# Exhibit 1

# THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

# Exhibit 2

# THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

# Exhibit 3

# THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

# Exhibit 4

# Exhibit 5

# THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

# Exhibit 6

## Linkify your Text!

*Posted by Dick Wall, Google Developer Programs*

**Background**: I have long been a fan of desktop wiki note pads. Tomboy on Linux, WikidPad on Windows, and VoodooPad on the Mac are great examples of these. The idea with a wiki note pad is to allow a user to take notes quickly, but also add structure by linking them up with wiki words, and add other, richer information like web URLs, email addresses, and so on. Instead of a list of isolated plain text notes, you get a rich, semantically linked and useful note pad with which to organize ideas.

I decided that something like this would be a good application for Android. Once devices are available, I would like to be able to capture all sorts of information on there, but the poor old note pad application seems to be an afterthought in many mobile devices. Well, not this time!

**Linkify**: The Linkify class in the SDK is perfect for creating a wiki note pad. This class lets you specify a regular expression to match, and a scheme to prepend. The scheme is a string that, when the matched text is added, forms a Content URI to allow the correct data to be looked up.

For example, in our case we want to look for a regular expression match for a WikiWord (that is, a word with camel case and no spaces). Linkify can then turn this into a Content URI -- something like `content://com.google.android.wikinotes.db.wikinotes/wikinotes/WikiWord` which can then be used to locate the correct wiki page from a content provider.

As a bonus, the Linkify class also defines several default matches, in particular it is able to turn web URLs, email addresses and telephone numbers into active links which fire Android intents automatically.

Linkify can be passed any TextView in your application, and will take care of creating the links and enabling their "clickability" for you.

**Default Linkify**: Using the set of default active link options is very straightforward - simply pass it a handle to a TextView with content in it, and the `Linkify.ALL` flag:

```
TextView noteView = (TextView) findViewById(R.id.noteview);
noteView.setText(someContent);
Linkify.addLinks(noteView, Linkify.ALL);
```

and that's it. The Linkify.ALL flag applies all of the predefined link actions, and the TextView will be immediately updated with a set of active links which, if you select them, fire default intents for the actions (e.g. a web URL will start the browser with that URL, a telephone number will bring up the phone dialer with that number ready to call, etc.).

**Custom Linkify:** So what about our WikiWord? There is no pre-defined action for that, so it needs to be defined and associated with a scheme.

The first task is to defined a regular expression that matches the kind of WikiWords we want to find. The regex in this case is:

```
\b[A-Z]+[a-z0-9]+[A-Z][A-Za-z0-9]+\b
```

Obvious no? Well actually this is equivalent to the following description: "Starting with a word boundary (the \b) find at least one upper case letter, followed by at least one lower case letter or a numeric digit, followed by another upper case letter, and then any mix of upper case, lower case or numeric until the next word boundary (the final \b)". Regular expressions are not very pretty, but they are an extremely concise and accurate way of specifying a search pattern.

We also need to tell Linkify what to do with a match to the WikiWord. Linkify will automatically append whatever is matched to a scheme that is supplied to it, so for the sake of argument let's assume we have a ContentProvider that matches the following content URI:

```
content://com.google.android.wikinotes.db.wikinotes/wikinotes/WikiWord
```

The WikiWord part will be appended by Linkify when it finds a match, so we just need the part before that as our scheme.

Now that we have these two things, we use Linkify to connect them up:

```
Pattern wikiWordMatcher = Pattern.compile("\\b[A-Z]+[a-z0-9]+[A-Z][A-Za-z0-
9]+\\b");
String wikiViewURL =
"content://com.google.android.wikinotes.db.wikinotes/wikinotes/";

Linkify.addLinks(noteView, wikiWordMatcher, wikiViewURL);
```

Note that the \b's had to be escaped for the Java Pattern.compile line.

Linkify can be used multiple times on the same view to add more links, so using this after the Default Linkify call means that the existing active links will be maintained and the new WikiWords will be added. You could define more Linkify actions and keep applying them to the same TextView if you wanted to.

Now, if we have a WikiWord in the TextView, let's say **MyTodoList**, Linkify will turn it into an active link with the content URI:
`content://com.google.android.wikinotes.db.wikinotes/wikinotes/MyTodoList` and if you click on it, Android will fire the default intent for that content URI.

For this to all work, you will need a ContentProvider that understands that Content URI, and you will need a default activity capable of doing something with the resulting data. I plan to cover these in future blog entries (and soon). In fact, the whole Wiki Note Pad application is currently undergoing some clean up and review, and will then hopefully be released as a sample application.

# Exhibit 7

# Intentionally Left Blank

# Exhibit 8

# WikiNotes for Android: Routing Intents

In the last article, we talked about using Linkify to turn wiki words (those that match a regular expression we defined) into a content: URI and defining a path to data that matched a note belonging to that wiki word. As an example, a matching word like ToDoList would be turned into a content: URI like content://com.google.android.wikinotes.db.wikinotes/wikinotes/ToDoList and then acted upon using the VIEW action from the Linkify class.

This article will examine how the Android operating system takes this combination of VIEW action and content: URI and finds the correct activity to fire in order to do something with the data. It will also explain how the other default links created by Linkify, like web URLs and telephone numbers, also result in the correct activity to handle that data type being fired. Finally, this article will start to examine the custom ContentProvider that has been created to handle WikiNotes data. The full description of the ContentProvider and what it does will span a couple more articles as well, because there is a lot to cover.

## The Linkify-calls-intent Workflow

At a high level, the steps for Linkify to invoke an intent and for the resulting activity (if any) to handle it looks like this:

1. Linkify is invoked on a TextView to turn matching text patterns into Intent links.
2. Linkify takes over monitoring for those Intent links being selected by the user.
3. When the user selects a link, Linkify calls the VIEW action using the content: URI associated with the link.
4. Android takes the content: URI that represents the data, and looks for a ContentProvider registered in the system that matches the URI.
5. If a match is found, Android queries the ContentProvider using the URI, and asks what MIME type the data that will be returned from the URI is.
6. Android then looks for an activity registered in the system with an intent-filter that matches both the VIEW action, and the MIME type for the data represented by the content: URI.
7. Assuming a match is found, Linkify then invokes the intent for the URI, at which point the activity takes over, and is handed the content: URI.
8. The activity can then use the URI to retrieve the data and act on it.

If this sounds complicated, it really is a simpler process than it sounds, and it is quite lightweight as well. Perhaps a more understandable statement about how it works might be:

Linkify is used to turn matching text into hot-links. When the user selects a hot-link, Android takes the data locator represented by the hot-link and looks for a data handler for

that data locator. If it finds one, it asks for what type of data is returned for that locator. It then looks for something registered with the system that handles that type of data for the VIEW action, and starts it, including the data locator in the request.

The real key here is the MIME type. MIME stands for Multipurpose Internet Mail Extensions - a standard for sending attachments over email. The MIME type (which is the part Android uses) is a way of describing certain kinds of data. That type is then used to look for an Activity that can do something with that data type. In this way, ContentProviders and Activities (or other IntentReceivers) are decoupled, meaning that a given Content URI might have a different ContentProvider to handle it, but could still use the same MIME type meaning that the same activity could be called upon to handle the resulting data.

## Linkify on a Wiki Word

Using the above workflow, let's take a look at exactly how the process works in WikiNotes for Android:

First, Linkify is used to turn text matching the wiki word regular expression into a link that provides a Content URI for that wiki word, for example content://com.google.android.wikinotes.db.wikinotes/wikinotes/ToDoList.

When the user clicks on the wiki word link, Linkify invokes the VIEW action on the Content URI. At this point, the Android system takes over getting the Intent request to the correct activity.

Next, Android looks for a ContentProvider that has been registered with the system to handle URIs matching our Content URI format.

In our case, we have a definition inside our application in the AndroidManifest.xml file that reads:

```
<provider android:name="com.google.android.wikinotes.db.WikiNotesProvider"
        android:authorities="com.google.android.wikinotes.db.wikinotes" />
```

This establishes that we have a ContentProvider defined in our application that provides the "root authority": `com.google.android.wikinotes.db.wikinotes`. This is the first part of the Content URI that we create for a wiki word link. Root Authority is just another way of

thinking about a descriptor that is registered with Android to allow requests for certain URLs to be routed to the correct class.

So, the whole definition is that a class called `com.google.android.wikinotes.db.WikiNotesProvider` is registered with the system as able to handle the `com.google.android.wikinotes.db.wikinotes` root authority (i.e. URIs starting with that identifier).

From here, Android takes the rest of the URI and present it to that ContentProvider. If you look at the WikiNotesProvider class and scroll to the very bottom - the static block there, you can see the pattern definitions to match the rest of the URL.

In particular, take a look at the two lines:

```
URI_MATCHER.addURI(WikiNote.WIKINOTES_AUTHORITY, "wikinotes", NOTES);

URI_MATCHER.addURI(WikiNote.WIKINOTES_AUTHORITY, "wikinotes/*", NOTE_NAME);
```

These are the definitions of URIs that our ContentProvider recognizes and can handle. The first recognizes a full URI of content://com.google.android.wikinotes.db.wikinotes/wikinotes and associates that with a constant called NOTES. This is used elsewhere in the ContentProvider to provide a list of all of the wiki notes in the database when the URI is requested.

The second line uses a wildcard - '*' - to match a request of the form that Linkify will create, e.g. content://com.google.android.wikinotes.db.wikinotes/wikinotes/ToDoList. In this example, the * matches the ToDoList part of the URI and is available to the handler of the request, so that it can fish out the matching note for ToDoList and return it as the data. This also associates that match with a constant called NOTE_NAME, which again is used as an identifier elsewhere in the ContentProvider.

The other matches in this static block are related to forms of searching that have been implemented in the WikiNotes for Android application, and will be covered in later articles. Likewise, how the data is obtained from this matching pattern will be the subject of the next article.

For right now we are concerned with the MIME type for the URI. This is defined in the getType() method also in the WikiNotesProvider class (about half way through the file). Take a quick look at this. The key parts for now are:

```
case NOTES:
    return "vnd.android.cursor.dir/vnd.google.wikinote";
```

and

```
case NOTE_NAME:
    return "vnd.android.cursor.item/vnd.google.wikinote";
```

These are the same constant names we defined in our pattern matchers. In the first case, that of the all notes URI, the MIME type returned is `vnd.android.cursor.dir/vnd.google.wikinote` which is like saying an Android list (dir) of Google wiki notes (the vnd bit is MIME speak for "vendor specific definition"). Likewise, in the case of a NOTE_NAME match, the MIME type returned is `vnd.android.cursor.item/vnd.google.wikinote` which is like saying an Android item of Google wiki notes.

Note that if you define your own MIME data types like this, the `vnd.android.cursor.dir` and `vnd.android.cursor.item` categories should be retained, since they have meaning to the Android system, but the actual item types should be changed to reflect your particular data type.

So far Android has been able to find a ContentProvider that handles the Content URI supplied by the Linkify Intent call, and has queried the ContentProvider to find out the MIME types for that URI. The final step is to find an activity that can handle the VIEW action for that MIME type. Take a look in the the AndroidManifest.xml file again. Inside the WikiNotes activity definition, you will see:

```
<intent-filter>
    <action android:name="android.intent.action.VIEW" />
    <category android:name="android.intent.category.DEFAULT" />
    <category android:name="android.intent.category.BROWSABLE" />
```

```
    <data android:mimeType="vnd.android.cursor.item/vnd.google.wikinote"/>
</intent-filter>
```

This is the correct combination of matches for the VIEW action on a WikiNote type that is requested from the LINKIFY class. The DEFAULT category indicates that the WikiNotes activity should be treated as a default handler (a primary choice) for this kind of data, and the BROWSABLE category means it can be invoked from a "browser", in this case the marked-up Linkified text.

Using this information, Android can match up the VIEW action request for the WikiNotes data type with the WikiNotes activity, and can then use the WikiNotes activity to handle the request.

## Why do it like this?

It's quite a trip through the system, and there is a lot to absorb here, but this is one of the main reasons I wanted to write WikiNotes in the first place. If you follow and understand the steps here, you'll have a good grasp of the whole Intents mechanism in Android, and how it helps loosely coupled activities cooperate to get things done.

In this case, we could have found another way to detect wiki words based on a regular expression, and maybe written our own handler to intercept clicks within the TextView and dig out the right data and display it. This would seem to accomplish the same functionality just as easily as using intents, so what is the advantage to using the full Intents mechanism?

