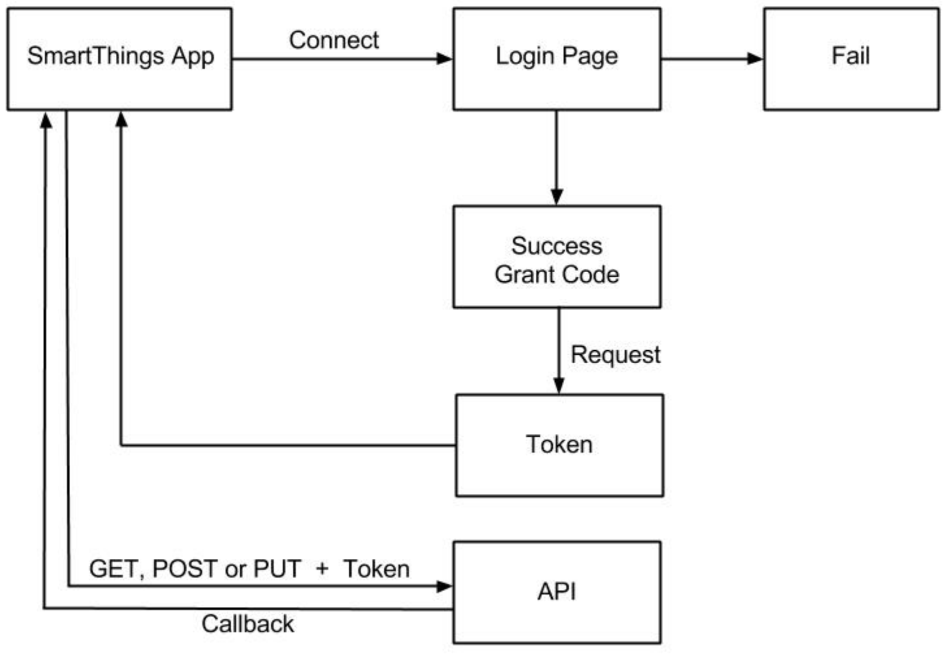
Cloud to Cloud Quick Start

The starter projects demonstrate the best practices to create an efficient cloud-to-cloud device integration.

Exploring the Quick Start Project Demo:

Flow:



Example:

Service Manager:
https://github.com/juano2310/SmartThings_templates/blob/master/harmony_connect.groovy

Device Type:
https://github.com/juano2310/SmartThings_templates/blob/master/harmony_dt.groovy

Step by Step:

1 - Set preferences and starting page

2 - Get the code
3 - Get Token
4 - List devices
5 - Add selected devices as children
6 - Define App specific functions
7 - Define Device Specific functions ()

1 - Set preferences and starting page

```
definition(
    name: "Logitech Harmony (Connect)",
    namespace: "smartthings",
    author: "Juan Pablo Risso",
    description: "Allows you to integrate your Logitech Harmony Hub with SmartThings.",
    category: "SmartThings Labs",
    iconUrl: "https://s3.amazonaws.com/smartapp-icons/Partner/harmony.png",
    iconX2Url: "https://s3.amazonaws.com/smartapp-icons/Partner/harmony%402x.png",
){
            appSetting "clientId"
            appSetting "clientSecret"
            appSetting "callbackUrl"
}
preferences {
    page(name: "Credentials", title: "Nest", content: "authPage", install: false)
}

def authPage() {
    log.debug "authPage"
    def description = null
    if (!state.HarmonyAccessToken) {
                        if (!state.accessToken) {
                                log.debug "About to create access token"
                                createAccessToken()
                        }
        description = "Click to enter Harmony Credentials"
        def redirectUrl = "${serverUrl}/oauth/initialize?appId=${app.id}&access_token=${state.accessToken}"
        return dynamicPage(name: "Credentials", title: "Harmony", nextPage: null, uninstall: true, install:false) {
              section { href url:redirectUrl, style:"embedded", required:true, title:"Harmony", description:description }
        }
    } else {
                        //device discovery request every 5 //25 seconds
                        int deviceRefreshCount = !state.deviceRefreshCount ? 0 : state.deviceRefreshCount as int
                        state.deviceRefreshCount = deviceRefreshCount + 1
                        def refreshInterval = 3

                        def options = discoverDevices() ?: []

                        def numFound = options.size() ?: 0
                        if((deviceRefreshCount % 5) == 0) {
            log.trace "Discovering..."
                                discoverDevices()
                        }
```

```
                return dynamicPage(name:"Credentials", title:"Discovery Started!", nextPage:"", refreshInterval:refreshInterval,
install:true, uninstall: true) {
                        section("Please wait while we discover your Harmony devices. Discovery can take five minutes or
more, so sit back and relax! Select your device below once discovered.") {
                                input "selecteddevice", "enum", required:false, title:"Select Harmony (${numFound} found)",
multiple:true, options:options
                        }
                }
  }
}
```

