Searchable Configuration

See also

- Creating a Search Interface
- Adding Recent Query Suggestions
- Adding Custom Suggestions

In order to implement search with assistance from the Android system (to deliver search queries to an activity and provide search suggestions), your application must provide a search configuration in the form of an XML file.

This page describes the search configuration file in terms of its syntax and usage. For more information about how to implement search features for your application, begin with the developer guide about Creating a Search Interface.

FILE LOCATION:

`res/xml/filename.xml`
Android uses the filename as the resource ID.

SYNTAX:

```
<?xml version="1.0" encoding="utf-8"?>
<searchable xmlns:android="http://schemas.android.com/apk/res/android"
    android:label="string resource"
    android:hint="string resource"
    android:searchMode=["queryRewriteFromData" | "queryRewriteFromText"]
    android:searchButtonText="string resource"
    android:inputType="inputType"
    android:imeOptions="imeOptions"
    android:searchSuggestAuthority="string"
    android:searchSuggestPath="string"
    android:searchSuggestSelection="string"
    android:searchSuggestIntentAction="string"
    android:searchSuggestIntentData="string"
    android:searchSuggestThreshold="int"
    android:includeInGlobalSearch=["true" | "false"]
    android:searchSettingsDescription="string resource"
    android:queryAfterZeroResults=["true" | "false"]
    android:voiceSearchMode=["showVoiceSearchButton" | "launchWebSearch" | "launchRecognizer"]
    android:voiceLanguageModel=["free-form" | "web_search"]
    android:voicePromptText="string resource"
    android:voiceLanguage="string"
    android:voiceMaxResults="int"
    >
    <actionkey
        android:keycode="KEYCODE"
        android:queryActionMsg="string"
        android:suggestActionMsg="string"
        android:suggestActionMsgColumn="string" />
</searchable>
```

ELEMENTS:

`<searchable>`

Defines all search configurations used by the Android system to provide assisted search.

attributes:

`android:label`

String resource. (Required.) The name of your application. It should be the same as the name applied to the `android:label` attribute of your `<activity>` or `<application>` manifest element. This label is only visible to the user when you set `android:includeInGlobalSearch` to "true", in which case, this label is used to identify your application as a searchable item in the system's search settings.

`android:hint`

String resource. (Recommended.) The text to display in the search text field when no text has been entered. It provides a hint to the user about what content is searchable. For consistency with other Android applications, you should format the string for `android:hint` as "Search *<content-or-product>*". For example, "Search songs and artists" or "Search YouTube".

`android:searchMode`

Keyword. Sets additional modes that control the search presentation. Currently available modes define how the query text should be rewritten when a custom suggestion receives focus. The following mode values are accepted:

Value	Description
`"queryRewriteFromText"`	Use the value from the `SUGGEST_COLUMN_TEXT_1` column to rewrite the query text.
`"queryRewriteFromData"`	Use the value from the `SUGGEST_COLUMN_INTENT_DATA` column to rewrite the query text. This should only be used when the values in `SUGGEST_COLUMN_INTENT_DATA` are suitable for user inspection and editing, typically HTTP URI's.

For more information, see the discussion about rewriting the query text in Adding Custom Suggestions.

`android:searchButtonText`

String resource. The text to display in the button that executes search. By default, the button shows a search icon (a magnifying glass), which is ideal for internationalization, so you should not use this attribute to change the button unless the behavior is something other than a search (such as a URL request in a web browser).

`android:inputType`

Keyword. Defines the type of input method (such as the type of soft keyboard) to use. For most searches, in which free-form text is expected, you don't need this attribute. See `inputType` for a list of suitable values for this attribute.

`android:imeOptions`

Keyword. Supplies additional options for the input method. For most searches, in which free-form text is expected, you don't need this attribute. The default IME is "actionSearch" (provides the "search" button instead of a carriage return in the soft keyboard). See `imeOptions` for a list of suitable values for this attribute.

