Show Runner Service

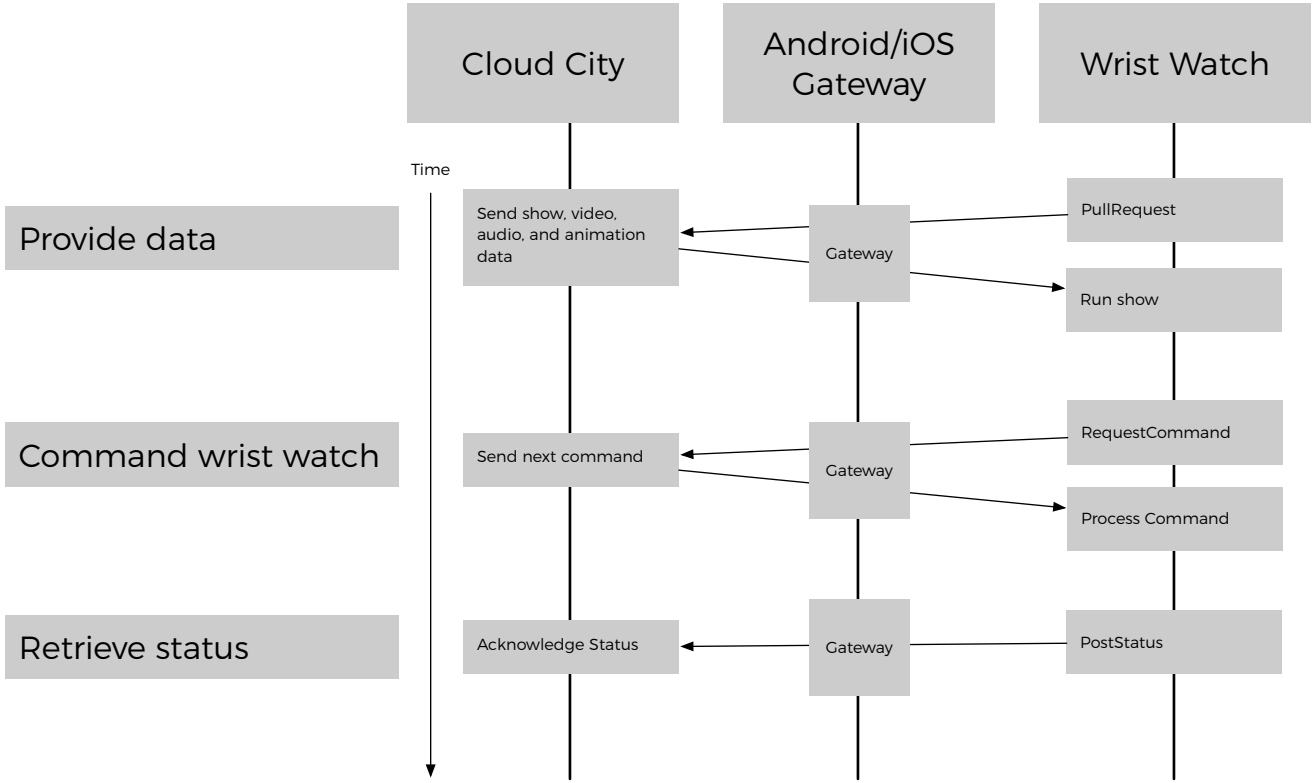
Version: February 25, 2021

fcohen@starlingwatch.com, (408) 364-5508

This document describes the technical details of the Reflections service to command and control the wrist watch. The design goals:

1) Three-tier architecture: Cloud City, Android/iOS Gateway, Wrist Watch Device
2) Provide video, audio, and animation data to the wrist watch device
3) Command the wrist watch device remotely from a Cloud service
4) Retrieve wrist watch status to the Cloud service

Show Runner Service defines communication from Cloud City, Gateway, and wrist watch for the Reflections project.

