17 Pages.

ST Electronics
Info-Software Systems
A member of ST Engineering

AMENDMENTS RECORD

Revision No.	Affected Page(s)	Details of Change (if necessary)	CP/DCR No. (where applicable)	Date of Implementation
1.0	All	New Creation	-	08-04-2016

TABLE OF CONTENTS

1. Introduction

1.1 Purpose

For the Tech Factor Challenge 2016, participants are required to design and develop:

- Equipment or robotics application ideas to improve mobility for the elderly in the context of a HDB home and build a prototype to demonstrate the idea. Solutions are to be non-invasive; or
- Software accompanying a hardware prototype for use in a smart-safe home or for health & wellness management applications. Solutions under the Smart H2 Challenge require both hardware and software (including apps) components.

To facilitate the competition, ST Electronics will be providing a Healthcare IoT cloud platform for participants who would like to upload their device and application data to the cloud, provide access using web apps and interact with mobile apps. Participants are strongly encouraged to leverage on ST Electronics' Healthcare IoT platform. This document describes the IoT Platform to be provided.

1.2 Scope

This document is applicable to the Tech Factor Challenge 2016 project for the development of:

- a common software system for the participants to use to integrate with their devices

2. Logical View

2.1 System Architecture

Figure 1: Architecture Overview shows an overview of the system architecture.
There are 3 main groups to the system:

1) **Device Group** – provided by participants
2) **Client Group** – provided by participants
3) **ST Electronics Healthcare IoT Platform (Here-on referred to as mediCAP)** – provided by ST Electronics

