Bringing Web Services to IoTivity

Opportunities, Challenges & Approaches

Sanjeev BA
Open Source Group
Samsung Electronics
Background

Vertical Domains (Health, Manufacturing, Education, Consumer)

Consumer Services

Enterprise Services

REST API

SOAP based

Cloud Solutions

JSON

XML
Opportunities
IoT & The App Conundrum

- Developers looking for niche areas
- Direct revenues from apps slowing down
- 52% make < $1000 / month
- 53% working on IoT related apps already.
- Estimated 5.5 Million mobile app devs.
- Smart Home and Wearables
- Users already have app fatigue

* Source: VisionMobile™ Developer Economics Q1 2015: State of the Developer Nation
Services in the IoT Context

**Critical**
- Legacy system integration
- Regulatory

**Contextual**
- Sensor + service integration

**Enriched Consumer Experience**
- Smart Shopping Cart
- Tap device to order
Challenges
SDK Overload for Developers

App Developers (Ex. Streaming Music)

- IoT SDK
- Iotivity SDK
- AllJoyn SDK
- MQTT SDK
- Platform SDK
  - Developer. Tizen.org
  - Developer. android.com
  - Developer. apple.com
- Web Service SDK
  - developers.facebook.com
  - dev.twitter.com
  - developer.paypal.com
THING MAKERS ARE LOOKING TO DIFFERENTIATE WITH THE HELP OF WEB SERVICES
App Overload For Users

User ≠ Service Integrator
Current Approaches
Dedicated Hub + Cloud Service Model
Service Plugin Model

- Package Download Server
- Cloud Service
- Web Service Plugin
- Web Service Profile

IoT Network
Distributed App Control Model

IoT Network
Distributed App Control

Service Integrator Backend
Web Service Interface
Web Service Interface (WSI)

- Enabler of interactions
- IoTivity applications & Web Services
- Web service abstraction
Key Concepts

- Roles
  - Web Service Provider
  - WSI
  - IoT Device Manufacturer
  - IoT App Developer
  - Users

- Service Description
  - Services
  - Capabilities
  - Authentication and Authorization
Roles

- Web Service Provider
- Users
- Application Developer
- Device Manufacturers
Service Description

- What is service description
- A way to represent web services to WSI
- Service Block (JSON)
  - Metadata
  - Authorization
  - Capability List
  - TAGS
Service Block Example

META DATA

"name": "Openweathermap",
"id": "org.openweathermap",
"description": "A service that has weather for more than 200,000 cities",
"logo": "http://openweathermap.org/images/OWM_logo32_32.png"

CAPABILITY LIST

capabilities:
{Capability1: },
{Capability2: },
...
{CapabilityN: }

TAGS

"tags": ["search",
  "weather"
]
Capability

- What is a capability
  - Represents different API offered by the Service
  - Capability Block (JSON)
    - Metadata block
    - Headers
    - Responses
    - TAGS

- Post Tweet
- Now Trending
- Timeline
- Status

Service Description

WSI
Capability Example

CAPABILITY DESCRIPTION

capability:
{
  id: org.openweathermap.findbycity,
  isauthrequired: true,
  endpoint: api.openweathermap.org/data/2.5/weather
  endpointtype: REST,
  operation: GET,
  params: {
    q: {{cityname}}
  },
  headers: {
    x-custom: {{cityname}}
  },
  response: {
    <response JSON body format>
  }
}
Authorization Example

auth:
  type:
  subtype:
    appcredentials:
      cust_key:
      cust_sec:
      oauth_key:
      oauth_sec:
Applying Service Description
- A Use Case
Use Case #1: Web Service + IoT Device

Web Service

Weather Service

IoT Alarm App

Alarm Event

Thin Device

Brew my coffee based on weather
Use Case #1

Initialization

Open WeatherMap

Service Description (Offline / Download)

WSI

WSI Connect()

IoTivity Coffee Maker

obServeResource()

onObserve()

Smart Alarm App (w/ IoTivity)

WSI Capability Request ("Weather", "Time", "Location")

Find List of Providers : Tag = Weather

Result From Service Description

App Logic

Hot Cappuccino

Smart Phone with WSI
Use Case #1: Adding Service Provider

**Smart Phone with WSI**

**WSI**

**Smart Alarm App (w/ IoTivity)**

**IoTivity Coffee Maker**

**Accuweather**

---

**Initialization**

WSI_Capability_Request

("Weather", "Time", "Location")

Find List of Providers: Tag = Weather

Result from Service Description

App Logic

Hot Cappuccino

---

RESTful endpoint

JSON Response

WSI_Connect()

obServeResource()

onObserve()
Use Case #2: Share with friends

- Tizen Evangelist @tizenevangelist

- Sharing Service

- IoT Alarm App

- Thin Device

- Coffee recipe
Use Case #2: Tweet The Recipe

Twitter

WSI

IoT Alarm App

IoTivity Coffee Maker

Check Service Description & local cache

Twitter Auth Flow

Cust Key + Secret
{MSG}

User Auth
If required
Update
AuthCache

Configuration Phase

Post Recipe to Twitter

"Share" To Twitter

Report Status

Hot Cappuccino With Cream Recipe

Smart Phone with WSI
Putting it all together

Web Services Interface

WSI Configuration App

WSI API Interface

Web Services Manager
Service Handler
WSI Daemon / Service
Auth Manager
Config & Caching

