Mycroft Ai Plasmoid Documentation

Author: Aditya Mehra | Email: Aix.m@outlook.com | Github: https://github.com/aiix

This documentation consist of the workings of the Mycroft Plasmoid which includes how the plasmoid communicates with Mycroft Core and Design ideas implemented for creating an interactive environment for the users. The documentation covers the following topics:

1. Why the Plasmoid & Goals

Introduction

Mycroft Plasmoid is a desktop frontend that intends to interact with the Mycroft Core backend which is developed by the team at Mycroft Ai. It is an open source community committed technology that ties natural language processing, text-to-speech, speech-to-text, and powerful APIs together to create a powerful experience allowing users to manipulate their devices through voice control. Mycroft Ai is the digital implementation of a modern day virtual assistant on the Linux Platform that can help a user conduct a multitude of task via voice / text.

Goals

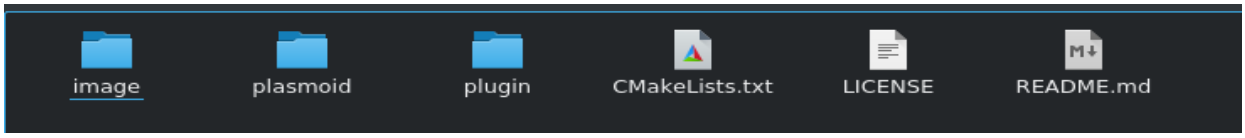
The Mycroft Plasmoid is developed with an aim towards integration of Mycroft Core backend with the Plasma 5 Desktop Environment to provide an open, user rich desktop and an interactive virtual assistant experience to the end users with keeping in mind the principles of free and open source software. This project also aims towards making the virtual assistant a central hub of information and desktop control assistance providing the user a new innovative layer of desktop interactive technology.

2. Packaging Format

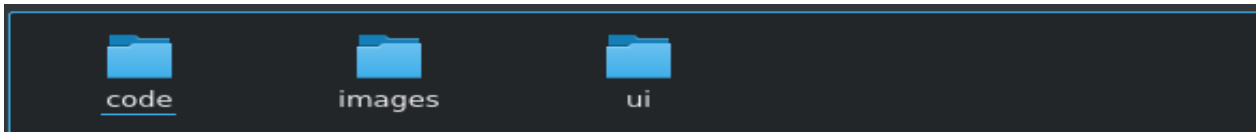
The Mycroft Plasmoid is currently packaged as other similar open source Plasmoids following the packaging guidelines provided by KDE for Plasma 5 Plasmoids.

The plasmoid package structure is as follows:

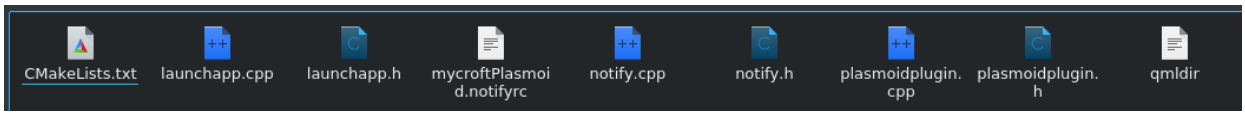
a) The root of the plasmoid consist of the Plasmoid Folder, Plug-ins Folder, Image Folder(System Theme Icons), CmakeLists.txt file



b) The Plasmoid Folder consist of a metadata file and contents which houses the code folder(Shell Scripts), images folder(Plasmoid Content Images) and the UI folder (QML files of the plasmoid)



c) The Plug-ins folder consist of the C++ headers and sources



3. C++ Plug-ins

The Mycroft Plasmoid requires the use of external libs to take advantage of its desktop environment and the C++ Plug-ins provides a base of getting the QML environment to interact with the system.

- The Mycroft Plasmoid makes use of the Qprocess Library to enable the plasmoid to launch Mycroft Start and Stop service scripts.
- The Mycroft Plasmoid makes use of Knotifications Library to enable the plasmoid to notify Mycroft's response to a user query asked via voice interaction
- The Mycroft Plasmoid uses a self plugin to register types with QML so the QML can interact with the Qprocess and Knotifications Libraries.

4. QML & Design

The main.qml plasmoid follows two representation methods Full Representation and Compact Representation. The compact representation displays an icon when clicked upon is expanded to full representation view.