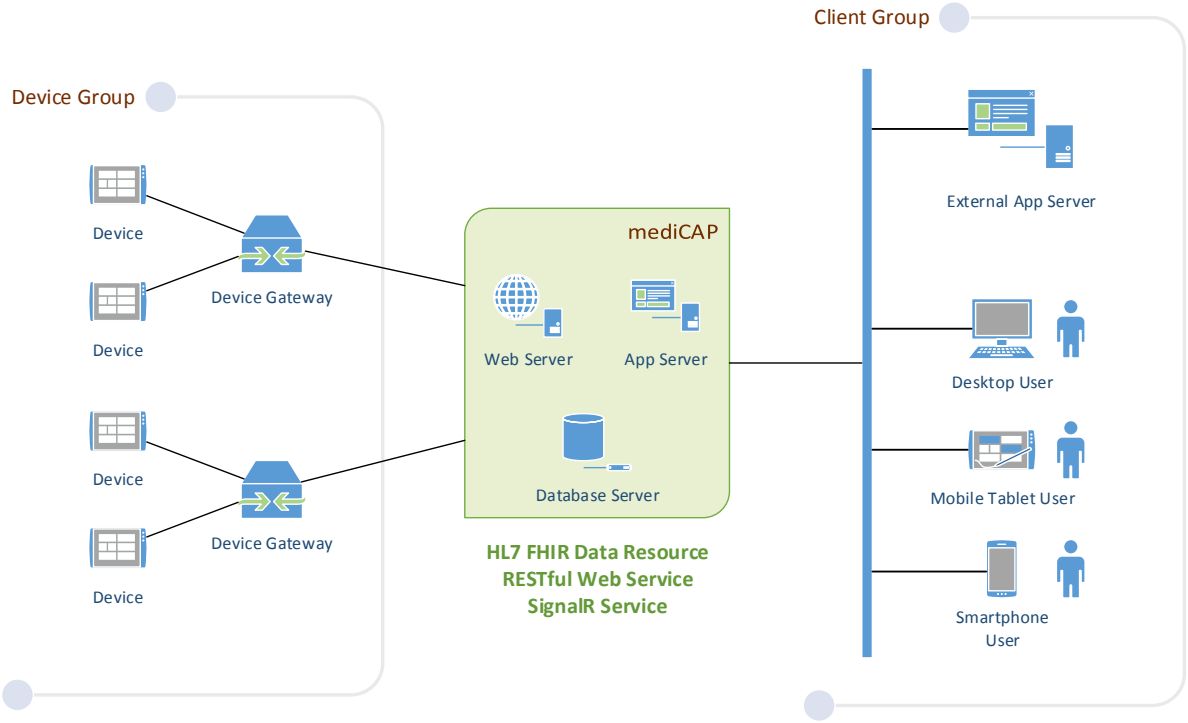


Figure 1: Architecture Overview

2.2 Device Group

The **Device Group** is one of the main focus areas for each participant of the TFC 2016 Challenge. This group consists of devices that are used for collecting health related data from patients. This data is usually collected in a home environment instead of a hospital. The collected data may consist of one or more of the following measurements, which are listed together with the respective devices that could collect them as below.

	Device	Measurement
1	Pulse Oximeter	Pulse, Oxygen Saturation
2	1-lead and 3-lead ECG recorders	1-lead and 3-lead ECG
3	Blood Pressure Monitor	Blood Pressure
4	Thermometer	Temperature
5	Weighing Scale	Weight
6	Glucose Meter	Blood Glucose Level
7	Body Composition Analyser	Body Composition
8	Peak Flow Meter	Peak Flow
9	Cardiac Output Analyser	Cardiac Output
10	Strength Analyzer	Strength
11	PT/INR Assay	INR (blood coagulation)

12	Cardio and Physical Activity Monitor	Cardio and Physical Activity
13	Insulin Pump Monitor	Glucose Levels
14	Respiration Rate Monitor	Respiration Rate
15	Urine Analyzer	Urine Analysis
16	Sleep Quality Monitor	Sleep Quality
17	Sleep Anpnoea Detector	Sleep Apnoea Breathing

The **Device Gateway** is a device or client that is used to transfer the data from one or more devices to the ST mediCAP System. The gateway OS and hardware could consist of any of the following:

- Android/IOS device
- Linux/Windows/Mac client

Each device communicates directly with the gateway using communication protocols (Bluetooth, ZigBee or Wifi). The data from the device is sent to the cloud server for storage and analysis. A cloud server will be provided for participants who do not have their own application server.

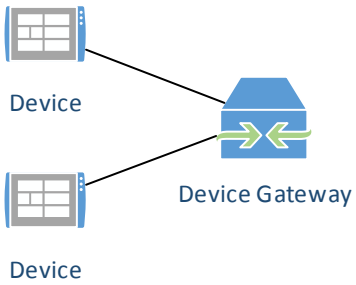


Figure 2: Device to Device Gateway

The participant shall implement their own **Device Gateway** if needed.
The **Device Gateway** must be capable of interacting with the **mediCAP** via HTTP RESTful Web Service.
If there is no **Device Gateway**, the Device itself must be capable of interacting with the **mediCAP** via HTTP RESTful Web Service.
(Refer to the section on Interface View for details)

2.3 Client Group

The **Client Group** is to be provided by the participant.
The **Client Group** consists of end user systems such as desktop/laptop, mobile tablet and smartphone etc.
It may also include a separate app server from the participant.

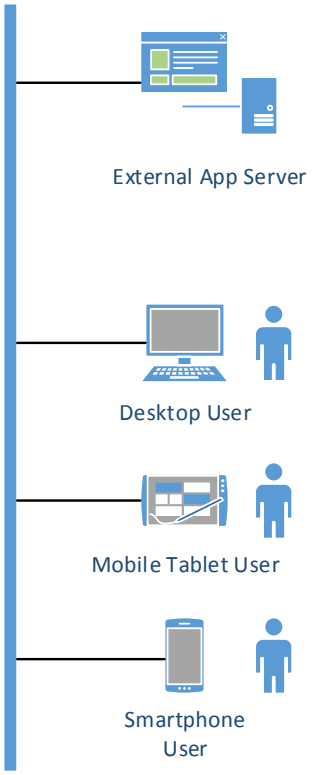


Figure 3: Client Systems

If the participant is using the **mediCAP** as the cloud server, their client systems must be capable of interacting with the server via HTTP RESTful Web Service.
If push messages are required, the client systems must support SignalR system. SignalR is the push messaging system that is supported by **mediCAP**.
(Refer to the section on Interface View for details)