2 - Get the code

a - map URLs

```
mappings {
        path("/oauth/callback") { action: [ GET: "callback" ] }
        path("/oauth/initialize") { action: [ GET: "init"] }
}
```

b - define "callback"

```
def callback() {
        def redirectUrl = null
        if (params.authQueryString) {
                redirectUrl = URLDecoder.decode(params.authQueryString.replaceAll(".+&redirect_url=", ""))
                log.debug "redirectUrl: ${redirectUrl}"
        } else {
                log.warn "No authQueryString"
        }

        if (state.HarmonyAccessToken) {
                log.debug "Access token already exists"
                discovery()
                success()
        } else {
                def code = params.code
                if (code) {
                        if (code.size() > 6) {
                                // Harmony code
                                log.debug "Exchanging code for access token"
                                receiveToken(redirectUrl)
                        } else {
                                // Initiate the Harmony OAuth flow.
                                init()
                        }
                } else {
                        log.debug "This code should be unreachable"
                        success()
                }
        }
}
```

c- define "init"

```
def init() {
            log.debug "Requesting Code"
   def oauthParams = [client_id: "${appSettings.clientId}", scope: "remote", response_type: "code", redirect_uri:
"${appSettings.callbackUrl}" ]
   redirect(location: "https://home.myharmony.com/oauth2/authorize?${toQueryString(oauthParams)}")
}
```

3 - Get Token

a - add callback URLs to mapping

```
mappings {
            path("/receivedToken") { action: [ POST: "receivedToken", GET: "receivedToken"] }
            path("/receiveToken") { action: [ POST: "receiveToken", GET: "receiveToken"] }
            path("/oauth/callback") { action: [ GET: "callback" ] }
            path("/oauth/initialize") { action: [ GET: "init"] }
}
```