In fact there are several advantages:

The most obvious is that because we are using the standard Intent based approach, we are not limited to just linking and navigating to other wiki notes. We get similar behavior to a number of other data types as well. For example, a telephone number or web URL in a wiki note will be marked up by Linkify, and using this same mechanism (VIEW action on the linked data type) the browser or dialer activities will be automatically fired.

It also means that each operation on a wiki note can be treated as a separate life cycle by our activity. We are not dealing with swapping data in and out of an existing activity - each activity works on a particular wiki note and that's all you have to worry about.

Another advantage is that we now have a public activity to handle VIEW actions in WikiNotes no matter where the request comes from. Another application could request to view a wiki note (perhaps without even knowing what kind of data it is) and our activity could start up and handle it.

The backstack is automatically maintained for you too. As you forward navigate through WikiNotes, Android maintains the history of notes visited, and so when you hit the back button you go back to the last note you were on. All this is free because we rely on the Android intents mechanism.

Finally, if you run WikiNotes for Android and then start DDMS to take a look at the Activity threads in the WikiNotes application while it is running, you can see that despite what you might think, letting Android manage the navigation is very efficient. Create a few linked notes, as many links deep as you like, and then follow them. If you follow links hundreds of notes deep, you will still only see a handful of WikiNotes activities. Android is managing the activities, closing the older ones as necessary and using the life cycle to swap data in and out.

## Next Time

This was a long article, and I apologize for the length. It demonstrates the importance of the Intents mechanism and to reinforce the notion that it should be used whenever possible for forward navigation, even within a single application. Illustrating this is one of the primary reasons I wrote WikiNotes for Android in the first place.

In the next article we will look deeper into the ContentProvider and examine how it turns a Content URI into a row (or several rows) of data that can be used by an activity.

# Exhibit 9

Applicant: Hedloy, Atle

Attorney Docket No.: 3324/106

Serial No.: Not yet assigned

For: Method, System and Computer Readable Medium for Addressing Handling from a Computer Program

Filed: July 22, 2010

# Accelerated Examination Support Document

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

## Table of Contents

Dear Sir:

This accelerated examination support document is provided in support of the petition for accelerated examination filed herewith.

**Listing of Claims**

There are 20 claims currently pending in the application. The claims read as follows:

1. At least one non-transitory computer readable medium encoded with instructions which, when loaded on a computer, establish processes for contact information handling, implemented by a document editing program running in the computer, the processes comprising:

allowing a user to enter textual information into a document using the document editing program;

displaying the textual information in the document electronically using the document editing program;

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed;

following user selection of textual information in the document, analyzing, by the document editing program, the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information and what type or types of contact information the selected textual information is;

providing an input device configured by the document editing program to allow the user to initiate an operation, such operation being of a type depending at least in part on the type or types of contact information of the selected textual information, the operation comprising identifying at least part of the selected textual information to use as a search term in order to find second information, of a specific type or types, associated in an information source with the search term;

after identifying at least part of the selected information to use as a search term, and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation, wherein the operation further comprises:

causing an electronic search in the information source, by an information

management program external to the document editing program, for the search term in order to find whether the search term is included in the information source; and performing an action having a type,

wherein the type of action depends at least in part on whether the search term is included in the information source, and if the search term is so included, and if the information source includes the second information, the action comprises causing insertion of at least part of the second information into the document.

2. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term.

3. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when (i) the information source includes the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document.

4. At least one non-transitory computer readable medium according to claim 3, wherein the instructions further establish processes wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use for insertion into the document.

5. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the selected textual information includes information that is not in the information source, the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source.

6. At least one non-transitory computer readable medium according to claim 5, wherein the instructions further establish processes wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact.

7. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source is available over a network.

8. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source includes an e-mail address and the operation includes causing an e-mail to be sent to the e-mail address.

9. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name.

10. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the document editing program is a spreadsheet program.

11. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the type of operation includes updating the document with information from the information source.

12. A method, for contact information handling, implemented by running a document editing program in a computer, the method comprising:

allowing a user to enter textual information into a document using the document editing program;

displaying the textual information in the document electronically using the document editing program;

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed;

following user selection of textual information in the document, analyzing, by the document editing program, the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information and what type or types of contact information the selected textual information is;

providing an input device configured by the document editing program to allow the user to initiate an operation, such operation being of a type depending at least in part on the type or types of contact information of the selected textual information, the operation comprising identifying at least part of the selected textual information to use as a search term in order to find second information, of a specific type or types, associated in an information source with the search term;

after identifying at least part of the selected information to use as a search term, and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation, wherein the operation further comprises:

causing an electronic search in the information source, by an information management program external to the document editing program, for the search term in order to find whether the search term is included in the information source; and performing an action having a type,

wherein the type of action depends at least in part on whether the search term is included in the information source, and if the search term is so included, and if the information source includes the second information, the action comprises causing insertion of at least part of the second information into the document.

13. A method according to claim 12, wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term; and

when (i) the information source does include the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document.

14. A method according to claim 12, wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use insertion into the document.

15. A method according to claim 12, wherein, when the selected textual information includes information that is not in the information source, and the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source;

wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact.

16. A method according to claim 12, wherein the information source is available over a network.

17. A method according to claim 12, wherein the information source includes an e-mail address and the operation includes causing an e-mail to be sent to the e-mail address.

18. A method according to claim 12, wherein, when the type or types of contact information includes a name, the operation includes causing display by the information

management program of at least part of a contact information record in the information source corresponding to the name.

19. A method according to claim 12, wherein the type of operation includes updating the document with information from the information source.

20. An apparatus for contact information handling, comprising:

a processor; and

a memory storing instructions executable by the processor to perform processes that include:

allowing a user to enter textual information into a document using a document editing program;

displaying the textual information in the document electronically using the document editing program;

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed;

following user selection of textual information in the document, analyzing, by the document editing program, the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information and what type or types of contact information the selected textual information is;

providing an input device configured by the program to allow the user to initiate an operation, such operation being of a type depending at least in part on the type or types of contact information of the selected textual information, the operation comprising identifying at least part of the selected textual information to use as a search term in order to find second information, of a specific type or types, associated in an information source with the search term;

after identifying at least part of the selected information to use as a search term, and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation, wherein the operation further comprises:

causing an electronic search in the information source, by an information management program external to the document editing program, for the search term in order to find whether the search term is included in the information source; and performing an action having a type,

wherein the type of action depends at least in part on whether the search term is included in the information source, and if the search term is so included, and if the information source includes the second information, the action comprises causing insertion of at least part of the second information into the document.

**9(A) References Deemed Most Closely Related:**

An Information Disclosure Statement (IDS) in compliance with 37 C.F.R. 1.98 is filed herewith citing each of the following references deemed most closely related to the subject matter of the claim. The IDS citing the references listed below is submitted in support of the petition for accelerated examination. A separate IDS is filed herewith that is not in support of the petition, but which discloses references for consideration by the examiner.

- Chalas U.S. Patent No. 5,392,386
- Pandit U.S. Patent No. 5,859,636
- Miller et al. U.S. Patent No. 5,946,647
- Ho et al. U.S. Patent No. 6,021,412
- Tso U.S. Patent No. 6,085,201
- Hachamovitch et al. U.S. Patent No. 6,377965
- Bocking et al. U.S. Application Pub. No. 2006/0047644
- Apple Data Detectors User's Manual, Apple Computer Inc., © 1997 ("ADD")
- Bonnie A. Nardi, James R. Miller & David J. Wright, Collaborative, Programmable Intelligent Agents, Communications of the ACM, Vol. 41, No. 3 (Mar. 1998).
- Anind K. Dey, Gregory D. Abowd & Andrew Wood, CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services, Knowledge-Based Systems, Vol. 11 (1998).
- Getting Result with Microsoft Office 97, Microsoft Corporation © 1995-1997
- User Manual for AddressMate and AddressMate Plus, Addressmate Software, © 1994-1995.

**9(B) Identification of Limitations Disclosed by References**

Applicant notes that this identification of limitations in the claims that "are disclosed" by the references is merely a mapping of actual or implied disclosures that are in the prior art or that may appear to be in the prior art. It is a good faith effort to draw the Examiner's attention to questions material to patentability. Mapping of limitations is distinct from patentability. Mapping is provided in order to satisfy the requirements for accelerated examination and to assist the Examiner. While a limitation may be mapped to a passage or element in a reference, a fair reading of that limitation given its broadest reasonable interpretation as would be understood by one of ordinary skill in the art in the context of the claim and the patent specification may not necessarily encompass the mapped passage or element. Whether the claimed invention defines over the prior art will be determined during actual prosecution of this application. Also, this document deals with patentability in section 9(C), including arguments for the inapplicability of some of the claim limitations on mapped counterparts.

***Chalas U.S. Patent No. 5,392,386***

1. At least one non-transitory computer readable medium encoded with instructions which (3:42-53), when loaded on a computer, establish processes for contact information handling (12:51-53), implemented by a document editing program running in the computer (14:57-15:2), the processes comprising:

allowing a user to enter textual information into a document using the document editing program (Fig. 4B);

displaying the textual information in the document electronically using the document editing program (Fig. 4B);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information (7:19-21) while the textual information is displayed (12:21-37);

following user selection of textual information in the document (4:58-67; 7:19-21), analyzing (Fig. 6, step 424; 12:45-68), by the document editing program (14:57-15:2), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (12:51-53, Chalas discloses detecting a zip code) and what type or types of contact information the selected textual information is (12:51-53, Chalas discloses detecting a zip code);

providing an input device (4:58-67; 8:39-44) configured by the document editing program (14:57-15:2) to allow the user to initiate an operation (12:45-68), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Not found), the operation comprising identifying at least part of the selected textual information to use as a search term (12:62-65) in order to find second information, of a specific type or types, associated in an information source with the search term (12:62-65);

after identifying at least part of the selected information to use as a search term (12:62-65), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (5:7-44, 12:45-68), wherein the operation further comprises:

causing an electronic search in the information source (12:62-65, 13:46-56), by an information management program external to the document editing

program (12:65-68, 13:46-56 ), for the search term in order to find whether the search term is included in the information source (12:62-65); and performing an action having a type (12:45-68),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (12:57-60).

2. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Not found).

3. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when (i) the information source includes the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Not found).

4. At least one non-transitory computer readable medium according to claim 3, wherein the instructions further establish processes wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use for insertion into the document (Not found).

5. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the selected textual information includes information that is not in the information source, the operation

further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Not found).

6. At least one non-transitory computer readable medium according to claim 5, wherein the instructions further establish processes wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

7. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source is available over a network (Not found).

8. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source includes an e-mail address (Not found) and the operation includes causing an e-mail to be sent to the e-mail address (Not found).

9. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Not found).

10. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the document editing program is a spreadsheet program (Not found).

11. At least one non-transitory computer readable medium according to claim 1,

wherein the instructions further establish processes wherein the type of operation includes updating the document with information from the information source (Not found).

12. A method, for contact information handling (12:51-53), implemented by running a document editing program in a computer (14:57-15:2), the method comprising:

allowing a user to enter textual information into a document using the document editing program (Fig. 4B);

displaying the textual information in the document electronically using the document editing program (Fig. 4B);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information (7:19-21) while the textual information is displayed (12:21-37);

following user selection of textual information in the document (4:58-67; 7:19-21), analyzing(Fig. 6, step 424; 12:45-68), by the document editing program (14:57-15:2), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (12:51-53, Chalas discloses detecting a zip code) and what type or types of contact information the selected textual information is (12:51-53, Chalas discloses detecting a zip code);

providing an input device (4:58-67; 8:39-44) configured by the document editing program (14:57-15:2) to allow the user to initiate an operation (12:45-68), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Not found), the operation comprising identifying at least part of the selected textual information to use as a search term (12:62-65) in order to find second information, of a specific type or types, associated in an information source with the search term (12:62-65);

after identifying at least part of the selected information to use as a search term (12:62-65), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (5:7-44, 12:45-68), wherein the operation further comprises:

causing an electronic search in the information source (12:62-65, 13:46-

56), by an information management program external to the document editing program (12:65-68, 13:46-56 ), for the search term in order to find whether the search term is included in the information source (12:62-65); and performing an action having a type (12:45-68),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (12:57-60).

13. A method according to claim 12, wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term; and

when (i) the information source does include the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Not found).

14. A method according to claim 12, wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use insertion into the document (Not found).

15. A method according to claim 12, wherein, when the selected textual information includes information that is not in the information source, and the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source;

wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage

comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

16. A method according to claim 12, wherein the information source is available over a network (Not found).

17. A method according to claim 12, wherein the information source includes an e-mail address and the operation includes causing an e-mail to be sent to the e-mail address (Not found).