2.4 ST Electronics' Healthcare IoT Platform - mediCAP System

The **mediCAP** is provided by ST Electronics.
It shall be a ready system for which the participants can optionally use.
It uses several common standard technologies so that there can be a unified communication between the devices and system as well as between the devices themselves.
(Refer to the section on Technologies for details on the technologies and standards that **mediCAP** uses)

3. Technologies

	What It Is	Used For
RESTful Web Service	Representational State Transfer Web Service	Communications between Device Group and Client Group to mediCAP
ASP.NET Razor Pages	Dynamic Web Pages Syntax	Web pages for client systems access over HTTP
HL7 FHIR	HL7 Fast Healthcare Interoperability Resources	Data Model and Entity Definition
SignalR	ASP.NET real-time web functionality	Getting pushed data from mediCAP to the client system display
Continua	Standards and Specifications for ensuring interoperability of devices	Communications between Devices and Device Gateways

Figure 4: Technologies Used

URL References:
RESTful: https://en.wikipedia.org/wiki/Representational_state_transfer
HL7 FHIR: https://www.hl7.org/fhir/
SignalR: http://www.asp.net/signalr
Continua: http://www.continuaalliance.org/products/design-guidelines

4. Interface View

4.1 Device to Device Gateway

Devices shall use either Bluetooth or ZigBee for connecting to the Device Gateway.
The message that is passed from the Device to the Device Gateway shall be compliant to the Continua standard.

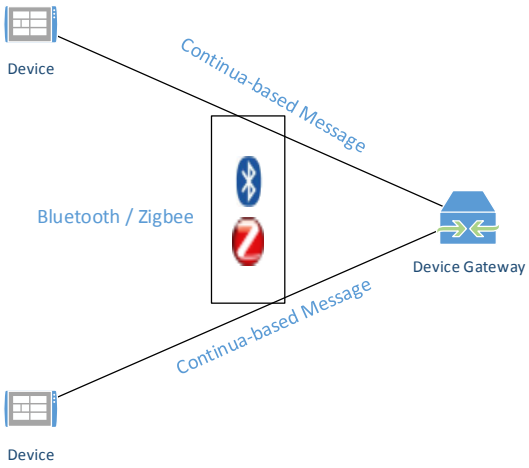


Figure 5: Device Interface to Device Gateway

4.2 Device Gateway to mediCAP

The Device Gateway receives data from the devices and then sends the data to the mediCAP. The mediCAP is a standards based system that supports HL7 FHIR standard for the data format. Therefore the Device Gateway shall send the data in the following format:

- HL7 FHIR data format
- JSON syntax based message format

E.g. of a message for blood pressure which uses the “Observation” resource type of the FHIR specification:


```
{"resourceType": "Observation",
 "code": {
  "coding": [
   {
    "system": "http://loinc.org",
    "code": "8310-5",
    "display": "Body temperature"
   }
  ]
 },
 "valueQuantity": {
  "value": 36.9,
  "units": "Cel"
 },
 "effectiveDateTime": "2015-08-21T11:27:00+08:00",
 "status": "final",
 "device": {
  "reference": "Device/Cardi-VS2014110210001"
 }
}
```


Figure 6: Sample JSON Message for Blood Pressure

The message is sent to the mediCAP via RESTful Web Service.
For example, to update the vital sign value for Patient with ID p001, the following steps are carried out:

1) The client makes a web service call:
   PUT http://sesserver/Patient/p001
2) The request body shall be a "Resource" with an id element that is the value "p001" specified in the URL. This "Resource" content shall contain the necessary data to be updated into the system.