b- define "receiveToken" and auxiliary functions

```
def receiveToken(redirectUrl = null) {
            log.debug "receiveToken"
   def oauthParams = [ client_id: "${appSettings.clientId}", client_secret: "${appSettings.clientSecret}", grant_type:
"authorization_code", code: params.code ]
   def params = [
     uri: "https://home.myharmony.com/oauth2/token?${toQueryString(oauthParams)}",
   ]
   httpPost(params) { response ->
            state.HarmonyAccessToken = response.data.access_token
   }

            discovery()
            if (state.HarmonyAccessToken) {
                        success()
            } else {
                        fail()
            }
}

def success() {
            def message = """
                        <p>Your Harmony Account is now connected to SmartThings!</p>
                        <p>Click 'Done' to finish setup.</p>
            """
            connectionStatus(message)
}

def fail() {
   def message = """
      <p>The connection could not be established!</p>
      <p>Click 'Done' to return to the menu.</p>
   """
```

```
    connectionStatus(message)
}

def connectionStatus(message, redirectUrl = null) {
                def redirectHtml = ""
                if (redirectUrl) {
                                redirectHtml = """
                                                <meta http-equiv="refresh" content="3; url=${redirectUrl}" />
                                """
                }

    def html = """
        <!DOCTYPE html>
        <html>
        <head>
        <meta name="viewport" content="width=640">
        <title>SmartThings Connection</title>
        <style type="text/css">
            @font-face {
                font-family: 'Swiss 721 W01 Thin';
                src: url('https://s3.amazonaws.com/smartapp-icons/Partner/fonts/swiss-721-thin-webfont.eot');
                src: url('https://s3.amazonaws.com/smartapp-icons/Partner/fonts/swiss-721-thin-webfont.eot?#iefix')
format('embedded-opentype'),
                     url('https://s3.amazonaws.com/smartapp-icons/Partner/fonts/swiss-721-thin-webfont.woff') format('woff'),
                     url('https://s3.amazonaws.com/smartapp-icons/Partner/fonts/swiss-721-thin-webfont.ttf') format('truetype'),
                     url('https://s3.amazonaws.com/smartapp-icons/Partner/fonts/swiss-721-thin-webfont.svg#swis721_th_btthin')
format('svg');
                font-weight: normal;
                font-style: normal;
            }
            @font-face {
                font-family: 'Swiss 721 W01 Light';
                src: url('https://s3.amazonaws.com/smartapp-icons/Partner/fonts/swiss-721-light-webfont.eot');
                src: url('https://s3.amazonaws.com/smartapp-icons/Partner/fonts/swiss-721-light-webfont.eot?#iefix')
format('embedded-opentype'),
                     url('https://s3.amazonaws.com/smartapp-icons/Partner/fonts/swiss-721-light-webfont.woff') format('woff'),
                     url('https://s3.amazonaws.com/smartapp-icons/Partner/fonts/swiss-721-light-webfont.ttf') format('truetype'),
                     url('https://s3.amazonaws.com/smartapp-icons/Partner/fonts/swiss-721-light-webfont.svg#swis721_lt_btlight')
format('svg');
                font-weight: normal;
                font-style: normal;
            }
            .container {
                width: 560px;
                padding: 40px;
                /*background: #eee;*/
                text-align: center;
            }
            img {
                vertical-align: middle;
            }
            img:nth-child(2) {
                margin: 0 30px;
            }
            p {
                font-size: 2.2em;
                font-family: 'Swiss 721 W01 Thin';
                text-align: center;
                color: #666666;
                padding: 0 40px;
                margin-bottom: 0;
            }
        /*
            p:last-child {
                margin-top: 0px;
            }
        */
            span {
                font-family: 'Swiss 721 W01 Light';
            }
        </style>
                        ${redirectHtml}
        </head>
        <body>
            <div class="container">
                <img src="https://s3.amazonaws.com/smartapp-icons/Partner/harmony@2x.png" alt="Harmony icon" />
                <img src="https://s3.amazonaws.com/smartapp-icons/Partner/support/connected-device-icn%402x.png" alt="connected
device icon" />
                <img src="https://s3.amazonaws.com/smartapp-icons/Partner/support/st-logo%402x.png" alt="SmartThings logo" />
                ${message}
            </div>
        </body>
        </html>
                """
                render contentType: 'text/html', data: html
}
```

c- define "receiveToken"

```
def receivedToken() {
                def message = """
                                <p>Your Harmony Account is already connected to SmartThings!</p>
                                <p>Click 'Done' to finish setup.</p>
                """
                connectionStatus(message)
}
```

4 - List devices

```
Map discoverDevices() {
    log.trace "Discovering devices"
    discovery()
    if (state.Harmonydevices.hubs) {
        def devices = state.Harmonydevices.hubs
        def map = [:]
        devices.each {
                def hub = it.key
                it.value.response.data.activities.each {
                def value = "${it.value.name}"
                def key = "harmony-${hub}-${it.key}"
                map["${key}"] = value
           }
        }
        state.HarmonyActivities = map
        map
    }
}

def discovery() {
    def Params = [auth: state.HarmonyAccessToken]
    def url = "https://home.myharmony.com/cloudapi/activity/all?${toQueryString(Params)}"
                try {
                                httpGet(uri: url, headers: ["Accept": "application/json"]) {response ->
                                if (response.status == 200) {
                log.debug "valid Token"
                state.Harmonydevices = response.data
                        }
                    }
                } catch (groovyx.net.http.HttpResponseException e) {
        if (e.statusCode == 401) { // token is expired
            state.remove("HarmonyAccessToken")
            log.warn "Harmony Access token has expired"
        }
                } catch (java.net.SocketTimeoutException e) {
                                log.warn "Connection timed out, not much we can do here"
                }
                poll()
    return null
}
```

5 - Add selected devices as children

```
def addDevice() {
    log.trace "Adding childs"
    selecteddevice.each { dni ->
        def d = getChildDevice(dni)
        if(!d) {
            def newAction = state.HarmonyActivities.find { it.key == dni }
            d = addChildDevice("smartthings", "Harmony Activity", dni, null, [label:"${newAction.value} [Harmony Activity]"])
            log.trace "created ${d.displayName} with id $dni"
            poll()
        } else {
            log.trace "found ${d.displayName} with id $dni already exists"
        }
    }
}
```