18. A method according to claim 12, wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Not found).

19. A method according to claim 12, wherein the type of operation includes updating the document with information from the information source (Not found).

20. An apparatus for contact information handling (12:51-53), comprising:

a processor (2:50-57); and

a memory (2:50-57) storing instructions executable by the processor to perform processes that include:

allowing a user to enter textual information into a document using the document editing program (Fig. 4B);

displaying the textual information in the document electronically using the document editing program (Fig. 4B);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information (7:19-21) while the textual information is displayed (12:21-37);

following user selection of textual information in the document (4:58-67; 7:19-21), analyzing(Fig. 6, step 424; 12:45-68), by the document editing program

(14:57-15:2), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (12:51-53, Chalas discloses detecting a zip code) and what type or types of contact information the selected textual information is (12:51-53, Chalas discloses detecting a zip code);

providing an input device (4:58-67; 8:39-44) configured by the document editing program (14:57-15:2) to allow the user to initiate an operation (12:45-68), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Not found), the operation comprising identifying at least part of the selected textual information to use as a search term (12:62-65) in order to find second information, of a specific type or types, associated in an information source with the search term (12:62-65);

after identifying at least part of the selected information to use as a search term (12:62-65), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (5:7-44, 12:45-68), wherein the operation further comprises:

causing an electronic search in the information source (12:62-65, 13:46-56), by an information management program external to the document editing program (12:65-68, 13:46-56 ), for the search term in order to find whether the search term is included in the information source (12:62-65); and performing an action having a type (12:45-68),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (12:57-60).

***Pandit*** ***U.S. Patent No. 5,859,636***

1. At least one non-transitory computer readable medium encoded with instructions which (5:27-34), when loaded on a computer, establish processes for contact information handling (1:42-48), implemented by a document editing program running in the computer (5:17-21), the processes comprising:

allowing a user to enter textual information into a document using the document editing program (5:18-21);

displaying the textual information in the document electronically using the document editing program (Figs. 1a-f show displayed text; 5:18-21);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (2:4-8; fig 1a: 11: fig. 2:21);

following user selection of textual information in the document, analyzing, by the document editing program (5:17-21), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information and what type or types of contact information the selected textual information is (3:50-59; 4:24-31; Pandit discloses recognizing dates, telephone numbers, and phone numbers);

providing an input device configured by the document editing program to allow the user to initiate an operation (Figs. 1a, 11; 1b, 18; 1c, 14; 1d, 19; 1e, 16; 1f, 20; Fig. 2, 23; 2:5-23; Pandit discloses using a menu bar and a menu), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (2:41-50; 2:56-63; 3:1-8; Pandit discloses a menu with different operations for dates, e-mail, and telephone numbers), the operation comprising identifying at least part of the selected textual information to use as a search term in order to find second information, of a specific type or types, associated in an information source with the search term (Not found);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (2:41-50; 2:56-63; 3:1-8; Pandit discloses, for example, dialing a phone number and preparing an e-mail

template based on the accented text from the document), wherein the operation further comprises:

causing an electronic search in the information source (Not found), by an information management program external to the document editing program (2:41-50; 2:56-63; 3:1-8; Pandit discloses opening programs such as a scheduling program), for the search term in order to find whether the search term is included in the information source (Not found); and performing an action having a type (2:41-50; 2:56-63; 3:1-8 Pandit discloses opening programs such as a scheduling program),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (Not found).

2. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Not found).

3. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when (i) the information source includes the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Not found).

4. At least one non-transitory computer readable medium according to claim 3, wherein the instructions further establish processes wherein, when the second information includes a plurality of addresses, the operation further comprises allowing

the user to choose one of the plurality of addresses to use for insertion into the document (Not found).

5. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the selected textual information includes information that is not in the information source (Not found), the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Fig. 1d, 19; 2;56-63; Pandit discloses adding an e-mail to an address book).

6. At least one non-transitory computer readable medium according to claim 5, wherein the instructions further establish processes wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

7. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source is available over a network (Not found).

8. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source includes an e-mail address (Not found, Figs. 1c and 1d; 2:51-63; Pandit discloses an address book, but the address book is not used as an information source in a search based on the accented text) and the operation includes causing an e-mail to be sent to the e-mail address (Not found, Figs. 1c and 1d; 2:51-63; Pandit discloses preparing an e-mail template for the recognized e-mail within the text, not second information associated with the recognized text).

9. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Not found).

10. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the document editing program is a spreadsheet program (Not found).

11. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the type of operation includes updating the document with information from the information source (Not found).

12. A method, for contact information handling (1:42-48), implemented by running a document editing program in a computer, the method comprising:

allowing a user to enter textual information into a document using the document editing program (5:18-21);

displaying the textual information in the document electronically using the document editing program (Figs. 1a-f show displayed text; 5:18-21);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (2:4-8; fig 1a: 11: fig. 2:21);

following user selection of textual information in the document, analyzing, by the document editing program (5:17-21), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information and what type or types of contact information the selected textual information is (3:50-59; 4:24-31; Pandit discloses recognizing dates, telephone numbers, and phone numbers);

providing an input device configured by the document editing program to allow

the user to initiate an operation (Figs. 1a, 11; 1b, 18; 1c, 14; 1d, 19; 1e, 16; 1f, 20; Fig. 2, 23; 2:5-23; Pandit discloses using a menu bar and a menu), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (2:41-50; 2:56-63; 3:1-8; Pandit discloses a menu with different operations for dates, e-mail, and telephone numbers), the operation comprising identifying at least part of the selected textual information to use as a search term in order to find second information, of a specific type or types, associated in an information source with the search term (Not found);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (2:41-50; 2:56-63; 3:1-8; Pandit discloses, for example, dialing a phone number and preparing an e-mail template based on the accented text from the document), wherein the operation further comprises:

causing an electronic search in the information source (Not found), by an information management program external to the document editing program (2:41-50; 2:56-63; 3:1-8; Pandit discloses opening programs such as a scheduling program), for the search term in order to find whether the search term is included in the information source (Not found); and performing an action having a type (2:41-50; 2:56-63; 3:1-8 Pandit discloses opening programs such as a scheduling program),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (Not found).

13. A method according to claim 12, wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Not found); and

when (i) the information source does include the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Not found).

14. A method according to claim 12, wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use insertion into the document (Not found).

15. A method according to claim 12, wherein, when the selected textual information includes information that is not in the information source, and the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Fig. 1d, 19; 2;56-63; Pandit discloses adding an e-mail to an address book);

wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

16. A method according to claim 12, wherein the information source is available over a network (Not found).

17. A method according to claim 12, wherein the information source includes an e-mail address (Not found, Figs. 1c and 1d; 2:51-63; Pandit discloses an address book, but the address book is not used as an information source in a search based on the accented text) and the operation includes causing an e-mail to be sent to the e-mail address (Not found, Figs. 1c and 1d; 2:51-63; Pandit discloses preparing an e-mail template for the recognized e-mail within the text, not second information associated with the recognized text).

18. A method according to claim 12, wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Not found).

19. A method according to claim 12, wherein the type of operation includes updating the document with information from the information source (Not found).

20. An apparatus for contact information handling (1:42-48), comprising:

a processor (5:25-45); and

a memory storing (5:25-45) instructions executable by the processor to perform processes that include:

allowing a user to enter textual information into a document using the document editing program (5:18-21);

displaying the textual information in the document electronically using the document editing program (Figs. 1a-f show displayed text; 5:18-21);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (2:4-8; fig 1a: 11: fig. 2:21);

following user selection of textual information in the document, analyzing, by the document editing program (5:17-21), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information and what type or types of contact information the selected textual information is (3:50-59; 4:24-31; Pandit discloses recognizing dates, telephone numbers, and phone numbers);

providing an input device configured by the document editing program to allow the user to initiate an operation (Figs. 1a, 11; 1b, 18; 1c, 14; 1d, 19; 1e, 16; 1f, 20; Fig. 2, 23; 2:5-23; Pandit discloses using a menu bar and a menu), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (2:41-50; 2:56-63; 3:1-8; Pandit discloses a menu with different operations for dates, e-mail, and telephone

numbers), the operation comprising identifying at least part of the selected textual information to use as a search term in order to find second information, of a specific type or types, associated in an information source with the search term (Not found);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (2:41-50; 2:56-63; 3:1-8; Pandit discloses, for example, dialing a phone number and preparing an e-mail template based on the accented text from the document), wherein the operation further comprises:

causing an electronic search in the information source (Not found), by an information management program external to the document editing program (2:41-50; 2:56-63; 3:1-8; Pandit discloses opening programs such as a scheduling program), for the search term in order to find whether the search term is included in the information source (Not found); and performing an action having a type (2:41-50; 2:56-63; 3:1-8 Pandit discloses opening programs such as a scheduling program),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (Not found).

***Miller et al. U.S. Patent No. 5,946,647***

1. At least one non-transitory computer readable medium encoded with instructions which (3:38-44), when loaded on a computer, establish processes for contact information handling (Abstract; 5:6-37), implemented by a document editing program running in the computer (Not found), the processes comprising:

allowing a user to enter textual information into a document using the document editing program (Not found);

displaying the textual information in the document electronically (Fig. 5; 3:36-38) using the document editing program (Not found);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (Not found);

following user selection of textual information in the document (Not found), analyzing (Fig. 6; Fig. 8, 820; 3:61-64; 5:6-18), by the document editing program (Not found, Fig. 1, 165, Miller discloses analysis using an analyzer server), the selected textual information (Not found) to determine if the selected textual information is regarded by the document editing program as contact information (5:6-37) and what type or types of contact information the selected textual information is (5:6-37);

providing an input device configured by the document editing program to allow the user to initiate an operation (Fig. 7, 710; 3:65 – 4:17; 4:27-31; 5:38-50), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (5:38-50), the operation comprising identifying at least part of the selected textual information to use as a search term (Not found) in order to find second information, of a specific type or types, associated in an information source (5:44-50; Miller discloses an electronic telephone book) with the search term (Not found);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (5:38-50, Miller discloses dialing a telephone number or putting the number from the document into an electronic telephone book), wherein the operation further comprises:

causing an electronic search in the information source (Not found), by an information management program external to the document editing program (Not found), for the search term in order to find whether the search term is included in the information source (Not found); and performing an action having a type (Not found),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (Not found).

2. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Not found).

3. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when (i) the information source includes the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Not found).

4. At least one non-transitory computer readable medium according to claim 3, wherein the instructions further establish processes wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use for insertion into the document (Not found).

5. At least one non-transitory computer readable medium according to claim 1,

wherein the instructions further establish processes wherein, when the selected textual information (Not found) includes information that is not in the information source (Not found), the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (5:44-50, Miller discloses adding a name to an electronic telephone book, but not when the selected textual information includes information that is not in the information source).

6. At least one non-transitory computer readable medium according to claim 5, wherein the instructions further establish processes wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

7. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source is available over a network (Not found).

8. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source includes (Not found) an e-mail address and the operation includes causing an e-mail to be sent (2:34-41) to the e-mail address (Not found).

9. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Not found).

10. At least one non-transitory computer readable medium according to claim 1,

wherein the instructions further establish processes wherein the document editing program is a spreadsheet program (Not found).

11. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the type of operation includes updating the document with information from the information source (Not found).

12. A method, for contact information handling (Abstract; 5:6-37), implemented by running a document editing program in a computer (Not found), the method comprising:

allowing a user to enter textual information into a document using the document editing program (Not found);

displaying the textual information in the document electronically (Fig. 5; 3:36-38) using the document editing program (Not found);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (Not found);

following user selection of textual information in the document (Not found), analyzing (Fig. 6; Fig. 8, 820; 3:61-64; 5:6-18), by the document editing program (Not found, Fig. 1, 165, Miller discloses analysis using an analyzer server), the selected textual information (Not found) to determine if the selected textual information is regarded by the document editing program as contact information (5:6-37) and what type or types of contact information the selected textual information is (5:6-37);

providing an input device configured by the document editing program to allow the user to initiate an operation (Fig. 7, 710; 3:65 – 4:17; 4:27-31; 5:38-50), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (5:38-50), the operation comprising identifying at least part of the selected textual information to use as a search term (Not found) in order to find second information, of a specific type or types, associated in an information source (5:44-50; Miller discloses an electronic telephone book) with the search term (Not found);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (5:38-50, Miller discloses dialing a telephone number or putting the number from the document into an electronic telephone book), wherein the operation further comprises:

causing an electronic search in the information source (Not found), by an information management program external to the document editing program (Not found), for the search term in order to find whether the search term is included in the information source (Not found); and performing an action having a type (Not found),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (Not found).