4.3 Client System to mediCAP

Typical clients of the system include:

1) Smartphone
2) Mobile Tablet
3) PC (Desktop / Laptop)
4) External App Server

The client system interacts with the mediCAP in the following ways:

- Option 1: via RESTful Web Service that returns the response in JSON format
- Option 2: via the mediCAP's ASP.NET Razor pages that mediCAP provides (e.g. Patient List, Device List, etc)
- Option 3: via SignalR client connection to the mediCAP's SignalR hub

The following table shows some scenarios for the above three options:

Scenario	**Option**
Android/iOS app displaying patient list in native Android/iOS UI component	1
Android/iOS app displaying mediCAP patient list mobile page in Android/iOS WebView component	2
Real-time update of patient's vital sign on the client's dashboard page	3

5. STE mediCAP System Functionalities

The mediCAP platform provides some functions out of the box:

- User Management
- Device Management
- Management of Patient/Device association

5.1 User Management

Users are accounts that have access to the mediCAP System.
mediCAP provides the management of user accounts by different systems. That is, each system in the mediCAP server has their own set of users.
This is the account that client system needs to provide for authentication when carrying out web service call.
mediCAP also allows the creation of new user account from client via web service.

5.2 Patient Management

mediCAP provides the management of patient records by different systems. That is, each system in the mediCAP server has their own set of patients.
mediCAP also allows the creation of new patient account from client via web service.

5.3 Device Management

mediCAP provides the management of device records by different systems. That is, each system in the mediCAP server has their own set of devices.
mediCAP also allows the creation of new device from client via web service.

5.4 Management of Patient / Device Association

In many cases, the client needs to know which patient or patients are associated with a device. The mediCAP provides web services for retrieving such information.
The mediCAP also allows the association / de-association of patients from the device.

6. Authentication

6.1 Overview

The **mediCAP** uses header keys to validate incoming requests.
The client has to request for a valid API Key before making an actual request.
Refer to the following diagram for the API Key request process.

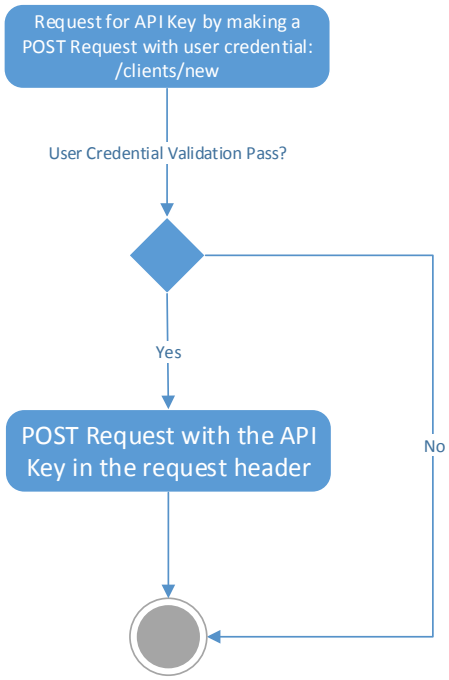


Figure 7: Requesting for API Key

The API header key and key type must be included in all API requests to the server.
`X-API-KEY: system-assigned-key`
`X-API-TYPE: SERVER / BROWSER / SENSOR`
An API Key Type tells the server what type of key the user is sending.

The API Key has an expiry time based on certain time of inactivity.
Once expired, the client needs to obtain a new API Key to make further requests.

6.2 Making Requests

All requests to the **mediCAP** require the following http header elements:

- Content-Type: `application/json`
- X-API-KEY: `system-assigned-key`
- X-API-KEY-TYPE: `key-type-here`

7. Annex A

Introduction to Continua

The Continua Design Guidelines (CDG) defines a framework of underlying standards and criteria required to ensure the interoperability of devices and data used for personal connected health. It also contains design guidelines that further clarify the underlying standards or specifications by reducing options or by adding a missing feature to improve interoperability. These guidelines focus on the following interfaces:

- TAN-IF – Interface between touch area network (TAN) health devices and application hosting devices (AHDs)
- PAN-IF – Interface between personal area network (PAN) health devices and AHDs
- LAN-IF – Interface between local area network (LAN) health devices and AHDs
- WAN-IF – Interface between AHDs and wide area network (WAN)
- HRN-IF – Interface between WAN health devices and Health Record Network

The CDG are a product of the Personal Connected Health Alliance which is an international not-for-profit industry organization enabling end-to-end, plug-and-play connectivity of devices and services for personal health management and healthcare delivery.