Platform Adaptation
Linux
Android
Web
Tizen
Web Service Interface Daemon

- **Web Services Manager**
  - Parses the service description
  - Provides representation of the description

- **Service Handler**
  - Invokes requested API
  - Processes the response
  - Uses Auth Manager to delegate Auth flows.
  - Uses platform HTTP stack

- **Auth Manager**
  - Performs Auth related flows
  - Manages application specific auth cache.
  - Token Refresh Handling

- **Config & Caching**
WSI API Interface

- Provides a simple set of API to discover and utilize services

  - **API overview**
    - `WSI_Connect()` Connect to the WSI Daemon / Service Application
    - `List<Service> WSI_Get_Services(“User Action Tag”)`
    - `List<Capability> WSI_Get_Capabilities(Service)`
    - `Request_Handle WSI_Capability_Request(Capability, Callback)`
    - `Callback(Request_Handle, Capability_Result)"`
Configuration App

- Provides frontend for users to configure web services.
- Enabling and disabling individual capabilities
- Helps with Auth flows involving a browser.
Deployment View: Linux, Tizen

Web Service

WSI Config App
WSI C++ Library

IoTivity Application
IoTivity SDK (C++)

Web Service Interface

WebService Manager
Auth Cache

WSI DBus Or Service App

Send Tweet
Get Sentiment
Challenges Addressed By WSI

SDK Overload?
- Partial

App Overload?
- No

Siloed Ecosystems?
- OIC & IoTivity

Enable Web Services Integration?
- Yes
Development Plan & Status

- **Phase 1 (Oct ‘15)**
  - Schema definitions for popular services
  - Schema Parsing, Data models
  - Demo Web Service Use Cases
  - Linux, Android and Tizen support

- **Phase 2 (Nov ‘15)**
  - Schema improvement (new features)
  - Multiuser support
Demo
Q & A
License Attribution For Images

- "simpleicon social media" is an icon pack by simpleicon under License: CC BY 3.0
- Icons made by Freepik from www.flaticon.com are licensed by CC BY 3.0
- Icons made by Rami McMin from www.flaticon.com are licensed by CC BY 3.0
- Icons made by Situ Herrera from www.flaticon.com are licensed by CC BY 3.0
- Icons made by Anton Saputra from www.flaticon.com are licensed by CC BY 3.0
Role: IoT Device Manufacturer

- **Function**
  - Makers of ‘Things’
  - Spec Compliance

- **Expectations**
  - Increased user adoption
  - Feature Differentiation
Roles: App Developers / Integrators

Function

- New breed of “apps”
  - {Web services & IoT devices} Apps
  - IFTTT, Uber, Makerspace

Expectations

- Maximize Asset Reuse
- Simpler development
- “What to” not “How to”
- New revenue streams
- Monetization opportunity
Users

Function
- Consume web service & IoT Devices
- Provide required credentials
  - Authorizing apps to use web services

Expectations
- Zero/Minimal Setup overhead
- Retain existing workflow
  - Enable new use cases.
Role: Web Service Provider

- **Function**
  - Hosts the web service
    - Maps, Photos, Music Streaming
  - Provides access via API and Apps.

- **Expectations**
  - Extending service into the IoTivity network
  - Minimal/No changes in existing workflows
  - More user engagement from multiple sources
Use Case #1: Auth Flow

OpenWeatherMap

WSI

IoT Alarm App

Register to get APPID/APIKEY: 833e2f1ded3644f180c6d724fejksdfsdkj

Service Description

http://api.openweathermap.org/data/2.5/forecast/city
Params
id={Location}
APPID={APIKEY}

Response JSON
{
  city: {name},
  LIST: {
    date: {d},
    temp: {t},
    unit: {units},
    humidity: {h},
  }
}

Get Capabilities ("Weather")

List of "Weather" Services
Params: {Location}, {APIKey}
Response: {name}, list <{date}, {temp}>

Get Weather
Location ID: 443742
APIKey: 833e2f1ded3644f180c6d724fejksdfsdkj

Callback
{name}, list <{date}, {temp}>

Cache APIKEY

http://api.openweathermap.org/data/2.5/forecast/city
Params
id=443742
APPID=833e2f1ded3644f180c6d724fejksdfsdkj
Use Case #2: Auth Flow

Service Description

REQUEST TOKEN
POST https://api.twitter.com/1.1/statuses/update.json
auth:{ ... }
Params status={msg}
Header Authorization: {Oauth}

ACCESS TOKEN
https://api.twitter.com/1.1/statuses/update.json
auth:{ ... }
Params status={msg}
Header Authorization: {Oauth}

STATUS
POST https://api.twitter.com/1.1/statuses/update.json
auth:{ ... }
Params status={msg}
Header Authorization: {Oauth}

Twitter
WSI
IoT Alarm App
Twitter User

POST /oauth/request_token
Req token and secret

Client Key + Secret

Cust Key + Secret

Authenticate User + Authorize App

Authorize App

Oauth Token + Secret

Post {msg} to twitter

Post {msg} to twitter