13. A method according to claim 12, wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Not found); and

when (i) the information source does include the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Not found).

14. A method according to claim 12, wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use insertion into the document (Not found).

15. A method according to claim 12, wherein, when the selected textual information

includes information that is not in the information source, and the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (5:44-50, Miller discloses adding a name to an electronic telephone book, but not when the selected textual information includes information that is not in the information source);

wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

16. A method according to claim 12, wherein the information source is available over a network (Not found).

17. A method according to claim 12, wherein the information source includes an e-mail address and the operation includes (Not found) an e-mail address and the operation includes causing an e-mail to be sent (2:34-41) to the e-mail address (Not found)..

18. A method according to claim 12, wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Not found).

19. A method according to claim 12, wherein the type of operation includes updating the document with information from the information source (Not found).

20. An apparatus for contact information handling (Abstract; 5:6-37), comprising:

a processor (3:22-33); and

a memory storing (3:22-33) instructions executable by the processor to perform processes that include:

allowing a user to enter textual information into a document using the

document editing program (Not found);

displaying the textual information in the document electronically (Fig. 5; 3:36-38) using the document editing program (Not found);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (Not found);

following user selection of textual information in the document (Not found), analyzing (Fig. 6; Fig. 8, 820; 3:61-64; 5:6-18), by the document editing program (Not found, Fig. 1, 165, Miller discloses analysis using an analyzer server), the selected textual information (Not found) to determine if the selected textual information is regarded by the document editing program as contact information (5:6-37) and what type or types of contact information the selected textual information is (5:6-37);

providing an input device configured by the document editing program to allow the user to initiate an operation (Fig. 7, 710; 3:65 – 4:17; 4:27-31; 5:38-50), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (5:38-50), the operation comprising identifying at least part of the selected textual information to use as a search term (Not found) in order to find second information, of a specific type or types, associated in an information source (5:44-50; Miller discloses an electronic telephone book) with the search term (Not found);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (5:38-50, Miller discloses dialing a telephone number or putting the number from the document into an electronic telephone book), wherein the operation further comprises:

causing an electronic search in the information source (Not found), by an information management program external to the document editing program (Not found), for the search term in order to find whether the search term is included in the information source (Not found); and

performing an action having a type (Not found),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (Not found).

***Ho et al.*** ***U.S. Patent No. 6,021,412***

1. At least one non-transitory computer readable medium encoded with instructions which (4:7-9), when loaded on a computer, establish processes for contact information handling (Not found), implemented by a document editing program running in the computer (4:10-13), the processes comprising:

allowing a user to enter textual information into a document using the document editing program (Fig. 3; 4:31-37);

displaying the textual information in the document electronically using the document editing program (Fig. 3; 4:31-37);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (Not found);

following user selection of textual information in the document (Not found), analyzing (Fig. 5, step 501; 3:40 – 4:2; 5:14-16), by the document editing program (4:10-13), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (Not found) and what type or types of contact information the selected textual information is (Not found);

providing an input device configured by the document editing program to allow the user to initiate an operation (Fig. 4; Fig. 5, step 503; 4:40-49; 5:34-37), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Not found), the operation comprising identifying at least part of the selected textual information to use as a search term (6:7-12) in order to find second information, of a specific type or types, associated in an information source with the search term (Fig. 5, step 506; 6:7-12);

after identifying at least part of the selected information (Not found) to use as a search term (6:7-12), and in consequence of receipt by the document editing program of an execute command from the input device (Fig. 4, 430, 431; 4:42-44), performing the operation (Fig. 5, step 506; 6:7-12), wherein the operation further comprises:

causing an electronic search in the information source (Fig. 5, step 506; 6:7-12), by an information management program external to the document editing program (Not found), for the search term (Not found) in order to find whether the search term is included in the information source (Not found); and performing an action having a type (Fig. 5, step 507; 6:13-17),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included (Not found), and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (Fig. 5, step 507; 6:13-17).

2. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Not found).

3. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when (i) the information source includes the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Not found).

4. At least one non-transitory computer readable medium according to claim 3, wherein the instructions further establish processes wherein, when the second information includes a plurality of addresses, the operation further comprises allowing

the user to choose one of the plurality of addresses to use for insertion into the document (6:21-34; Ho discloses selecting from multiple images for insertion).

5. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the selected textual information includes information that is not in the information source, the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Not found).

6. At least one non-transitory computer readable medium according to claim 5, wherein the instructions further establish processes wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

7. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source is available over a network (Not found).

8. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source includes an e-mail address (Not found) and the operation includes causing an e-mail to be sent to the e-mail address (Not found).

9. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Not found).

10. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the document editing program is a spreadsheet program (Not found).

11. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the type of operation includes updating the document with information from the information source (Not found).

12. A method, for contact information handling (Not found), implemented by running a document editing program in a computer (4:10-13), the method comprising:

allowing a user to enter textual information into a document using the document editing program (Fig. 3; 4:31-37);

displaying the textual information in the document electronically using the document editing program (Fig. 3; 4:31-37);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (Not found);

following user selection of textual information in the document (Not found), analyzing (Fig. 5, step 501; 3:40 – 4:2; 5:14-16), by the document editing program (4:10-13), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (Not found) and what type or types of contact information the selected textual information is (Not found);

providing an input device configured by the document editing program to allow the user to initiate an operation (Fig. 4; Fig. 5, step 503; 4:40-49; 5:34-37), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Not found), the operation comprising identifying at least part of the selected textual information to use as a search term (6:7-12) in order to find second information, of a specific type or types, associated in an information source with the search term (Fig. 5, step 506; 6:7-12);

after identifying at least part of the selected information (Not found) to use as a search term (6:7-12), and in consequence of receipt by the document editing program of an execute command from the input device (Fig. 4, 430, 431; 4:42-44), performing the operation (Fig. 5, step 506; 6:7-12), wherein the operation further comprises:

causing an electronic search in the information source (Fig. 5, step 506; 6:7-12), by an information management program external to the document editing program (Not found), for the search term (Not found) in order to find whether the search term is included in the information source (Not found); and performing an action having a type (Fig. 5, step 507; 6:13-17),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included (Not found), and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (Fig. 5, step 507; 6:13-17).

13. A method according to claim 12, wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Not found); and

when (i) the information source does include the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Not found).

14. A method according to claim 12, wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use insertion into the document (6:21-34; Ho discloses selecting from multiple images for insertion).

15. A method according to claim 12, wherein, when the selected textual information

includes information that is not in the information source, and the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Not found);

wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

16. A method according to claim 12, wherein the information source is available over a network (Not found).

17. A method according to claim 12, wherein the information source includes an e-mail address (Not found) and the operation includes causing an e-mail to be sent to the e-mail address (Not found).

18. A method according to claim 12, wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Not found).

19. A method according to claim 12, wherein the type of operation includes updating the document with information from the information source (Not found).

20. An apparatus for contact information handling (Not found), comprising:

a processor (4:3-10); and

a memory (4:3-10) storing instructions executable by the processor to perform processes that include:

allowing a user to enter textual information into a document using the document editing program (Fig. 3; 4:31-37);

displaying the textual information in the document electronically using the

document editing program (Fig. 3; 4:31-37);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (Not found);

following user selection of textual information in the document (Not found), analyzing (Fig. 5, step 501; 3:40 – 4:2; 5:14-16), by the document editing program (4:10-13), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (Not found) and what type or types of contact information the selected textual information is (Not found);

providing an input device configured by the document editing program to allow the user to initiate an operation (Fig. 4; Fig. 5, step 503; 4:40-49; 5:34-37), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Not found), the operation comprising identifying at least part of the selected textual information to use as a search term (6:7-12) in order to find second information, of a specific type or types, associated in an information source with the search term (Fig. 5, step 506; 6:7-12);

after identifying at least part of the selected information (Not found) to use as a search term (6:7-12), and in consequence of receipt by the document editing program of an execute command from the input device (Fig. 4, 430, 431; 4:42-44), performing the operation (Fig. 5, step 506; 6:7-12), wherein the operation further comprises:

causing an electronic search in the information source (Fig. 5, step 506; 6:7-12), by an information management program external to the document editing program (Not found), for the search term (Not found) in order to find whether the search term is included in the information source (Not found); and performing an action having a type (Fig. 5, step 507; 6:13-17),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the

search term is so included (Not found), and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (Fig. 5, step 507; 6:13-17).

***Tso*** ***U.S. Patent No. 6,085,201***

1. At least one non-transitory computer readable medium encoded with instructions which (7:64–8:4), when loaded on a computer, establish processes for contact information handling (Not found), implemented by a document editing program running in the computer (4:25-31, Tso discloses an e-mail application), the processes comprising:

allowing a user to enter textual information into a document using the document editing program (Not found);

displaying the textual information in the document electronically using the document editing program (4:32-47);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (4:32-47);

following user selection of textual information in the document(4:32-47), analyzing (4:48-67), by the document editing program (4:25-31; Tso discloses an e-mail application), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (Not found) and what type or types of contact information the selected textual information is (Not found);

providing an input device configured by the document editing program to allow the user to initiate an operation (4:43-47; Tso discloses invoking the template engine), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Not found), the operation comprising identifying at least part of the selected textual information to use as a search term (5:4-6) in order to find second information, of a specific type or types, associated in an information source with the search term (5:7-11);

after identifying at least part of the selected information to use as a search term

(5:4-6), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (4:43-47, Tso discloses invoking the template engine), wherein the operation further comprises:

causing an electronic search in the information source (5:7-11), by an information management program (3:21-28; Tso discloses a template engine) external to the document editing program (Not found), for the search term in order to find whether the search term is included in the information source (5:18-22); and performing an action having a type (5:42-44),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (5:42-44), the action comprises causing insertion of at least part of the second information (5:42-44) into the document (Not found).

2. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Not found).

3. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when (i) the information source includes the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Not found).

4. At least one non-transitory computer readable medium according to claim 3, wherein the instructions further establish processes wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use for insertion into the document

(Not found).

5. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the selected textual information includes information that is not in the information source, the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Not found).

6. At least one non-transitory computer readable medium according to claim 5, wherein the instructions further establish processes wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

7. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source is available over a network (8:5-14).

8. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source includes an e-mail address (Not found) and the operation includes causing an e-mail to be sent (4:32-34; Tso discloses generating e-mail message) to the e-mail address (Not found).

9. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Not found).

10. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the document editing program is a spreadsheet program (Not found).

11. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the type of operation includes updating the document with information from the information source (Not found).

12. A method, for contact information handling (Not found), implemented by running a document editing program in a computer (4:25-31; Tso discloses an e-mail application), the method comprising:

allowing a user to enter textual information into a document using the document editing program (Not found);

displaying the textual information in the document electronically using the document editing program (4:32-47);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (4:32-47);

following user selection of textual information in the document(4:32-47), analyzing (4:48-67), by the document editing program (4:25-31; Tso discloses an e-mail application), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (Not found) and what type or types of contact information the selected textual information is (Not found);

providing an input device configured by the document editing program to allow the user to initiate an operation (4:43-47; Tso discloses invoking the template engine), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Not found), the operation comprising identifying at least part of the selected textual information to use as a search term (5:4-6)

in order to find second information, of a specific type or types, associated in an information source with the search term (5:7-11);

after identifying at least part of the selected information to use as a search term (5:4-6), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (4:43-47, Tso discloses invoking the template engine), wherein the operation further comprises:

causing an electronic search in the information source (5:7-11), by an information management program (3:21-28; Tso discloses a template engine) external to the document editing program (Not found), for the search term in order to find whether the search term is included in the information source (5:18-22); and performing an action having a type (5:42-44),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (5:42-44), the action comprises causing insertion of at least part of the second information (5:42-44) into the document (Not found).

13. A method according to claim 12, wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Not found); and

when (i) the information source does include the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Not found).

14. A method according to claim 12, wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use insertion into the document (Not found).

15. A method according to claim 12, wherein, when the selected textual information includes information that is not in the information source, and the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Not found);

wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

16. A method according to claim 12, wherein the information source is available over a network (8:5-14).

17. A method according to claim 12, wherein the information source includes an e-mail address (Not found) and the operation includes causing an e-mail to be sent (4:32-34; Tso discloses generating e-mail message) to the e-mail address (Not found).

18. A method according to claim 12, wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Not found).

19. A method according to claim 12, wherein the type of operation includes updating the document with information from the information source (Not found).

