

|  |
| --- |
| **SyneMobiSafe – KIA**  Vishal Bhardwaj, Assistant Manager – Software  May 2017 |
|  |

Table of Contents

[Introduction 3](#_Toc478485824)

[Document Purpose 3](#_Toc478485825)

[Current State 3](#_Toc478485826)

[Future State 4](#_Toc478485827)

[Development Environment Setup 4](#_Toc478485828)

[ Configuration: New DB Schema 5](#_Toc478485829)

[ Configuration: SyneMobiSafe.xmlFile](#_Toc478485830) 5

[Code Changes 5](#_Toc478485831)

[ DAO (Implementation) Class 5](#_Toc478485834)

[ Module(Localization Implementation) Class 6](#_Toc478485835)

[Enhancement Impact 8](#_Toc478485836)

[ Conclusion 8](#_Toc478485837)

# Introduction

KIA is one of a kind, knowledge based value-add feature provided by SyneMobiSafe which enables the users to get tips and other relevant information they might be seeking about their device or on any technical issue. Users also have the flexibility to perform search on anything related to their device, it might range from security settings to Bluetooth or even Wi-Fi connection issues. However, as the contents or articles begin to grow in number, the search performance is impacted and it takes time to filter the results. Apart from it, the system also doesn't have multi lingual support to perform search.

# Document Purpose

The purpose of this document is to introduce a new mechanism which needs to be integrated to the system in order to provide faster and accurate search results along with multi lingual support for custom search functionality. The document provides the details of the enhancements and implementation techniques analyzed for the SyneMobiSafe KIA Module. The document also provides the impact analysis, details of code and configuration.

# Current State

As per current enhancement and implementation, when a user search for a keyword on the application a service call is made to the search engine looking for all the articles matching the same and are delivered to the user, if available.

# Future State

The proposed enhancement to the KIA module is to update the current system with a new mechanism that will consist of 2 separate sub modules which will be designed and integrated into it. To perform this enhancement-

1. A new content repository will be designed, which will be responsible for tagging the generated articles and then indexing them to the search engine. A tag is like a identifier for the article, for e.g. if the article is related to the bluetooth connectivity or settings, it will be tagged with the keyword "bluetooth". Similarly, all the articles will be tagged after generation, so when a request comes, the search engine will filter out on the basis of these keywords.

2. A new Translation API channel will be designed which will take care of translating searched keywords and auto detecting the native language(from the search text). Once the language is detected, search engine will filter the articles in the specific language itself and forward it to the user.

So, for e.g. if a user search something in Spanish, only Spanish articles will be delivered, if existing.

(Please refer below proposed design).

# Development Environment Setup

This section provides the details for setting up development environment to update the module and its upgrade mechanism. The following configurations are required to be implemented to enhance and upgrade the module:

### Configuration: New DB schema

The database needs to be configured with the details fetched from the new modified type of request object. The schema needs to be modified like -

|  |  |
| --- | --- |
| Column Name | Column Type |
| userID | Number |
| locale | String |
| requestType | String |
| keyword | String |
| status | Boolean |

|  |  |
| --- | --- |
| Column Name | Column Type |
| articleType | String |
| articleTag | String |
| articleIndex | String |
| availability | Boolean |

### Configuration: SyneMobiSafe.xml File

The SyneMobiSafe.xml files needs to be updated and configured as follows –

Locale configurations are declared in this section and these values could be overridden from the values declared in class files.

<LocaleConfig>

<RequestType>GET</RequestType>

<LocaleResponseTimeout>50000</LocaleResponseTimeout>

<SearchSorurce>API</SearchSorurce>

<KeywordType>String</KeywordType>

<SSL>true</SSL>

<SyncFlow>false</SyncFlow>

</LocaleConfig>

# Code Changes

### DAO(Implementation Class)

User Information and search keywords along with other details are fetched from the request object and are processed further to achieve the desired results in order to complete the functionality. Following is one of the sample model classes that can provide the implementation–

public class LocaleDAO {

private Integer userId;

private String locale;

private String requestType;

private String keyword;

private boolean status;

public Integer getUserId() {

return userId;

}

public void setUserId(Integer userId) {

this.userId = userId;

}

public String getLocale() {

return locale;

}

public void setLocale(String locale) {

this.locale = locale;

}

public String getRequestType() {

return requestType;

}

public void setRequestType(String requestType) {

this.requestType = requestType;

}

public String getKeyword() {

return keyword;

}

public void setKeyword(String keyword) {

this.keyword = keyword;

}

public Boolean getStatus() {

return status;

}

public void setStatus(Boolean status) {

this.status = status;

}

}

### Module(Localization Implementation) Class

Following is one of the Translation API class that can be used as per the implementation –

public class LocalizationModule {

private static LocalizationModule mLocModule;

Activity activity;

public static LocalizationModule getLocModuleInstance() {

if (null == mLocModule) {

mLocModule = new LocalizationModule();

}

return mLocModule;

}

public void getLocaleInstance() {

locale = localeObject.getLocaleObject();

locale.init();

}

public String getUserMessage() {

String keyword = activity.getApplicationContext().getUserSearchedKeyword();

return keyword.toString();

}

public void detectUserLanguage(String keyword){

String detectLangURI = getLocaleInstance().getLocaleAPI();

HttpClient client = new HttpClient();

client.createHttpClient();

HttpResponse resp = client.execute(detectLangURI);

handleResponse(resp);

}

public void translatetoNativeLang(String keyword){

String translateToNativeLangURI = getLocaleInstance().getTanslateAPI();

HttpClient client = new HttpClient();

client.createHttpClient();

HttpResponse resp = client.execute(translateToNativeLangURI);

handleResponse(resp);

}

}

# Enhancement Impact

As mentioned earlier, the KIA module needs to be updated and integrated with a new mechanism i.e. a translation API and a tagged content repo to provide faster results. Following are the main advantages of the this implementation.

* **Well Designed:** The tagged content repository is designed in such a way that it can be seamlessly integrated to the search engine and feed tagged articles to it.
* **More Efficient:** The implementation also focuses on saving the time consumed while filtering search response to provide proper and accurate results swiftly, hence the system is more fast and efficient.
* **Easy Implementation:** The translation API provides an easy and brisk implementation, which can be integrated with minimum efforts and less code changes. It just needs to be feed with the data and it will come up with timely reponse.

### Conclusion

Based on the above analysis, we can conclude that, implementing the multi-lingual support through a new Translation API and also integrating a new tagged repository feature to the SyneMobiSafe will help achieve search results swiftly and efficiently and is also user friendly. As a result, performance of the system is also enhanced.