Its mission is:

Generating greater awareness, availability and access to plug-and-play, consumer-friendly personal health technologies to empower individuals to better manage their health and wellness, anywhere at any time.

8. Annex B

Introduction to HL7 FHIR

FHIR® – Fast Healthcare Interoperability Resources (hl7.org/fhir) – is a next generation standards framework created by HL7. FHIR combines the best features of HL7's v2, HL7 v3 and CDA product lines while leveraging the latest web standards and applying a tight focus on implementability.

FHIR solutions are built from a set of modular components called "Resources". These resources can easily be assembled into working systems that solve real world clinical and administrative problems at a fraction of the price of existing alternatives. FHIR is suitable for use in a wide variety of contexts – mobile phone apps, cloud communications, EHR-based data sharing, server communication in large institutional healthcare providers, and much more.

Why FHIR is better

FHIR offers many improvements over existing standards:

- A strong focus on implementation – fast and easy to implement (multiple developers have had simple interfaces working in a single day)
- Multiple implementation libraries, many examples available to kick-start development
- Specification is free for use with no restrictions
- Interoperability out-of-the-box– base resources can be used as is, but can also be adapted for local requirements
- Evolutionary development path from HL7 Version 2 and CDA – standards can co-exist and leverage each other
- Strong foundation in Web standards– XML, JSON, HTTP, OAuth, etc.
- Support for RESTful architectures and also seamless exchange of information using messages or documents
- Concise and easily understood specifications
- A Human-readable wire format for ease of use by developers
- Solid ontology-based analysis with a rigorous formal mapping for correctness

Flexibility

A central challenge for healthcare standards is how to handle variability caused by diverse healthcare processes. Over time, more fields and optionality are added to the specification, gradually adding cost and complexity to the resulting implementations. The alternative is relying on custom extensions, but these create many implementation problems too.

FHIR solves this challenge by defining a simple framework for extending and adapting the existing resources. All systems, no matter how they are developed, can easily read these extensions and extension definitions can be retrieved using the same framework as retrieving other resources.

In addition, each resource carries a human-readable text representation using html as a fallback display option for clinical safety. This is particularly important for complex clinical information where many systems take a simple textual/document based approach.

Example Resource: Patient

This simple example shows the important parts of a resource: a local extension, the human readable HTML presentation, and the standard defined data content.

standard defined data content.

```
<Patient xmlns="http://hl7.org/fhir">
  <id value="glossy"/>
  <meta>
    <lastUpdated value="2014-11-13T11:41:00+11:00"/>
  </meta>
  <text>
    <status value="generated"/>
    <div xmlns="http://www.w3.org/1999/xhtml">
      <p>Henry Levin the 7th</p>
      <p>MRN: 123456. Male, 24-Sept 1932</p>
    </div>
  </text>
  <extension url="http://example.org/StructureDefinition/trials">
    <valueCode value="renal"/>
  </extension>
  <identifier>
    <use value="usual"/>
    <type>
      <coding>
        <system value="http://hl7.org/fhir/v2/0203"/>
        <code value="MR"/>
      </coding>
    </type>
    <system value="http://www.goodhealth.org/identifiers/mrn"/>
    <value value="123456"/>
  </identifier>
  <active value="true"/>
  <name>
    <family value="Levin"/>
    <given value="Henry"/>
    <suffix value="The 7th"/>
  </name>
  <gender value="male"/>
  <birthDate value="1932-09-24"/>
  <careProvider>
    <reference value="Organization/2"/>
    <display value="Good Health Clinic"/>
  </careProvider>
</Patient>
```

Resource Identity & Metadata

Human Readable Summary

Extension with URL to definition

Standard Data:
- MRN
- Name
- Gender
- Birth Date
- Provider

FHIR has resources for administrative concepts such as patient, provider, organization and device as well as a wide variety of clinical concepts covering problems, medications, diagnostics, care plans, financial concerns and more.