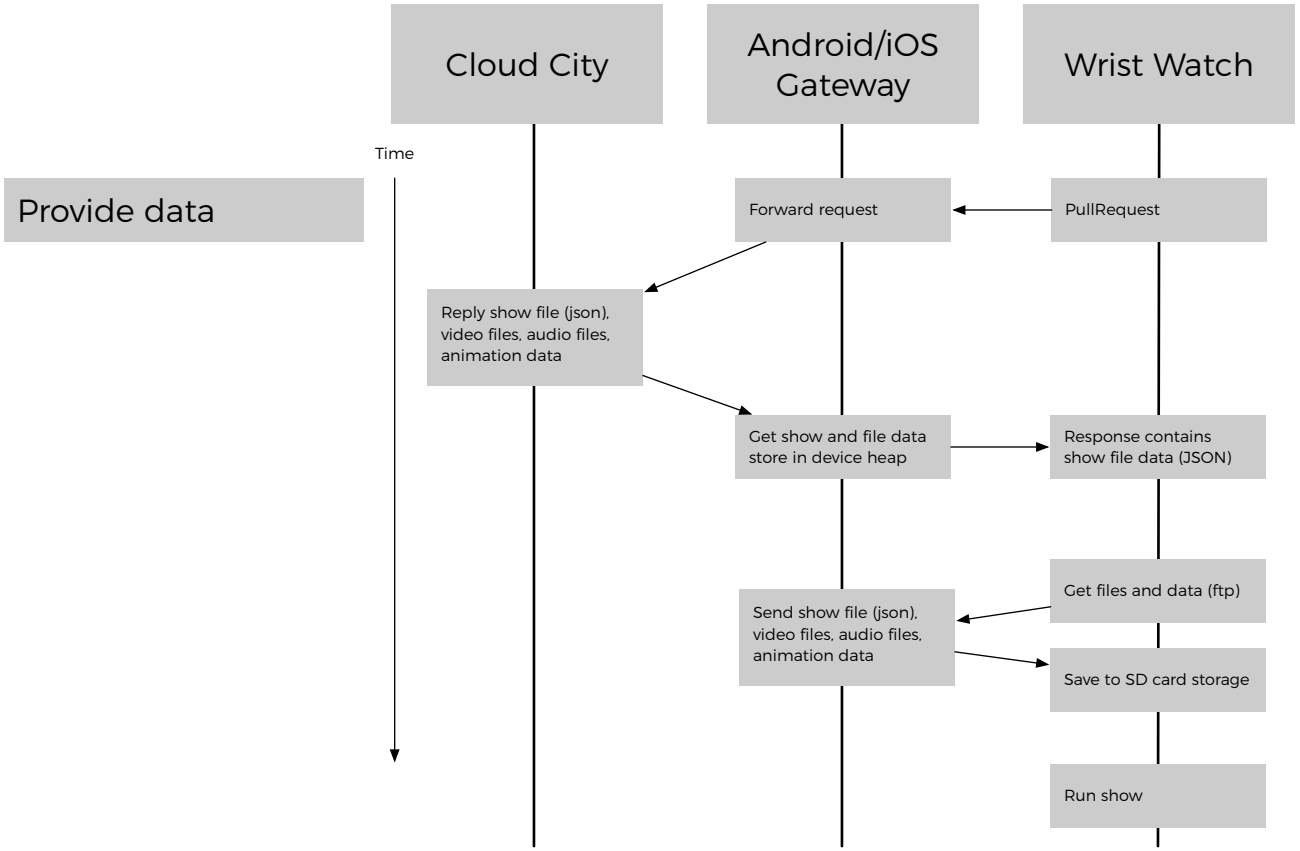


Software implementing the Show Runner Service is at https://github.com/frankcohen/ReflectionsOS

Provide Data Service

Version: February 25, 2021

fcohen@starlingwatch.com, (408) 364-5508



Provide Data flow is asynchronous. The wrist watch initiates a PullRequest and the rest of the flow happens as the show data is ready to transfer from Cloud City and the Gateway.

PullRequest happens over Bluetooth Classic and Serial Bridge. The request is JSON encoded. The response contains the Show in JSON encoding.

Wrist watch uses Bluetooth Classic and Service Bridge as a transport for FTP protocols.

