Sensible Primer

ABSTRACT

Sensible is the name given to the reference implementation of Monohm's proposed Web of Things technology stack. The philosophy behind the proposal is that technologies which are accepted in the Web and Internet community can be repurposed to fulfil requirements presented by the Web of Things.

ARCHITECTURE

A Sensible application represents a service on a network and facilitates two distinct functions- (1) discovery of the service and (2) interaction with the service. Sensible uses Multicast DNS, perhaps better known by the name of Apple's implementation - Bonjour - to facilitate discovery, and HTML or JSON over HTTP to facilitate interaction.

A Sensible application is essentially two applications in one. The first is a wrapper which facilitates discovery and interaction. The second is the actual application which interacts with the service, "served" by the wrapper's interaction facility. For clarity, these applications will be referred to as "wrapper application" and "service application" from hence forth.

For example, when Sensible is running under Node.js, the wrapper is an application written in Javascript, which advertises the service via MDNS and implements a web server to serve the resources which comprise the service application itself. When Sensible is running under Firefox OS or Chrome, the wrapper is *also* a web application, implementing the same functionality as the Node.js application.

INTERACTION OPTIONS

For the actual implementation of the device application, Sensible offers two options. Ideally the manufacturer of the device would opt to author a branded experience, which is then delivered to the user as a conventional web application. This then offers UI to interact with the facilities offered by the underlying service or device. However, Sensible also offers another option. The manufacturer can opt instead to simply choose some images, author JSON files describing the properties managed by the service or device, and have the default Sensible service application generate UI from there. The sample service applications included in the Sensible distribution usually follow the second pattern.

CONFIGURATION

Sensible applications require some configuration before they can be advertised and served. Minimally, Sensible requires the name of the service, for example "Garage Door", its type, which is currently always "_sensible._tcp.local", and the TCP port on which the service is hosted. The latter can be anything, but Monohm recommends a value above 1024 so that the service does not need to be run as root on Unix compliant platforms.

(Ultimately, Monohm aims to establish a reserved port number space for Sensible applications.)

By default, the configuration of a Sensible application is public, and can be obtained by connecting to the configured TCP port and asking for the /config/get REST URL. Configuration can be updated by sending

HTTP parameter key and value pairs to the /config/set REST URL. It is expected however that security considerations will in future restrict access to the configuration.

Also by default, the Sensible configuration is held in a file called sensible_config.json in the root level of the application directory. Example follows --

```
{
  "name" : "Jukebox",
  "type" : "_sensible._tcp.local",
  "port" : 3003,
  "hostname" : "jukebox.local"
}
```

name
This is the name of the service, which will appear in MDNS/Bonjour searches. This can be anything and is recommended to be something human-readable and descriptive of the service. For example, if your office had five jukeboxes, you might want to name them after their location, like “Engineering Jukebox”.

type
This is the MDNS/Bonjour service type. Currently, Sensible uses “_sensible._tcp.local” for all its services, and by default, Sensible discovery applications only search for this type. This may well change in the future.

port
This is the TCP port number on which the service application is available.

hostname
If this property is set, Sensible will resolve MDNS/Bonjour host resolution requests, with the machine’s IP number. This can be useful if you have clients which do not directly have MDNS/Bonjour discovery capability - for example, interacting with the Jukebox could be done from any browser simply by going to the “jukebox.local:3003” URL.

CUSTOMISATION

Sensible applications deal with the outside world via their properties, which are set and retrieved by HTTP and WebSockets requests. Essentially, the service application’s job is to maintain state between its property set and the hardware or service that it is representing.

By default, the property set of a Sensible application is public, and can be obtained by connecting to the the configured TCP port and asking for the /properties/get REST URL. Properties can be updated by sending HTTP parameter key and value pairs to the /properties/set REST URL. It is expected however that security considerations will in future restrict access to the property set.

Also by default, the description of Sensible properties for a given application is held in a file called sensible_properties.json in the root level of the application directory. This file contains name, priority, type, and range information, and is used by the default Sensible service application to derive UI.

```
[
  {
    "name":"latitude",
    "priority":0,
    "type":"number",
    "readonly":true,
    "maximum":90,
    "minimum":-90,
    "value":30
  },
  {
    "name":"latitude",
    "priority":0,
    "type":"number",
    "readonly":true,
    "maximum":180,
    "minimum":-180,
    "value":-122
  }
]
```

name
This is the name of the property, which will appear in HTTP requests and JSON responses. Although any character can be used in a JSON property, names are also used as HTTP parameters, and so lowercase alphanumerics are recommended. Names are unique across the property space.

priority
The priority of a property determines how important it is. For example, a garage door's state - open, closed, opening, etc - would high priority (low number), but the speed at which it is configured to be opening or closing would be of lower priority (high number). The default Sensible service application presents UI for properties in priority order, and commonly, default values are unlikely to be provided for high priority properties.

One might imagine an application which shows the priority zero properties for all discoverable services, showing the summary of device state (the oven is off, the back door is closed, the laundry has stopped, etc) and thereby enabling peace of mind upon departing the house.

type
This is the data type of the property. It is in theory possible to derive data type from the JSON structure itself, but this would be limited to types the language understands, and it is intended that the type space allow extensive validation.

readonly
If this flag is set, then the property is readonly and can not be set remotely.

maxlength
For string-type properties, the maximum length of the string.

minimum
For numeric properties, the minimum value of the property.

maximum
For numeric properties, the maximum value of the property.

Minimally, all a Sensible application needs to do to be compliant is to register listeners for updates to its state, and then communicate those updates to its property set. Once the properties are updated, then the regular Sensible mechanisms can be used to update clients.

For example, a Firefox OS Sensible application which socialises its geolocation would register an event listener for the “geolocation” event, then set its latitude and longitude properties whenever the listener fired. These properties can then be obtained via HTTP or WebSockets via the /properties/get REST URL.

COMPONENTS

The location of the customisation point for a Sensible wrapper application depends on which platform the application is deployed. For Node.js, Firefox OS, and Chrome, respectively, the mainline Javascript files are sensible_app.js, fxos.js, and main.js - each contains a call to the Sensible application factory to construct an application instance appropriate for the platform. The callback for this call is then the customisation point - code which implements the application’s mission belongs here.

Given that a service application is generally a regular web application, it will have normal application structure -- minimally an index.html file and some supporting graphics and style specification files. Depending on how a particular service application implements the persistence of its configuration and properties, there may also be JSON files persisting these data items, also.

In the default service application structure, the following files are present --

index.html - web application representing the service’s UI
sensible-app-icon.jpg - full-size app icon
sensible-app-list-icon.jpg - smaller size app icon
sensible-config.json - description of configuration parameters
sensible-properties.json - description of service properties

The structure is usually accompanied by the following in the case of hosting on Node.js --

sensible-app.command - bash script for launching the wrapper
sensible-app.js - Node.js application launcher, customisation point

The structure is usually accompanied by the following in the case of hosting on Firefox OS --

fxos.html - web application for launching the wrapper
fxos.js - Firefox OS application launcher, customisation point
manifest.webapp - Firefox OS application manifest

The structure is usually accompanied by the following in the case of hosting on Chrome --

main.html - web application for launching the wrapper
main.js - Chrome application launcher, customisation point

manifest.json - Chrome application manifest