20. An apparatus for contact information handling (Not found), comprising:

a processor (8:51); and

a memory (5:47) storing instructions executable by the processor to perform processes that include:

allowing a user to enter textual information into a document using the document editing program (Not found);

displaying the textual information in the document electronically using the document editing program (4:32-47);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (4:32-47);

following user selection of textual information in the document(4:32-47), analyzing (4:48-67), by the document editing program (4:25-31; Tso discloses an e-mail application), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (Not found) and what type or types of contact information the selected textual information is (Not found);

providing an input device configured by the document editing program to allow the user to initiate an operation (4:43-47; Tso discloses invoking the template engine), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Not found), the operation comprising identifying at least part of the selected textual information to use as a search term (5:4-6) in order to find second information, of a specific type or types, associated in an information source with the search term (5:7-11);

after identifying at least part of the selected information to use as a search term (5:4-6), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (4:43-47, Tso discloses invoking the template engine), wherein the operation further comprises:

causing an electronic search in the information source (5:7-11), by an information management program (3:21-28; Tso discloses a template engine) external to the document editing program (Not found), for the search term in order to find whether the search term is included in the information source (5:18-22); and performing an action having a type (5:42-44),

wherein the type of action depends at least in part on whether the

search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (5:42-44), the action comprises causing insertion of at least part of the second information (5:42-44) into the document (Not found).

***Hachamovitch et al.*** ***U.S. Patent No. 6,377,965***

1. At least one non-transitory computer readable medium encoded with instructions which (4:53-55), when loaded on a computer, establish processes for contact information handling (Not found), implemented by a document editing program running in the computer (9:44-52), the processes comprising:

allowing a user to enter textual information into a document using the document editing program (Figs 2A, 2B, 2C, and 4A);

displaying the textual information in the document electronically using the document editing program (Figs 2A, 2B, 2C, and 4A);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (Fig. 4A; 12:37-45; Fig. 7, step 706; 16:31-36; 11:14-35, Hachamovitch discloses the user entering text into a structured data field);

following user selection of textual information in the document (Not found), analyzing, by the document editing program, the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (Not found) and what type or types of contact information the selected textual information is (Not found);

providing an input device configured by the document editing program to allow the user to initiate an operation (15:55-59), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Not found), the operation comprising identifying at least part of the selected textual information to use as a search term (Not found, The search, Fig. 5, step 506, happens before receipt of an execute command from the input device) in order to find second information, of a specific type or types, associated in an information source with the

search term (Not found, The search, Fig. 5, step 506, happens before receipt of an execute command from the input device);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program of an execute command from the input device (Fig. 5, step 516; 15:55-59), performing the operation (Fig. 5, 518; 15:62-67), wherein the operation further comprises:

causing an electronic search in the information source (Not found, The search, Fig. 5, step 506, happens before receipt of an execute command from the input device), by an information management program (Not found) external to the document editing program (Not found), for the search term in order to find whether the search term is included in the information source (Not found, The search, Fig. 5, step 506, happens before receipt of an execute command from the input device); and performing an action having a type (Not found),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (Fig. 5, 518; 15:62-67), the action comprises causing insertion of at least part of the second information (Fig. 5, 518; 15:62-67) into the document (Fig. 5, 518; 15:62-67).

2. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Not found).

3. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when (i) the information source includes the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further

comprises causing insertion of at least part of the address into the document (Not found).

4. At least one non-transitory computer readable medium according to claim 3, wherein the instructions further establish processes wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use for insertion into the document (Not found).

5. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the selected textual information includes information that is not in the information source, the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Not found).

6. At least one non-transitory computer readable medium according to claim 5, wherein the instructions further establish processes wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

7. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source is available over a network (Not found).

8. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source includes an e-mail address (11:30-35) and the operation includes causing an e-mail to be sent to the e-mail address (Not found).

9. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Not found).

10. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the document editing program is a spreadsheet program (Not found).

11. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the type of operation includes updating the document with information from the information source (Not found).

12. A method, for contact information handling (Not found), implemented by running a document editing program in a computer (9:44-52), the method comprising:

allowing a user to enter textual information into a document using the document editing program (Figs 2A, 2B, 2C, and 4A);

displaying the textual information in the document electronically using the document editing program (Figs 2A, 2B, 2C, and 4A);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (Fig. 4A; 12:37-45; Fig. 7, step 706; 16:31-36; 11:14-35, Hachamovitch discloses the user entering text into a structured data field);

following user selection of textual information in the document (Not found), analyzing, by the document editing program, the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (Not found) and what type or types of contact information the selected textual information is (Not found);

providing an input device configured by the document editing program to allow

the user to initiate an operation (15:55-59), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Not found), the operation comprising identifying at least part of the selected textual information to use as a search term (Not found, The search, Fig. 5, step 506, happens before receipt of an execute command from the input device) in order to find second information, of a specific type or types, associated in an information source with the search term (Not found, The search, Fig. 5, step 506, happens before receipt of an execute command from the input device);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program of an execute command from the input device (Fig. 5, step 516; 15:55-59), performing the operation (Fig. 5, 518; 15:62-67), wherein the operation further comprises:

causing an electronic search in the information source (Not found, The search, Fig. 5, step 506, happens before receipt of an execute command from the input device), by an information management program (Not found) external to the document editing program (Not found), for the search term in order to find whether the search term is included in the information source (Not found, The search, Fig. 5, step 506, happens before receipt of an execute command from the input device); and performing an action having a type (Not found),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (Fig. 5, 518; 15:62-67), the action comprises causing insertion of at least part of the second information (Fig. 5, 518; 15:62-67) into the document (Fig. 5, 518; 15:62-67).

13. A method according to claim 12, wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Not found); and

when (i) the information source does include the search term, (ii) the selected

textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Not found).

14. A method according to claim 12, wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use insertion into the document (Not found).

15. A method according to claim 12, wherein, when the selected textual information includes information that is not in the information source, and the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Not found);

wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

16. A method according to claim 12, wherein the information source is available over a network (Not found).

17. A method according to claim 12, wherein the information source includes an e-mail address (11:30-35) and the operation includes causing an e-mail to be sent to the e-mail address (Not found).

18. A method according to claim 12, wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Not found).

19. A method according to claim 12, wherein the type of operation includes updating the document with information from the information source (Not found).

20. An apparatus for contact information handling, comprising:

a processor (8:29-35, 41-50); and

a memory (4:53-55; 8:41-50) storing instructions executable by the processor to perform processes that include:

allowing a user to enter textual information into a document using the document editing program (Figs 2A, 2B, 2C, and 4A);

displaying the textual information in the document electronically using the document editing program (Figs 2A, 2B, 2C, and 4A);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (Fig. 4A; 12:37-45; Fig. 7, step 706; 16:31-36; 11:14-35, Hachamovitch discloses the user entering text into a structured data field);

following user selection of textual information in the document (Not found), analyzing, by the document editing program, the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (Not found) and what type or types of contact information the selected textual information is (Not found);

providing an input device configured by the document editing program to allow the user to initiate an operation (15:55-59), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Not found), the operation comprising identifying at least part of the selected textual information to use as a search term (Not found, The search, Fig. 5, step 506, happens before receipt of an execute command from the input device) in order to find second information, of a specific type or types, associated in an information source with the search term (Not found, The search, Fig. 5, step 506, happens before receipt of an execute command from the input device);

after identifying at least part of the selected information to use as a search

term (Not found), and in consequence of receipt by the document editing program of an execute command from the input device (Fig. 5, step 516; 15:55-59), performing the operation (Fig. 5, 518; 15:62-67), wherein the operation further comprises:

causing an electronic search in the information source (Not found, The search, Fig. 5, step 506, happens before receipt of an execute command from the input device), by an information management program (Not found) external to the document editing program (Not found), for the search term in order to find whether the search term is included in the information source (Not found, The search, Fig. 5, step 506, happens before receipt of an execute command from the input device); and performing an action having a type (Not found),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (Fig. 5, 518; 15:62-67), the action comprises causing insertion of at least part of the second information (Fig. 5, 518; 15:62-67) into the document (Fig. 5, 518; 15:62-67).

***Bocking et al. U.S. Application Pub. No. 2006/0047644***

1. At least one non-transitory computer readable medium encoded with instructions which ([0050]), when loaded on a computer, establish processes for contact information handling (Fig. 10B, step 278; 4:[0055]), implemented by a document editing program running in the computer (Not found), the processes comprising:

allowing a user to enter textual information into a document (Fig. 4, Fig. 5, [0055]) using the document editing program (Not found);

displaying the textual information in the document electronically (Fig. 4, Fig. 5, [0055]) using the document editing program (Not found);

allowing, in the document editing program (Not found), a user to select in the document at least a portion of the textual information (Not found) while the textual information is displayed (Not found);

following user selection of textual information in the document (Not found), analyzing ([0078]-[0079]), by the document editing program (Not found), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (Not found) and what type or types of contact information the selected textual information is (Not found);

providing an input device ([0051], [0055], Fig. 4) configured by the document editing program to allow the user to initiate an operation ([0074]), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Not found), the operation comprising identifying at least part of the selected textual information to use as a search term (Not found) in order to find second information, of a specific type or types, associated in an information source with the search term (Fig. 8, Fig. 9B, [0055], [0067]);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (Figs. 9A-9B, [0074], [0083]-[0086]), wherein the operation further comprises:

causing an electronic search in the information source ([0074], [0079]), by an information management program external to the document editing program (Not found), for the search term in order to find whether the search term is included in the information source (Fig. 10B, steps 274-292, [0079]); and performing an action having a type (Figs. 9A-9B, Fig. 10B, step 294),

wherein the type of action depends at least in part on whether the search term is included in the information source ([0083]-[0086]), and if the search term is so included ([0083]-[0086]), and if the information source includes the second information, the action comprises causing insertion of at least part of the second information into the document (Not found).

2. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the

search term (Fig. 8; [0060]).

3. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when (i) the information source includes the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Not found).

4. At least one non-transitory computer readable medium according to claim 3, wherein the instructions further establish processes wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use for insertion into the document (Not found).

5. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the selected textual information includes information that is not in the information source, the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Not found).

6. At least one non-transitory computer readable medium according to claim 5, wherein the instructions further establish processes wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

7. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source is

available over a network (Not found).

8. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source includes an e-mail address ([0055]) and the operation includes causing an e-mail to be sent to the e-mail address ([0065]).

9. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Fig. 9B; Fig. 10A2, step 258).

10. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the document editing program is a spreadsheet program (Not found).

11. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the type of operation includes updating the document with information from the information source (Not found).

12. A method, for contact information handling (Fig. 10B, step 278; 4; [0055]), implemented by running a document editing program in a computer (Not found), the method comprising:

allowing a user to enter textual information into a document (Fig. 4, Fig. 5, [0055]) using the document editing program (Not found);

displaying the textual information in the document electronically (Fig. 4, Fig. 5, [0055]) using the document editing program (Not found);

allowing, in the document editing program (Not found), a user to select in the document at least a portion of the textual information (Not found) while the textual

information is displayed (Not found);

following user selection of textual information in the document (Not found), analyzing ([0078]-[0079]), by the document editing program (Not found), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (Not found) and what type or types of contact information the selected textual information is (Not found);

providing an input device ([0051], [0055], Fig. 4) configured by the document editing program to allow the user to initiate an operation ([0074]), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Not found), the operation comprising identifying at least part of the selected textual information to use as a search term (Not found) in order to find second information, of a specific type or types, associated in an information source with the search term (Fig. 8, Fig. 9B, [0055], [0067]);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (Figs. 9A-9B, [0074], [0083]-[0086]), wherein the operation further comprises:

causing an electronic search in the information source ([0074], [0079]), by an information management program external to the document editing program (Not found), for the search term in order to find whether the search term is included in the information source (Fig. 10B, steps 274-292, [0079]); and

performing an action having a type (Figs. 9A-9B, Fig. 10B, step 294),

wherein the type of action depends at least in part on whether the search term is included in the information source ([0083]-[0086]), and if the search term is so included ([0083]-[0086]), and if the information source includes the second information, the action comprises causing insertion of at least part of the second information into the document (Not found).

13. A method according to claim 12, wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the

search term (Fig. 8; [0060]); and

when (i) the information source does include the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Not found).

14. A method according to claim 12, wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use insertion into the document (Not found).

15. A method according to claim 12, wherein, when the selected textual information includes information that is not in the information source, and the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Not found);

wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

16. A method according to claim 12, wherein the information source is available over a network (Not found).

17. A method according to claim 12, wherein the information source includes an e-mail address ([0055]) and the operation includes causing an e-mail to be sent to the e-mail address ([0065]).