Show file contains all needed instructions to run a show on the wrist watch device.

Provide Data Response

Version: February 25, 2021

fcohen@starlingwatch.com, (408) 364-5508


```
{
 "ReflectionsShow": {
  "title": "Franks First Show",
 “showname”: “frank1”,
  "events": {
   "event": {
    "Name": "Grim Grinning Ghosts from Disneyland Haunted Mansion",
    "trigger": "OnStart",
    "comment", "Plays a sequence once upon startup",
    "sequence": {
     "PlayAtOnce": {
      "comment": "PlayAtOnce plays from the same starting time, and stops when either ends.",
      "playvideo": “Ghosts.mpg,
      "playaudio": "Ghosts.wav"
     }
     "PlayAtOnce": {
      "playvideo": "Hurry.mpg”,
      "playaudio": "Hurry.wav"
     }
    }
   },
   "event": {
    "trigger": "OnHour",
    "comment": "Plays time announcement on the hour",
    "sequence": {
     "PlayAtOnce": {
      "playvideo": “Time$hour.mjpeg",
      "playaudio": "Time$hour.wav"
     }
    }
   },
   "event": {
    "trigger": "ButtonPressed",
    "RunAndReturn": “button”,
    "comment": "This runs the show file and then returns to this show"
    "Arguments": "$trigger"
   }
  }
 }
}
```

The wrist watch initiates a PullRequest and receives a Show response. Response is JSON encoded data.

Triggers are event handlers:
OnStart - happens when the wrist watch finishes running a show and is ready to run the next show
OnHour - happens once an hour
OnQuarterHour - happens 4 times an hour
OnMinute - happens each minute
ButtonPressed - happens when a button is pressed

Sequences happen serially, one after another. They contain one or more Play commands.

PlayAtOnce - starts the play commands at the same moment. For example, video and audio start at the same time.

Shows have an embedded system of variables and resolution. For example, $hour is replaced with the hour of the day.

playvideo - finds the video file in a subdirectory named for the show. If it doesn’t find the file it alternatively uses a file from the /reflections/basedata directory

playaudio - finds the audio file in a subdirectory named for the show. If it doesn’t find the file it alternatively uses a file from the /reflections/basedata directory

RunAndReturn allows one show to run another, pass arguments/values, and return to continue running the show

File names conform to the FAT 8.3 standard

Show Runner and Environment

Version: February 25, 2021

fcohen@starlingwatch.com, (408) 364-5508


```
SD Card
/
 frank1
  show.jsn
  ghosts.mpg
  ghosts.wav
  hurry.mpg
  hurry.wav

 basedata
   device.dat

   button
    show.jsn
    timebig.wav
    timesml.wav
    time1.mpg
    time2.mpg
    time3.mpg
    time4.mpg
    time5.mpg
    time6.mpg
    time7.mpg
    time8.mpg
    time9.mpg
    time10.mpg
    time11.mpg
    time12.mpg

  startup
   show.jsn
   start.mpg
   start.wav
```

The wrist watch runs a Show Runner program.

The wrist watch SD card stores new shows retrieved from Cloud City. It also stores default data to operate the watch. For example, Show Runner uses the startup show stored in the basedata directory when the wrist watch is powered on.

Show Runner registers event handlers from the Button show upon start-up. It then makes a PullRequest to Cloud City.

Internet connectivity is not assumed to be available. Basedata contains a set of pre-programmed shows to play while the watch tries to pair with the gateway and establishes Internet connectivity to Cloud City.