Full Representation {}

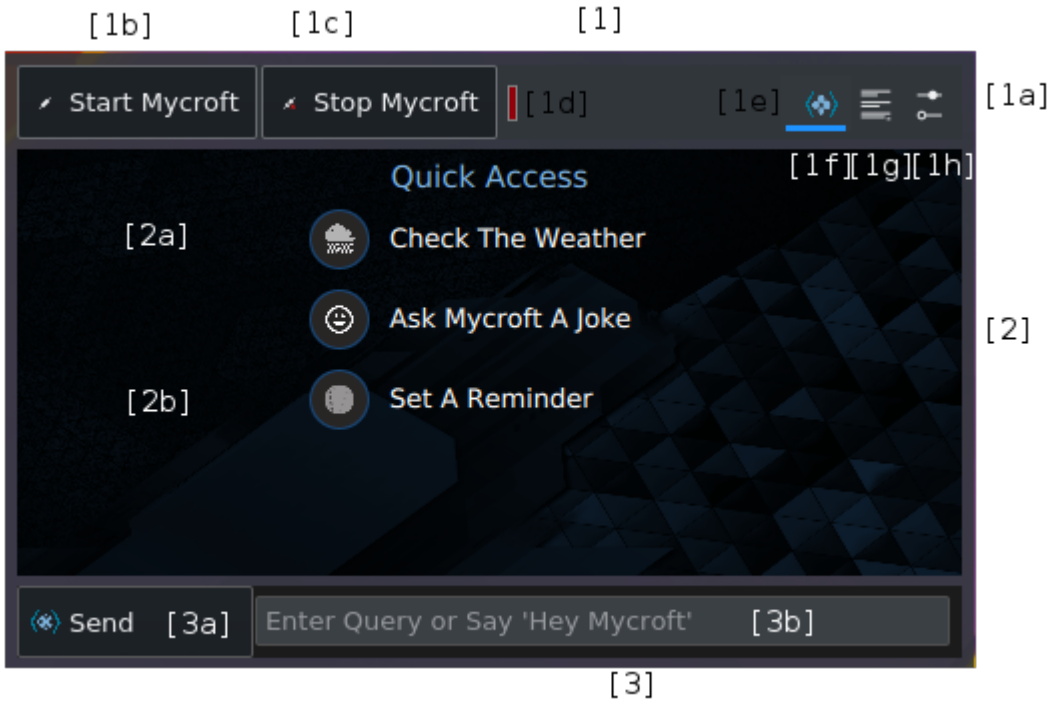


Image: Full Representation Layout View of the Plasmoid

The Full Representation Layout of the plasmoid is divided into three major sections which are represented in Full numbers in the above image. Details about each section and components within the section are mentioned below:

[1] Type: Rectangle – Top Bar: This houses the top bar grid layout consisting of buttons, status expression and tabular bar buttons. The top bar is visible in all 3 tabs.

- [1a] Grid Layout – Top Bar Grid Layout
- [1b] PlasmaComponents.Button – Start Service Button
- [1c] PlasmaComponents.Button – Stop Service Button
- [1d] Rectangle – Status of Service (Colour Code – Define Status)
- [1e] Right Margin – PlasmaComponents.TabBar
- [1f] PlasmaComponents.TabButton – Mycroft Main Interaction Tab
- [1g] PlasmaComponents.TabButton – Skills Information Tab
- [1h] PlasmaComponents.TabButton – Plasmoid Settings Tab

[2] Type: Rectangle – Result Box: This houses the Input & Output Display areas including Loaders for widgets and animations.

- [2a] PlasmaComponents.Paragraphs – Query Input / Query Output
- [2b] Loader Area for Skill Widgets {} & Qt.createcomponents display space

[3] Type: Rectangle – Bottom Bar: This houses the Send Query Button & Text Input Field.

[3a] PlasmaComponents.Button – Send Button

[3b] PlasmaComponents.TextField – Text Input Box

5. Interacting with Mycroft Core

Mycroft Core utilizes the Web-sockets protocol to communicate to with client applications which interacts with mainly three services:

- Service – Connects to Mycroft's infrastructure to process web request. Creates a web-socket local server which the web-socket based client connects too. In this case the Mycroft Plasmoid.
- Skills – Connects the skills services client to the service api via web-sockets to provide skill detection and response based on intent received from the client. In this case example "hello world" string is sent from client ("Mycroft Plasmoid") via web-socket. The skill services matches intent of the client to the intent mentioned within a skill based on which a set dialogue and response ("Hello Human") is generated and delivered over web-sockets back to the client application.
- Voice – Processes handling of Speech to Text based on the STT engine selected as a preference by the user. Mycroft currently supports multiple STT engines including open speech to text which is their own open source STT engine.