18. A method according to claim 12, wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information

source corresponding to the name (Fig. 9B; Fig. 10A2, step 258).

19. A method according to claim 12, wherein the type of operation includes updating the document with information from the information source (Not found).

20. An apparatus for contact information handling (Fig. 10B, step 278; 4; [0055]), comprising:

a processor ([0017]); and

a memory ([0017]) storing instructions executable by the processor to perform processes that include:

allowing a user to enter textual information into a document (Fig. 4, Fig. 5, [0055]) using the document editing program (Not found);

displaying the textual information in the document electronically (Fig. 4, Fig. 5, [0055]) using the document editing program (Not found);

allowing, in the document editing program (Not found), a user to select in the document at least a portion of the textual information (Not found) while the textual information is displayed (Not found);

following user selection of textual information in the document (Not found), analyzing ([0078]-[0079]), by the document editing program (Not found), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (Not found) and what type or types of contact information the selected textual information is (Not found);

providing an input device ([0051], [0055], Fig. 4) configured by the document editing program to allow the user to initiate an operation ([0074]), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Not found), the operation comprising identifying at least part of the selected textual information to use as a search term (Not found) in order to find second information, of a specific type or types, associated in an information source with the search term (Fig. 8, Fig. 9B, [0055], [0067]);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (Figs. 9A-9B, [0074], [0083]-[0086]), wherein the operation further comprises:

causing an electronic search in the information source ([0074], [0079]), by an information management program external to the document editing program (Not found), for the search term in order to find whether the search term is included in the information source (Fig. 10B, steps 274-292, [0079]); and performing an action having a type (Figs. 9A-9B, Fig. 10B, step 294),

wherein the type of action depends at least in part on whether the search term is included in the information source ([0083]-[0086]), and if the search term is so included ([0083]-[0086]), and if the information source includes the second information, the action comprises causing insertion of at least part of the second information into the document (Not found).

***Apple Data Detectors User's Manual, Apple Computer Inc., © 1997 ("ADD")***

1. At least one non-transitory computer readable medium encoded with instructions which (p. 3, software requires installation), when loaded on a computer, establish processes for contact information handling (p. 1, "Apple Data Detectors is a new technology that enables your computer to recognize and then act on certain types of information, or *data,* in your documents"), implemented by a document editing program running in the computer (Not found), the processes comprising:

allowing a user to enter textual information into a document using the document editing program (Page 5, step 1);

displaying the textual information in the document electronically using the document editing program (Page 5, step 1);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (Page 5, step 1);

following user selection of textual information in the document (Page 5, step 1), analyzing (p. 4), by the document editing program (Not found, the word processing document is separate from ADD), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (p. 4) and what type or types of contact information the selected textual information is (p. 4);

providing an input device (p. 4, p. 5, step 2, page 6, step 3) configured by the document editing program (Not found) to allow the user to initiate an operation (p. 4, p. 5, step 2, page 6, step 3), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (p. 4, p. 5, step 2, page 6, step 3), the operation comprising identifying at least part of the selected textual information to use as a search term (Not found) in order to find second information, of a specific type or types, associated in an information source(Not found) with the search term (Not found);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program (Not found, ADD does not disclose receipt by a document editing program) of an execute command from the input device, performing the operation (p. 4, p. 5, step 2, page 6, step 3), wherein the operation further comprises:

causing an electronic search in the information source (Not found), by an information management program external to the document editing program (Not found), for the search term in order to find whether the search term is included in the information source (Not found); and performing an action having a type (p. 4, p. 5, step 2, page 6, step 3),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (Not found).

2. At least one non-transitory computer readable medium according to claim 1,

wherein the instructions further establish processes wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Not found).

3. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when (i) the information source includes the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Not found).

4. At least one non-transitory computer readable medium according to claim 3, wherein the instructions further establish processes wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use for insertion into the document (Not found).

5. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the selected textual information includes information that is not in the information source, the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Not found).

6. At least one non-transitory computer readable medium according to claim 5, wherein the instructions further establish processes wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

7. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source is available over a network (Not found).

8. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source includes an e-mail address (Not found) and the operation includes causing an e-mail to be sent (p.4) to the e-mail address (Not found).

9. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Not found).

10. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the document editing program is a spreadsheet program (Not found).

11. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the type of operation includes updating the document with information from the information source (Not found).

12. A method, for contact information handling (p. 1, "Apple Data Detectors is a new technology that enables your computer to recognize and then act on certain types of information, or *data,* in your documents"), implemented by running a document editing program in a computer (Not found), the method comprising:

allowing a user to enter textual information into a document using the document editing program (Page 5, step 1);

displaying the textual information in the document electronically using the document editing program (Page 5, step 1);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (Page 5, step 1);

following user selection of textual information in the document (Page 5, step 1), analyzing (p. 4), by the document editing program (Not found, the word processing document is separate from ADD), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (p. 4) and what type or types of contact information the selected textual information is (p. 4);

providing an input device (p. 4, p. 5, step 2, page 6, step 3) configured by the document editing program (Not found) to allow the user to initiate an operation (p. 4, p. 5, step 2, page 6, step 3), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (p. 4, p. 5, step 2, page 6, step 3), the operation comprising identifying at least part of the selected textual information to use as a search term (Not found) in order to find second information, of a specific type or types, associated in an information source(Not found) with the search term (Not found);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program (Not found, ADD does not disclose receipt by a document editing program) of an execute command from the input device, performing the operation (p. 4, p. 5, step 2, page 6, step 3), wherein the operation further comprises:

causing an electronic search in the information source (Not found), by an information management program external to the document editing program (Not found), for the search term in order to find whether the search term is included in the information source (Not found); and performing an action having a type (p. 4, p. 5, step 2, page 6, step 3),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so

included, and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (Not found).

13. A method according to claim 12, wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Not found); and

when (i) the information source does include the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Not found).

14. A method according to claim 12, wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use insertion into the document (Not found).

15. A method according to claim 12, wherein, when the selected textual information includes information that is not in the information source, and the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Not found);

wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

16. A method according to claim 12, wherein the information source is available over a network (Not found).

17. A method according to claim 12, wherein the information source includes an e-mail address (Not found)and the operation includes causing an e-mail (p.4) to be sent to the e-mail address (Not found).

18. A method according to claim 12, wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Not found).

19. A method according to claim 12, wherein the type of operation includes updating the document with information from the information source (Not found).

20. An apparatus for contact information handling (p. 1, "Apple Data Detectors is a new technology that enables your computer to recognize and then act on certain types of information, or *data,* in your documents"), comprising:

a processor (p.2, "System requirements"); and

a memory (p.2, "System requirements") storing instructions executable by the processor to perform processes that include:

allowing a user to enter textual information into a document using the document editing program (Page 5, step 1);

displaying the textual information in the document electronically using the document editing program (Page 5, step 1);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (Page 5, step 1);

following user selection of textual information in the document (Page 5, step 1), analyzing (p. 4), by the document editing program (Not found, the word processing document is separate from ADD), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (p. 4) and what type or types of contact information the selected textual information is (p. 4);

providing an input device (p. 4, p. 5, step 2, page 6, step 3) configured by the document editing program (Not found) to allow the user to initiate an operation (p. 4, p. 5, step 2, page 6, step 3), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (p. 4, p. 5, step 2, page 6, step 3), the operation comprising identifying at least part of the selected textual information to use as a search term (Not found) in order to find second information, of a specific type or types, associated in an information source(Not found) with the search term (Not found);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program (Not found, ADD does not disclose receipt by a document editing program) of an execute command from the input device, performing the operation (p. 4, p. 5, step 2, page 6, step 3), wherein the operation further comprises:

causing an electronic search in the information source (Not found), by an information management program external to the document editing program (Not found), for the search term in order to find whether the search term is included in the information source (Not found); and performing an action having a type (p. 4, p. 5, step 2, page 6, step 3),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (Not found).

***Bonnie A. Nardi, James R. Miller & David J. Wright, Collaborative, Programmable Intelligent Agents, Communications of the ACM, Vol. 41, No. 3 (Mar. 1998).***

1. At least one non-transitory computer readable medium encoded with instructions which (p. 97, Fig. 1), when loaded on a computer, establish processes for contact information handling (p. 97, Fig. 1), implemented by a document editing program running in the computer (Not found, p. 98, Fig. 2, Apple Data Detectors is separate from

the application programs), the processes comprising:

allowing a user to enter textual information into a document using the document editing program (Not found);

displaying the textual information in the document electronically (p. 97, Fig. 1, Nardi discloses displaying an e-mail message) using the document editing program (Not found);

allowing, in the document editing program (Not found), a user to select in the document at least a portion of the textual information while the textual information is displayed (p. 97, Fig. 1, p. 98, "User interface" section, Nardi discloses selection of text displayed by an e-mail reader program describing an upcoming seminar);

following user selection of textual information in the document (p. 97, Fig. 1), analyzing (p. 97, Fig. 1, p. 99, Fig. 4, Nardi discloses identifying an e-mail address, a date/time, and a telephone number), by the document editing program (Not found), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (p. 97, Fig. 1, p. 99, Fig. 4, Nardi discloses identifying an e-mail address, a date/time, and a telephone number) and what type or types of contact information the selected textual information is (p. 97, Fig. 1, p. 99, Fig. 4, Nardi discloses identifying an e-mail address, a date/time, a telephone number);

providing an input device (p. 97, Fig. 1, Nardi discloses a pop-up menu) configured by the document editing program (Not found) to allow the user to initiate an operation (p. 97, Fig. 1, p. 98, "User interface" section, p. 99, Fig. 4, Nardi discloses placing a meeting on a calendar, adding an address to an address book, dialing a phone number, and generating word processor letterhead), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (p. 97, Fig. 1, Nardi discloses, for example, date/time specific operations such as placing a meeting on a calendar), the operation comprising identifying at least part of the selected textual information to use as a search term (p. 99, Fig. 4, Nardi discloses identifying a telephone number) in order to find second information, of a specific type or types, associated in an information source (p. 99, Fig. 4, "Now Contact 3.5") with the search term (p. 99, Fig. 4, "telephone number");

after identifying at least part of the selected information to use as a search term (p. 99, Fig. 4, "telephone number"), and in consequence of receipt by the document editing program of an execute command from the input device (Not found, Nardi does not disclose receipt of an execute command by a document editing program), performing the operation, wherein the operation further comprises:

causing an electronic search in the information source (p. 99, Fig. 4, Nardi discloses searching for a person and address information in Now Contact 3.5), by an information management program external to the document editing program (p. 99, Fig. 4, "Now Contact 3.5"), for the search term in order to find whether the search term is included in the information source (p. 99, Fig. 4, "Now Contact 3.5"); and performing an action having a type (p. 97, Fig. 1; p. 98, User interface, Nardi discloses placing a meeting on a calendar, adding an address to an address book, and dialing a phone number),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (p. 99, Fig. 4, Nardi discloses generating word processor letterhead using name and address), the action comprises causing insertion of at least part of the second information into the document (Not found).

2. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Not found).

3. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when (i) the information source includes the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further

comprises causing insertion of at least part of the address into the document (Not found).

4. At least one non-transitory computer readable medium according to claim 3, wherein the instructions further establish processes wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use for insertion into the document (Not found).

5. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the selected textual information includes information that is not in the information source, the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (p. 97, Fig. 1, p. 98, "User interface" section, Nardi discloses adding an address to an address book).

6. At least one non-transitory computer readable medium according to claim 5, wherein the instructions further establish processes wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

7. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source is available over a network (Not found).

8. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source includes an e-mail address (Not found) and the operation includes causing an e-mail to be sent to the e-mail address (Not found).

9. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Not found).

10. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the document editing program is a spreadsheet program (Not found).

11. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the type of operation includes updating the document with information from the information source (Not found).