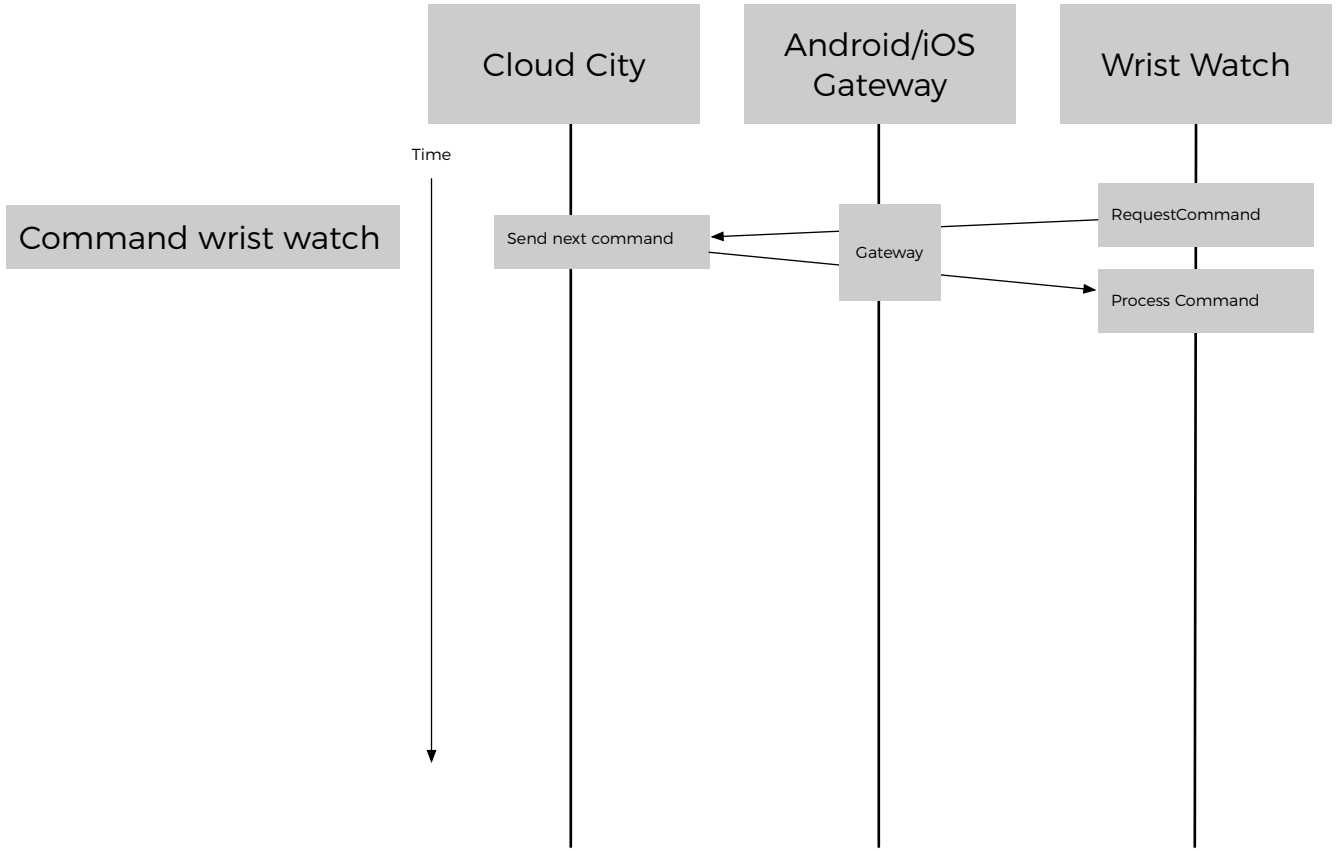
PullRequest receives the show. Show Runner creates a directory on the SD card using the showname element. Show Runner parses the show and downloads referenced files into the directory. It stores the contents of the show to the show.jsn file.

basedata/device.dat contains JSON coded data with a GUID for the device and the client-side encryption key.

Command Service

Version: February 25, 2021

fcohen@starlingwatch.com, (408) 364-5508



Command Wrist Watch identifies the wrist watch device to Cloud City and sends a command to the watch.

At the time of manufacture we create the Device.dat file with GUID and client-side key.

RequestCommand sends the Device GUID value. Cloud City registers new GUID values as devices.

All communication is made in encrypted protocols - using the client and server-side keys.