6 - Define App specific functions

```
def installed() {
                enableCallback()
                if (!state.accessToken) {
                                log.debug "About to create access token"
                                createAccessToken()
                } else {
                                initialize()
```

```
        }
}

def updated() {
        enableCallback()
        if (!state.accessToken) {
                log.debug "About to create access token"
                createAccessToken()
        } else {
                initialize()
        }
}

def uninstalled() {
        def devices = getChildDevices()
        log.trace "deleting ${devices.size()} device"
        devices.each {
                deleteChildDevice(it.deviceNetworkId)
        }
        if (state.HarmonyAccessToken) {
                try {
        log.debug "Success disconnecting Harmony from SmartThings"
                } catch (groovyx.net.http.HttpResponseException e) {
                        log.error "Error disconnecting Harmony from SmartThings: ${e.statusCode}"
                }
        }
}

def initialize() {
        if (selecteddevice) {
                addDevice()
    runEvery5Minutes("discovery")
        }
}
```

7 - Define Device Specific functions ()

```
def activity(dni,mode) {
  def Params = [auth: state.HarmonyAccessToken]
  if (dni == all) {
    def url = "https://home.myharmony.com/cloudapi/activity/off?${toQueryString(Params)}"
  } else {
    def aux = dni.split('-')
    def hubId = aux[1]
    def activityId = aux[2]
    if (mode == "hub")
        def url = "https://home.myharmony.com/cloudapi/hub/${hubId}/activity/off?${toQueryString(Params)}"
    else
        def url = "https://home.myharmony.com/cloudapi/hub/${hubId}/activity/${activityId}/${mode}?${toQueryString(Params)}"
        }
        try {
        httpPostJson(uri: url) { response ->
        if (response.data.code == 200)
        return "Command sent succesfully"
                        else
        return "Command failed"
        runIn(20, "poll", [overwrite: true])
                    }
    } catch (groovyx.net.http.HttpResponseException ex) {
        log.error ex
    }
}

def poll() {
            // check if there are devices installed :)
    def Params = [auth: state.HarmonyAccessToken]
    def url = "https://home.myharmony.com/cloudapi/state?${toQueryString(Params)}"
            try {
                    httpGet(uri: url, headers: ["Accept": "application/json"]) {response ->
            def map = [:]
        response.data.hubs.each {
            map["${it.key}"] = "${it.value.response.data.currentAvActivity},${it.value.response.data.activityStatus}"
        }
        def activities = getChildDevices()
        activities.each { activity ->
            def act = activity.deviceNetworkId.split('-')
            def aux = map.find { it.key == act[1] }
            if (aux) {
                def aux2 = aux.value.split(',')
                def childDevice = getChildDevice(activity.deviceNetworkId)
                if (act[2] == aux2[0] && (aux2[1] == 1 || aux2[1] == 2)) {
                    childDevice?.sendEvent(name: "switch", value: "on")
                    if (aux2[1] == 1)
                        runIn(5, "poll", [overwrite: true])
                } else {
                    childDevice?.sendEvent(name: "switch", value: "off")
                    if (aux2[1] == 3)
                        runIn(5, "poll", [overwrite: true])
                }
            }
        }
                            return "Poll completed"
                }
            } catch (groovyx.net.http.HttpResponseException e) {
        if (e.statusCode == 401) { // token is expired
            state.remove("HarmonyAccessToken")
            return "Harmony Access token has expired"
        }
    }
}
```

Best Practices, Considerations and FAQ

The API redirects to a fixed URL. How can I handle this?

- We are set up to handle fixed callback URLs. The trick is to use a pre-defined redirect hosted by SmartThings for the initial URL, rather than linking directly to the Wink site. For example:

```
def vendorOauthLink = apiServerUrl("/oauth/initialize?appId=${app.id}&access_token=${state.accessToken}")
```

You also need to set up these mappings:

```
mappings {
 path("/oauth/initialize") {
 action: [
 GET: "initializeLink"
 ]
 }
 path("/oauth/callback") {
 action: [
 GET: "authCallback"
 ]
 }
}
```

and create a method that handles the redirect:

```
def initializeLink() {
    def authorizationUrl = "https://winkapi.quirky.com/oauth2/authorize?..."
  redirect(location: authorizationUrl)
}
```

The system automatically puts the appId and access_token in the mobile client session state and retrieves them from there when it gets the callback to /oauth/callback. So you just defined a method to handle the callback as you normally would:

```
def authCallback() {
  ...
}
```

Feedback and next steps

Community dedicated thread?