12. A method, for contact information handling (p. 97, Fig. 1), implemented by running a document editing program in a computer (Not found, p. 98, Fig. 2, Apple Data Detectors is separate from the application programs), the method comprising:

allowing a user to enter textual information into a document using the document editing program (Not found);

displaying the textual information in the document electronically (p. 97, Fig. 1, Nardi discloses displaying an e-mail message) using the document editing program (Not found);

allowing, in the document editing program (Not found), a user to select in the document at least a portion of the textual information while the textual information is displayed (p. 97, Fig. 1, p. 98, "User interface" section, Nardi discloses selection of text displayed by an e-mail reader program describing an upcoming seminar);

following user selection of textual information in the document (p. 97, Fig. 1), analyzing (p. 97, Fig. 1, p. 99, Fig. 4, Nardi discloses identifying an e-mail address, a date/time, and a telephone number), by the document editing program (Not found), the

selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (p. 97, Fig. 1, p. 99, Fig. 4, Nardi discloses identifying an e-mail address, a date/time, and a telephone number) and what type or types of contact information the selected textual information is (p. 97, Fig. 1, p. 99, Fig. 4, Nardi discloses identifying an e-mail address, a date/time, a telephone number);

providing an input device (p. 97, Fig. 1, Nardi discloses a pop-up menu) configured by the document editing program (Not found) to allow the user to initiate an operation (p. 97, Fig. 1, p. 98, "User interface" section, p. 99, Fig. 4, Nardi discloses placing a meeting on a calendar, adding an address to an address book, dialing a phone number, and generating word processor letterhead), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (p. 97, Fig. 1, Nardi discloses, for example, date/time specific operations such as placing a meeting on a calendar), the operation comprising identifying at least part of the selected textual information to use as a search term (p. 99, Fig. 4, Nardi discloses identifying a telephone number) in order to find second information, of a specific type or types, associated in an information source (p. 99, Fig. 4, "Now Contact 3.5") with the search term (p. 99, Fig. 4, "telephone number");

after identifying at least part of the selected information to use as a search term (p. 99, Fig. 4, "telephone number"), and in consequence of receipt by the document editing program of an execute command from the input device (Not found, Nardi does not disclose receipt of an execute command by a document editing program), performing the operation, wherein the operation further comprises:

causing an electronic search in the information source (p. 99, Fig. 4, Nardi discloses searching for a person and address information in Now Contact 3.5), by an information management program external to the document editing program (p. 99, Fig. 4, "Now Contact 3.5"), for the search term in order to find whether the search term is included in the information source (p. 99, Fig. 4, "Now Contact 3.5"); and performing an action having a type (p. 97, Fig. 1; p. 98, User interface, Nardi discloses placing a meeting on a calendar, adding an address to an address book, and dialing a phone number),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (p. 99, Fig. 4, Nardi discloses generating word processor letterhead using name and address), the action comprises causing insertion of at least part of the second information into the document (Not found).

13. A method according to claim 12, wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Not found); and

when (i) the information source does include the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Not found).

14. A method according to claim12, wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use insertion into the document (Not found).

15. A method according to claim 12, wherein, when the selected textual information includes information that is not in the information source, and the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (p. 97, Fig. 1, p. 98, "User interface" section, Nardi discloses adding an address to an address book);

wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

16. A method according to claim 12, wherein the information source is available over a network (Not found).

17. A method according to claim 12, wherein the information source includes an e-mail address (Not found) and the operation includes causing an e-mail to be sent to the e-mail address (Not found).

18. A method according to claim 12, wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Not found).

19. A method according to claim 12, wherein the type of operation includes updating the document with information from the information source (Not found).

20. An apparatus for contact information handling (p. 97, Fig. 1), comprising:

a processor (Not found); and

a memory (Not found) storing instructions executable by the processor to perform processes that include:

allowing a user to enter textual information into a document using the document editing program (Not found);

displaying the textual information in the document electronically (p. 97, Fig. 1, Nardi discloses displaying an e-mail message) using the document editing program (Not found);

allowing, in the document editing program (Not found), a user to select in the document at least a portion of the textual information while the textual information is displayed (p. 97, Fig. 1, p. 98, "User interface" section, Nardi discloses selection of text displayed by an e-mail reader program describing an upcoming seminar);

following user selection of textual information in the document (p. 97, Fig.

1), analyzing (p. 97, Fig. 1, p. 99, Fig. 4, Nardi discloses identifying an e-mail address, a date/time, and a telephone number), by the document editing program (Not found), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (p. 97, Fig. 1, p. 99, Fig. 4, Nardi discloses identifying an e-mail address, a date/time, and a telephone number) and what type or types of contact information the selected textual information is (p. 97, Fig. 1, p. 99, Fig. 4, Nardi discloses identifying an e-mail address, a date/time, a telephone number);

providing an input device (p. 97, Fig. 1, Nardi discloses a pop-up menu) configured by the document editing program (Not found) to allow the user to initiate an operation (p. 97, Fig. 1, p. 98, "User interface" section, p. 99, Fig. 4, Nardi discloses placing a meeting on a calendar, adding an address to an address book, dialing a phone number, and generating word processor letterhead), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (p. 97, Fig. 1, Nardi discloses, for example, date/time specific operations such as placing a meeting on a calendar), the operation comprising identifying at least part of the selected textual information to use as a search term (p. 99, Fig. 4, Nardi discloses identifying a telephone number) in order to find second information, of a specific type or types, associated in an information source (p. 99, Fig. 4, "Now Contact 3.5") with the search term (p. 99, Fig. 4, "telephone number");

after identifying at least part of the selected information to use as a search term (p. 99, Fig. 4, "telephone number"), and in consequence of receipt by the document editing program of an execute command from the input device (Not found, Nardi does not disclose receipt of an execute command by a document editing program), performing the operation, wherein the operation further comprises:

causing an electronic search in the information source (p. 99, Fig. 4, Nardi discloses searching for a person and address information in Now Contact 3.5), by an information management program external to the document editing program (p. 99, Fig. 4, "Now Contact 3.5"), for the

search term in order to find whether the search term is included in the information source (p. 99, Fig. 4, "Now Contact 3.5"); and performing an action having a type (p. 97, Fig. 1; p. 98, User interface, Nardi discloses placing a meeting on a calendar, adding an address to an address book, and dialing a phone number),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (p. 99, Fig. 4, Nardi discloses generating word processor letterhead using name and address), the action comprises causing insertion of at least part of the second information into the document (Not found).

***Anind K. Dey, Gregory D. Abowd & Andrew Wood, CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services, Knowledge-Based Systems, Vol. 11 (1998).***

1. At least one non-transitory computer readable medium encoded with instructions which (Abstract), when loaded on a computer, establish processes for contact information handling (p. 3, Section 1), implemented by a document editing program running in the computer (Not found), the processes comprising:

allowing a user to enter textual information into a document using the document editing program (Not found);

displaying the textual information in the document electronically (Fig. 1, p. 4) using the document editing program (Not found);

allowing, in the document editing program (Not found), a user to select in the document at least a portion of the textual information while the textual information is displayed (Fig. 2, p. 4, Dey discloses selecting a URL or a name);

following user selection of textual information in the document (Fig. 2, p. 4), analyzing (p. 5-6, Section 3.2, data is "observed by interested components"), by the document editing program (Not found), the selected textual information to determine if the selected textual information is regarded by the document editing program (Not found)

as contact information and what type or types of contact information the selected textual information is (p. 5-6, Section 3.2, data is "observed by interested components");

providing an input device (Fig. 2, Fig. 4, Dey discloses buttons) configured by the document editing program (Not found) to allow the user to initiate an operation (p. 5-6, Section 3.2, see also p.7-8, section 4.1), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Fig. 2, Fig. 4, p. 4, Section 2.1), the operation comprising identifying at least part of the selected textual information to use as a search term (Not found) in order to find second information, of a specific type or types, associated in an information source with the search term (Fig. 4, p. 4, Section 2.1, Dey discloses searching for a phone number and address associated with a name);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program (Not found) of an execute command from the input device (Fig. 2, Fig. 4, Dey discloses buttons), performing the operation (p. 4, Section 2.1, p. 5-6, Section 3.2), wherein the operation further comprises:

causing an electronic search in the information source (Fig. 4, p. 4, Section 2.1, Dey discloses searching for a phone number and address associated with a name), by an information management program external to the document editing program (Fig. 4, p. 4, Section 2.1, "Contact manager"), for the search term in order to find whether the search term is included in the information source (Fig. 4, p. 4, Section 2.1, Dey discloses searching for a phone number and address associated with a name); and performing an action having a type (Fig. 4, p. 4, Section 2.1, p. 5-6, Section 3.2),

wherein the type of action depends at least in part on whether the search term is included in the information source (Fig. 4, p. 4, Section 2.1), and if the search term is so included, and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (Not found).

2. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Not found).

3. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when (i) the information source includes the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Not found).

4. At least one non-transitory computer readable medium according to claim 3, wherein the instructions further establish processes wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use for insertion into the document (Not found).

5. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the selected textual information includes information that is not in the information source, the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (p. 4, Section 2.1, p. 9-10, Section 7., Fig 11).

6. At least one non-transitory computer readable medium according to claim 5, wherein the instructions further establish processes wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not

found).

7. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source is available over a network (p. 7, Section 4.1).

8. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source includes an e-mail address (Not found) and the operation includes causing an e-mail to be sent (p. 8, Section 5) to the e-mail address (Not found).

9. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Not found).

10. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the document editing program is a spreadsheet program (Not found).

11. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the type of operation includes updating the document with information from the information source (Not found).

12. A method, for contact information handling (p. 3, Section 1), implemented by running a document editing program in a computer (Not found), the method comprising:

allowing a user to enter textual information into a document using the document editing program (Not found);

displaying the textual information in the document electronically (Fig. 1, p. 4)

using the document editing program (Not found);

allowing, in the document editing program (Not found), a user to select in the document at least a portion of the textual information while the textual information is displayed (Fig. 2, p. 4, Dey discloses selecting a URL or a name);

following user selection of textual information in the document (Fig. 2, p. 4), analyzing (p. 5-6, Section 3.2, data is "observed by interested components"), by the document editing program (Not found), the selected textual information to determine if the selected textual information is regarded by the document editing program (Not found) as contact information and what type or types of contact information the selected textual information is (p. 5-6, Section 3.2, data is "observed by interested components");

providing an input device (Fig. 2, Fig. 4, Dey discloses buttons) configured by the document editing program (Not found) to allow the user to initiate an operation (p. 5-6, Section 3.2, see also p.7-8, section 4.1), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Fig. 2, Fig. 4, p. 4, Section 2.1), the operation comprising identifying at least part of the selected textual information to use as a search term (Not found) in order to find second information, of a specific type or types, associated in an information source with the search term (Fig. 4, p. 4, Section 2.1, Dey discloses searching for a phone number and address associated with a name);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program (Not found) of an execute command from the input device (Fig. 2, Fig. 4, Dey discloses buttons), performing the operation (p. 4, Section 2.1, p. 5-6, Section 3.2), wherein the operation further comprises:

causing an electronic search in the information source (Fig. 4, p. 4, Section 2.1, Dey discloses searching for a phone number and address associated with a name), by an information management program external to the document editing program (Fig. 4, p. 4, Section 2.1, "Contact manager"), for the search term in order to find whether the search term is included in the information source (Fig. 4, p. 4, Section 2.1, Dey discloses searching for a phone number and address associated with a name); and performing an action having a type (Fig. 4, p. 4,

Section 2.1, p. 5-6, Section 3.2),

wherein the type of action depends at least in part on whether the search term is included in the information source (Fig. 4, p. 4, Section 2.1), and if the search term is so included, and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (Not found).

13. A method according to claim 12, wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Not found); and

when (i) the information source does include the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Not found).

14. A method according to claim 12, wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use insertion into the document (Not found).

15. A method according to claim 12, wherein, when the selected textual information includes information that is not in the information source, and the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (p. 4, Section 2.1, p. 9-10, Section 7., Fig 11);

wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

16. A method according to claim 12, wherein the information source is available over a network (p. 7, Section 4.1).

17. A method according to claim 12, wherein the information source includes an e-mail address (Not found) and the operation includes causing an e-mail to be sent (p. 8, Section 5) to the e-mail address (Not found).

18. A method according to claim 12, wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Not found).

19. A method according to claim 12, wherein the type of operation includes updating the document with information from the information source (Not found).