The Mycroft Plasmoid utilizes Qt Web-socket library which consist of the following methods and sample of implementation in the plasmoid:

```
import Qt.WebSockets 1.0

 WebSocket {
        id: socket
        url: "ws://0.0.0.0:8181/core"
        onTextMessageReceived: {                          <--- On response from WebSockets service
             var somestring = JSON.parse(message)
             var msgType = somestring.type;
             console.log(msgType);
        }
        socket.active = true;  <--- Initializing Socket connection
    }

 PlasmaComponents.TextField {
         id: qinput
    onAccepted: {   <--- Sending User Input via WebSockets
        var socketmessage = {};
        socketmessage.data.utterances = [qinput.text];
        socket.sendTextMessage(JSON.stringify(socketmessage));
        }
}
```

Image: Example of web-socket implementation

a) Initializing Socket Connection: Setting socket state property to active allows the plasmoid to establish a connection over local-host to Mycroft Core services. In real case scenario the socket state on plasmoid expansion is set to false by default as a user is required to first start the background services provided by Mycroft Core. The "Socket.Active" state is set after a delay of 1200ms to establish a connection after the user interacts with the

PlasmaComponents.Button (“Start Service”). A similar concept is applicable on (“Stop Service”) where “Socket.Active” is reset to false.

b) Sending User Input via Web-sockets: User Interaction is captured by the PlasmaComponents.TextField text property which onAccepted is sent via a JSON container over the established socket connection which houses the user utterance(intent).

c) Response: onMessageReceived the JavaScript Message is parsed via JSON and filtered to match the “Type: Speak” which houses the query response to the users intent message.

6. Interacting with Mycroft Skills

A skill is a class that extends the MycroftSkill class. It is instantiated by the skills container via a create_skill method on the skill module. Mycroft provides users the ability to create their own skills via the simple Mycroft Skills Framework.

The Mycroft Plasmoid interacts with the skills via filtering the onMessageReceived parsed JSON via matching the dialogue of a specific skill that has user information to display. On matching a dialogue the plasmoid is able to create a component or activate a loader based on the response of the skill. Example Implementation:

Weather.qml {}

```
Item {
  anchors.fill: parent
  property alias currentweatherparam: weatherwidgetcurrenttemp.text

    Text {
                id: weatherwidgetcurrenttemp
                color: "#ffffff"
                text: qsTr(" ");
                anchors.top: parent.top
                anchors.topMargin: 30
                font.bold: true
        }
}
```

FullRepresentation.qml

```
onTextMessageReceived: {
            var somestring = JSON.parse(message)
            var msgType = somestring.type;
            console.log(msgType);
            var post = somestring.data.utterances
            outputresponse.text = post
            if (msgType === "speak" && outputresponse.text.isIndexof("Weather") === -1 && outputresponse.text.isIndexof("CurrentTemp") === -1) {
               var componentweather = Qt.createComponent("Weather.qml")
               var loadwin = componentweather.createObject(weatherloader)
}
      }
```

7. Plan for Compliance With The KDE Manifesto

- **Open Governance:** The plasmoid project is and will always remain open to public participation and engagement with collective wisdom to benefit the end user. This project will comply by any decision making processes implied by leadership through a positive discussion channel open to all contributors.
- **Free Software:** The plasmoid project will ensure that the results are always available to all the people at all times. The current alpha release is also available in full open source with everyone having access to it at the time of writing where its currently hosted. It is currently licensed under GPL v3 following the same Mycroft Core licence. The license is always open to being changed to fit the needs of the project which will ensure open source and availability.
- **Inclusivity:** Everyone including all users and developers are welcome to join and participate in the improvement of the plasmoid project.
- **Innovation:** The engagement of future technologies and constantly evolving desktop features ensures the ability to innovate in the project to greater extents. The base of the project is also an innovative idea in itself where by incorporating multiple open source technologies a diverse assistant can be matured to provide maximum user benefits.
- **Common ownership:** Anyone can contribute directly to the plasmoid.
- **End-user focus:** Maturing the User Interface and Desktop Integration to provide reliability and high level functionality to end users.