Search suggestion attributes

If you have defined a content provider to generate search suggestions, you need to define additional attributes that configure communications with the content provider. When providing search suggestions, you need some of the following `<searchable>` attributes:

`android:searchSuggestAuthority`

String. (Required to provide search suggestions.) This value must match the authority string provided in the `android:authorities` attribute of the Android manifest `<provider>` element.

`android:searchSuggestPath`

String. This path is used as a portion of the suggestions query `Uri`, after the prefix and authority, but before the standard suggestions path. This is only required if you have a single content provider issuing different types of suggestions (such as for different data types) and you need a way to disambiguate the suggestions queries when you receive them.

`android:searchSuggestSelection`

String. This value is passed into your query function as the `selection` parameter. Typically this is a WHERE clause for your database, and should contain a single question mark, which is a placeholder for the actual query string that has been typed by the user (for example, `"query=?"`). However, you can also use any non-null value to trigger the delivery of the query text via the `selectionArgs` parameter (and then ignore the `selection` parameter).

`android:searchSuggestIntentAction`

String. The default intent action to be used when a user clicks on a custom search suggestion (such as `"android.intent.action.VIEW"`). If this is not overridden by the selected suggestion (via the `SUGGEST_COLUMN_INTENT_ACTION` column), this value is placed in the action field of the `Intent` when the user clicks a suggestion.

`android:searchSuggestIntentData`

String. The default intent data to be used when a user clicks on a custom search suggestion. If not overridden by the selected suggestion (via the `SUGGEST_COLUMN_INTENT_DATA` column), this value is placed in the data field of the `Intent` when the user clicks a suggestion.

`android:searchSuggestThreshold`

Integer. The minimum number of characters needed to trigger a suggestion look-up. Only guarantees that the system will not query your content provider for anything shorter than the threshold. The default value is 0.

For more information about the above attributes for search suggestions, see the guides for Adding Recent Query Suggestions and Adding Custom Suggestions.

Quick Search Box attributes

To make your custom search suggestions available to Quick Search Box, you need some of the following `<searchable>` attributes:

`android:includeInGlobalSearch`

Boolean. (Required to provide search suggestions in Quick Search Box.) Set to "true" if you want your suggestions to be included in the globally accessible Quick Search Box. The user must still enable your application as a searchable item in the system search settings before your suggestions will appear in Quick Search Box.

`android:searchSettingsDescription`

String. Provides a brief description of the search suggestions that you provide to Quick Search Box, which is displayed in the searchable items entry for your application. Your description should concisely describe the content that is searchable. For example, "Artists, albums, and tracks" for a music application, or "Saved notes" for a notepad application.

`android:queryAfterZeroResults`

Boolean. Set to "true" if you want your content provider to be invoked for supersets of queries that have returned zero results in the past. For example, if your content provider returned zero results for "bo", it should be requiried for "bob". If set to "false", supersets are ignored for a single session ("bob" does not invoke a requery). This lasts only for the life of the search dialog or the life of the activity when using the search widget (when the search dialog or activity is reopened, "bo" queries your content provider again). The default value is false.

Voice search attributes

To enable voice search, you'll need some of the following `<searchable>` attributes:

`android:voiceSearchMode`

Keyword. (Required to provide voice search capabilities.) Enables voice search, with a specific mode for voice search. (Voice search may not be provided by the device, in which case these flags have no effect.) The following mode values are accepted:

Value	Description
`"showVoiceSearchButton"`	Display a voice search button, if voice search is available on the device. If set, then either `"launchWebSearch"` or `"launchRecognizer"` must also be set (separated by the pipe | character).
`"launchWebSearch"`	The voice search button takes the user directly to a built-in voice web search activity. Most applications don't need this flag, as it takes the user away from the activity in which search was invoked.
`"launchRecognizer"`	The voice search button takes the user directly to a built-in voice recording activity. This activity prompts the user to speak, transcribes the spoken text, and forwards the resulting query text to the searchable activity, just as if the user typed it into the search UI and clicked the search button.