9. Annex C

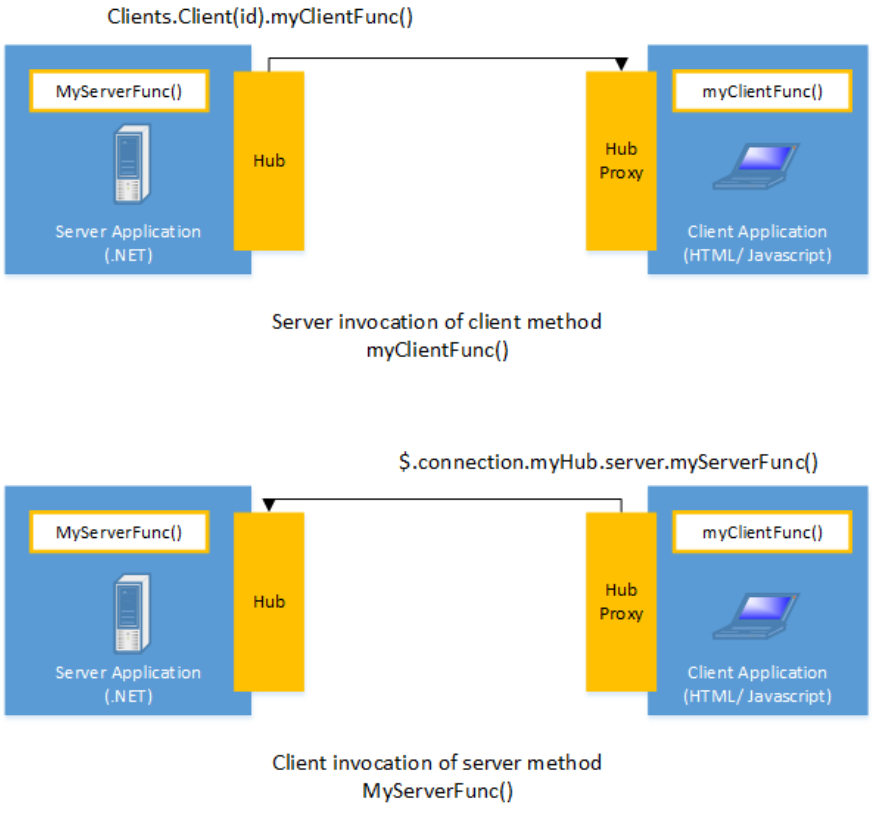
Introduction to SignalR

ASP.NET SignalR is a library for ASP.NET developers that simplify the process of adding real-time web functionality to applications. Real-time web functionality is the ability to have server code push content to connected clients instantly as it becomes available, rather than having the server wait for a client to request new data.

SignalR can be used to add any sort of "real-time" web functionality to your ASP.NET application. While chat is often used as an example, you can do a whole lot more. Any time a user refreshes a web page to see new data, or the page implements long polling to retrieve new data, it is a candidate for using SignalR. Examples include dashboards and monitoring applications, collaborative applications (such as simultaneous editing of documents), job progress updates, and real-time forms.

SignalR also enables completely new types of web applications that require high frequency updates from the server, for example, real-time gaming.

SignalR provides a simple API for creating server-to-client remote procedure calls (RPC) that call JavaScript functions in client browsers (and other client platforms) from server-side .NET code. SignalR also includes API for connection management (for instance, connect and disconnect events), and grouping connections.



SignalR handles connection management automatically, and lets you broadcast messages to all connected clients simultaneously, like a chat room. You can also send messages to specific clients. The connection between the client and server is persistent, unlike a classic HTTP connection, which is re-established for each communication.

SignalR supports "server push" functionality, in which server code can call out to client code in the browser using Remote Procedure Calls (RPC), rather than the request-response model common on the web today.

SignalR applications can scale out to thousands of clients using Service Bus, SQL Server or Redis.

SignalR is open-source and accessible through GitHub.