Response has one of these commands:
PlayShow - plays show by name
SetTime - sets internal clock
SetBaseData - saves a new show to basedata, and does not run the show
Sleep - 1 minute
DeepSleep - 30 minutes

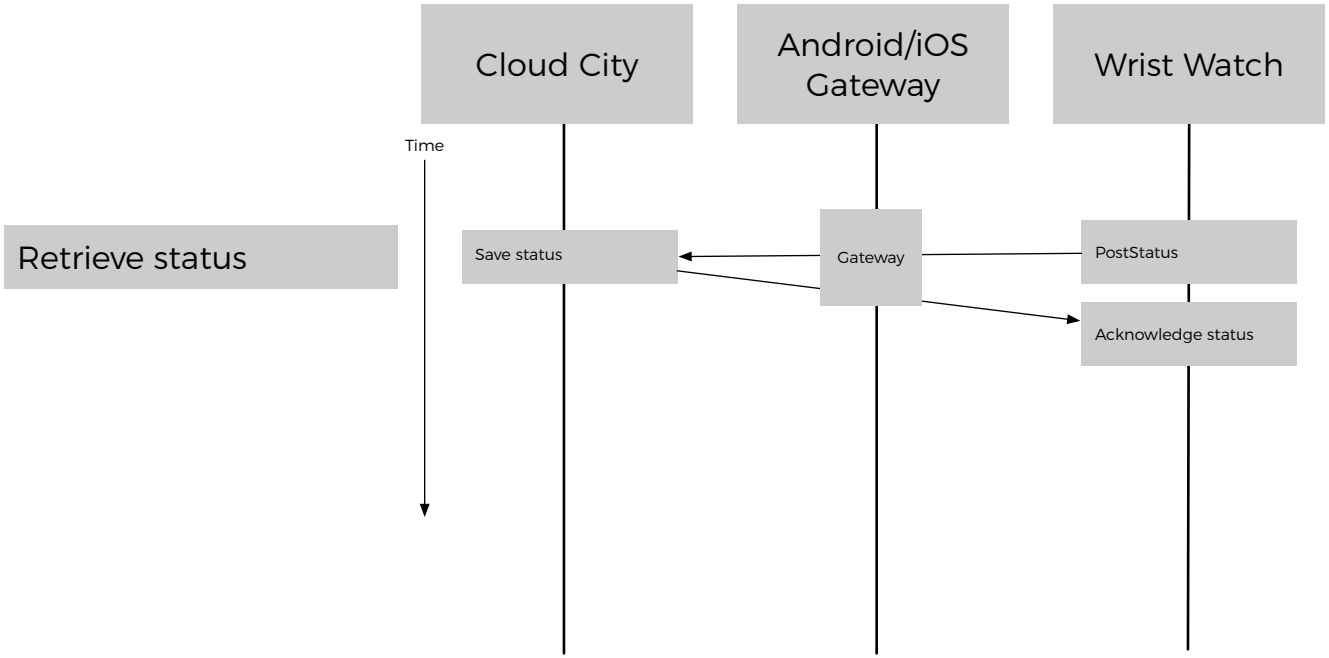
```
{
  "ReflectionsIdentity": {
    "comment": "Wrist watch identity and client-side security key",
    "Device": "a4fa0992-abbc-46f2-96c6-e768e50dfc1e",
    "ClientKey": "IEEE802.00000adc1a7a DSA* 4f:98:25:60:3b:fe:00:ba:db:ad:56:32:c3:e2:8b:3e"
  }
}
```

An example device.json file.

Retrieve Status Service

Version: February 25, 2021

fcohen@starlingwatch.com, (408) 364-5508



Retrieve Status informs Cloud City of the watch device status.

```
{
 "ReflectionsStatus": {
  "comment": "Sends wrist watch status to Cloud City",
  "Device": "a4fa0992-abbc-46f2-96c6-e768e50dfc1e",
  "status", "running",
  "showname", "frank1"
  "lastshow": "hurry",
  "time": "2018-12-10T13:49:51.141Z",
  "version", "1.1",
  "productname", "reflections",
  "leftbutton", "pressed"",
  "middlebutton", "notpressed",
  "rightbutton", "notpressed",
  "batterycycles", "512",
  "batterylevel", "55"
 }
}
```

An example status.json