`android:voiceLanguageModel`

Keyword. The language model that should be used by the voice recognition system. The following values are accepted:

Value	Description
`"free_form"`	Use free-form speech recognition for dictating queries. This is primarily optimized for English. This is the default.
`"web_search"`	Use web-search-term recognition for shorter, search-like phrases. This is available in more languages than "free_form".

Also see `EXTRA_LANGUAGE_MODEL` for more information.

`android:voicePromptText`

String. An additional message to display in the voice input dialog.

`android:voiceLanguage`

String. The spoken language to be expected, expressed as the string value of a constants in `Locale` (such as `"de"` for German or `"fr"` for French). This is needed only if it is different from the current value of `Locale.getDefault()`.

`android:voiceMaxResults`

Integer. Forces the maximum number of results to return, including the "best" result which is always provided as the `ACTION_SEARCH` intent's primary query. Must be 1 or greater. Use `EXTRA_RESULTS` to get the results from the intent. If not provided, the recognizer chooses how many results to return.

`<actionkey>`

Defines a device key and behavior for a search action. A search action provides a special behavior at the touch of a button on the device, based on the current query or focused suggestion. For example, the Contacts application provides a search action to initiate a phone call to the currenly focused contact suggestion at the press of the CALL button.

Not all action keys are available on every device, and not all keys are allowed to be overriden in this way. For example, the "Home" key cannot be used and must always return to the home screen. Also be sure not to define an action key for a key that's needed for typing a search query. This essentially limits the available and reasonable action keys to the call button and menu button. Also note that action keys are not generally discoverable, so you should not provide them as a core user feature.

You must define the `android:keycode` to define the key and at least one of the other three attributes in order to define the search action.

attributes:

`android:keycode`

String. (Required.) A key code from `KeyEvent` that represents the action key you wish to respond to (for example `"KEYCODE_CALL"`). This is added to the `ACTION_SEARCH` intent that is passed to your searchable activity. To examine the key code, use `getIntExtra(SearchManager.ACTION_KEY)`. Not all keys are supported for a search action, as many of them are used for typing, navigation, or system functions.

`android:queryActionMsg`

String. An action message to be sent if the action key is pressed while the user is entering query text. This is added to the `ACTION_SEARCH` intent that the system passes to your searchable activity. To examine the string, use `getStringExtra(SearchManager.ACTION_MSG)`.

`android:suggestActionMsg`

String. An action message to be sent if the action key is pressed while a suggestion is in focus. This is added to the intent that the system passes to your searchable activity (using the action you've defined for the suggestion). To examine the string, use `getStringExtra(SearchManager.ACTION_MSG)`. This should only be used if all your suggestions support this action key. If not all suggestions can handle the same action key, then you must instead use the following `android:suggestActionMsgColumn` attribute.

`android:suggestActionMsgColumn`

String. The name of the column in your content provider that defines the action message for this action key, which is to be sent if the user presses the action key while a suggestion is in focus. This attribute lets you control the action key on a suggestion-by-suggestion basis, because, instead of using the `android:suggestActionMsg` attribute to define the action message for all suggestions, each entry in your content provider provides its own action message.

First, you must define a column in your content provider for each suggestion to provide an action message, then provide the name of that column in this attribute. The system looks at your suggestion cursor, using the string provided here to select your action message column, and then select the action message string from the Cursor. That string is added to the intent that the system passes to your searchable activity (using the action you've defined for suggestions). To examine the string, use `getStringExtra(SearchManager.ACTION_MSG)`. If the data does not exist for the selected suggestion, the action key is ignored.

EXAMPLE:

XML file saved at `res/xml/searchable.xml`:

```
<?xml version="1.0" encoding="utf-8"?>
<searchable xmlns:android="http://schemas.android.com/apk/res/android"
    android:label="@string/search_label"
    android:hint="@string/search_hint"
    android:searchSuggestAuthority="dictionary"
    android:searchSuggestIntentAction="android.intent.action.VIEW"
    android:includeInGlobalSearch="true"
    android:searchSettingsDescription="@string/settings_description" >
</searchable>
```