20. An apparatus for contact information handling (p. 3, Section 1), comprising:

a processor (Not found); and

a memory (Not found) storing instructions executable by the processor to perform processes that include:

allowing a user to enter textual information into a document using the document editing program (Not found);

displaying the textual information in the document electronically (Fig. 1, p. 4) using the document editing program (Not found);

allowing, in the document editing program (Not found), a user to select in the document at least a portion of the textual information while the textual information is displayed (Fig. 2, p. 4, Dey discloses selecting a URL or a name);

following user selection of textual information in the document (Fig. 2, p. 4), analyzing (p. 5-6, Section 3.2, data is "observed by interested components"), by the document editing program (Not found), the selected textual information to determine if the selected textual information is regarded by the document editing program (Not found) as contact information and what type or types of contact

information the selected textual information is (p. 5-6, Section 3.2, data is "observed by interested components");

providing an input device (Fig. 2, Fig. 4, Dey discloses buttons) configured by the document editing program (Not found) to allow the user to initiate an operation (p. 5-6, Section 3.2, see also p.7-8, section 4.1), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Fig. 2, Fig. 4, p. 4, Section 2.1), the operation comprising identifying at least part of the selected textual information to use as a search term (Not found) in order to find second information, of a specific type or types, associated in an information source with the search term (Fig. 4, p. 4, Section 2.1, Dey discloses searching for a phone number and address associated with a name);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program (Not found) of an execute command from the input device (Fig. 2, Fig. 4, Dey discloses buttons), performing the operation (p. 4, Section 2.1, p. 5-6, Section 3.2), wherein the operation further comprises:

causing an electronic search in the information source (Fig. 4, p. 4, Section 2.1, Dey discloses searching for a phone number and address associated with a name), by an information management program external to the document editing program (Fig. 4, p. 4, Section 2.1, "Contact manager"), for the search term in order to find whether the search term is included in the information source (Fig. 4, p. 4, Section 2.1, Dey discloses searching for a phone number and address associated with a name); and performing an action having a type (Fig. 4, p. 4, Section 2.1, p. 5-6, Section 3.2),

wherein the type of action depends at least in part on whether the search term is included in the information source (Fig. 4, p. 4, Section 2.1), and if the search term is so included, and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (Not

found).

***Getting Result with Microsoft Office 97, Microsoft Corporation © 1995-1997.***

1. At least one non-transitory computer readable medium encoded with instructions which (p. 206-207), when loaded on a computer, establish processes for contact information handling (p. 206-214), implemented by a document editing program running in the computer (p. 208, Microsoft Word), the processes comprising:

allowing a user to enter textual information into a document using the document editing program (p. 208, Mail Merger discloses drafting a form letter in Microsoft Word);

displaying the textual information in the document electronically using the document editing program (p. 208, Mail Merger discloses drafting a form letter in Microsoft Word);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (Not found; Mail Merger at p. 210 discloses inserting merge fields into the document);

following user selection of textual information in the document (Not found), analyzing (Not found; p. 212, "Merge fields in the form letter tell Word where to insert information"), by the document editing program (p. 208, Microsoft Word), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (Not found) and what type or types of contact information the selected textual information is (Not found);

providing an input device (p. 212, "View Merged Data") configured by the document editing program (p. 208, Microsoft Word) to allow the user to initiate an operation (p. 212, merging information such as name and address into a form letter), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Not found), the operation comprising identifying at least part of the selected textual information to use as a search term (Not found) in order to find second information, of a specific type or types, associated in an information source (pp. 210-212, a particular type of contact information is associated with each merge field) with the search term (pp. 210-212, a particular type of contact information is associated with each merge field);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program (p. 208, Microsoft Word) of an execute command from the input device (p. 212, "View Merged Data"), performing the operation (p. 212, merging information such as name and address into a form letter), wherein the operation further comprises:

causing an electronic search in the information source (Not found), by an information management program external to the document editing program (p. 208, Mail merger discloses an address book"), for the search term in order to find whether the search term is included in the information source (Not found); and performing an action having a type (p. 212, merging information such as name and address into a form letter),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (p. 212, Mail Merger discloses displaying the final merge letter).

2. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Not found).

3. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when (i) the information source includes the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (p. 212, Mail Merger discloses displaying the final merge letter).

4. At least one non-transitory computer readable medium according to claim 3, wherein the instructions further establish processes wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use for insertion into the document (Not found).

5. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the selected textual information includes information that is not in the information source, the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Not found).

6. At least one non-transitory computer readable medium according to claim 5, wherein the instructions further establish processes wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

7. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source is available over a network (Not found).

8. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source includes an e-mail address (Not found) and the operation includes causing an e-mail to be sent to the e-mail address (Not found).

9. At least one non-transitory computer readable medium according to claim 1,

wherein the instructions further establish processes wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (p. 212, Mail Merger discloses displaying the name and an address).

10. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the document editing program is a spreadsheet program (Not found).

11. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the type of operation includes updating the document with information from the information source (p. 212, Mail Merger discloses displaying the final merged letter).

12. A method, for contact information handling (p. 206-214), implemented by running a document editing program in a computer (p. 208, Microsoft Word), the method comprising:

allowing a user to enter textual information into a document using the document editing program (p. 208, Mail Merger discloses drafting a form letter in Microsoft Word);

displaying the textual information in the document electronically using the document editing program (p. 208, Mail Merger discloses drafting a form letter in Microsoft Word);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (Not found; Mail Merger at p. 210 discloses inserting merge fields into the document);

following user selection of textual information in the document (Not found), analyzing (Not found; p. 212, "Merge fields in the form letter tell Word where to insert information"), by the document editing program (p. 208, Microsoft Word), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (Not found) and what type or types of

contact information the selected textual information is (Not found);

providing an input device (p. 212, "View Merged Data") configured by the document editing program (p. 208, Microsoft Word) to allow the user to initiate an operation (p. 212, merging information such as name and address into a form letter), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Not found), the operation comprising identifying at least part of the selected textual information to use as a search term (Not found) in order to find second information, of a specific type or types, associated in an information source (pp. 210-212, a particular type of contact information is associated with each merge field) with the search term (pp. 210-212, a particular type of contact information is associated with each merge field);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program (p. 208, Microsoft Word) of an execute command from the input device (p. 212, "View Merged Data"), performing the operation (p. 212, merging information such as name and address into a form letter), wherein the operation further comprises:

causing an electronic search in the information source (Not found), by an information management program external to the document editing program (p. 208, Mail merger discloses an address book"), for the search term in order to find whether the search term is included in the information source (Not found); and performing an action having a type (p. 212, merging information such as name and address into a form letter),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (p. 212, Mail Merger discloses displaying the final merge letter).

13. A method according to claim 12, wherein:

when the information source does not include the search term, the action

comprises causing indication to the user that the information source does not include the search term (Not found); and

when (i) the information source does include the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (p. 212, Mail Merger discloses displaying the final merge letter).

14. A method according to claim 12, wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use insertion into the document (Not found).

15. A method according to claim 12, wherein, when the selected textual information includes information that is not in the information source, and the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Not found);

wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

16. A method according to claim 12, wherein the information source is available over a network (Not found).

17. A method according to claim 12, wherein the information source includes an e-mail address (Not found) and the operation includes causing an e-mail to be sent to the e-mail address (Not found).

18. A method according to claim 12, wherein, when the type or types of contact information includes a name, the operation includes causing display by the information

management program of at least part of a contact information record in the information source corresponding to the name (p. 212, Mail Merger discloses displaying the name and an address).

19. A method according to claim 12, wherein the type of operation includes updating the document with information from the information source (p. 212, Mail Merger discloses displaying the final merged letter).

20. An apparatus for contact information handling (p. 206-214), comprising:

a processor (p.114); and

a memory (p. 402) storing instructions executable by the processor to perform processes that include:

allowing a user to enter textual information into a document using the document editing program (p. 208, Mail Merger discloses drafting a form letter in Microsoft Word);

displaying the textual information in the document electronically using the document editing program (p. 208, Mail Merger discloses drafting a form letter in Microsoft Word);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (Not found; Mail Merger at p. 210 discloses inserting merge fields into the document);

following user selection of textual information in the document (Not found), analyzing (Not found; p. 212, "Merge fields in the form letter tell Word where to insert information"), by the document editing program (p. 208, Microsoft Word), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (Not found) and what type or types of contact information the selected textual information is (Not found);

providing an input device (p. 212, "View Merged Data") configured by the document editing program (p. 208, Microsoft Word) to allow the user to

initiate an operation (p. 212, merging information such as name and address into a form letter), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Not found), the operation comprising identifying at least part of the selected textual information to use as a search term (Not found) in order to find second information, of a specific type or types, associated in an information source (pp. 210-212, a particular type of contact information is associated with each merge field) with the search term (pp. 210-212, a particular type of contact information is associated with each merge field);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program (p. 208, Microsoft Word) of an execute command from the input device (p. 212, "View Merged Data"), performing the operation (p. 212, merging information such as name and address into a form letter), wherein the operation further comprises:

causing an electronic search in the information source (Not found), by an information management program external to the document editing program (p. 208, Mail merger discloses an address book"), for the search term in order to find whether the search term is included in the information source (Not found); and performing an action having a type (p. 212, merging information such as name and address into a form letter),

wherein the type of action depends at least in part on whether the search term is included in the information source (Not found), and if the search term is so included, and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (p. 212, Mail Merger discloses displaying the final merge letter).

***User Manual for AddressMate and AddressMate Plus, Addressmate Software, © 1994-1995***

1. At least one non-transitory computer readable medium encoded with instructions which (p. 36, 43), when loaded on a computer, establish processes for contact information handling (p. 43), implemented by a document editing program running in the computer (p. 36, 43, AddressMate is a macro), the processes comprising:

allowing a user to enter textual information into a document using the document editing program (p. 44, step 1, "Microsoft Word");

displaying the textual information in the document electronically using the document editing program (p. 44, step 1, "Microsoft Word");

allowing, in the document editing program (p. 44, "Microsoft Word"), a user to select in the document at least a portion of the textual information while the textual information is displayed (p. 44, step 2);

following user selection of textual information in the document (p. 44, step 2), analyzing (Not found), by the document editing program (p. 36, 43), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (Not found) and what type or types of contact information the selected textual information is (Not found);

providing an input device (p. 44, step 3, "Amate/Retrieve command") configured by the document editing program (p. 36, 43) to allow the user to initiate an operation (p. 44, step 3, insertion of an address), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Not found), the operation comprising identifying at least part of the selected textual information to use as a search term (Not found) in order to find second information, of a specific type or types, associated in an information source with the search term (p. 43-44, searching for an address in AddressMate database);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program (p. 36, 43) of an execute command from the input device (p. 44, step 3, "Amate/Retrieve command"), performing the operation (p. 43-44, searching for an address and inserting the address into a document), wherein the operation further comprises:

causing an electronic search in the information source (p. 43-44, searching for an address), by an information management program (p. 43) external to the document editing program, for the search term in order to find whether the search term is included in the information source (p. 43-44, searching for an address); and performing an action having a type (p. 43-44, inserting the address into a document),

wherein the type of action depends at least in part on whether the search term is included in the information source (p. 44, step 3), and if the search term is so included, and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (p. 44-45, inserting the address into a document).

2. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Not found).

3. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when (i) the information source includes the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (p. 64, "Harrison Schreppel").

4. At least one non-transitory computer readable medium according to claim 3, wherein the instructions further establish processes wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use for insertion into the document (Not found).

5. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the selected textual information includes information that is not in the information source, the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Not found).

6. At least one non-transitory computer readable medium according to claim 5, wherein the instructions further establish processes wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

7. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source is available over a network (Not found).

8. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source includes an e-mail address (Not found) and the operation includes causing an e-mail to be sent to the e-mail address (Not found).

9. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (p. 44-45, inserting a name and address into a document).

10. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the document editing program is a spreadsheet program (Not found).

11. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the type of operation includes updating the document with information from the information source (p. 44-45, inserting a name and address into a document).

12. A method, for contact information handling (p. 43), implemented by running a document editing program in a computer (p. 36, 43, AddressMate is a macro), the method comprising:

allowing a user to enter textual information into a document using the document editing program (p. 44, step 1, "Microsoft Word");

displaying the textual information in the document electronically using the document editing program (p. 44, step 1, "Microsoft Word");

allowing, in the document editing program (p. 44, "Microsoft Word"), a user to select in the document at least a portion of the textual information while the textual information is displayed (p. 44, step 2);

following user selection of textual information in the document (p. 44, step 2), analyzing (Not found), by the document editing program (p. 36, 43), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (Not found) and what type or types of contact information the selected textual information is (Not found);

providing an input device (p. 44, step 3, "Amate/Retrieve command") configured by the document editing program (p. 36, 43) to allow the user to initiate an operation (p. 44, step 3, insertion of an address), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Not found), the operation comprising identifying at least part of the selected textual information to use as a search term (Not found) in order to find second information, of a

specific type or types, associated in an information source with the search term (p. 43-44, searching for an address in AddressMate database);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program (p. 36, 43) of an execute command from the input device (p. 44, step 3, "Amate/Retrieve command"), performing the operation (p. 43-44, searching for an address and inserting the address into a document), wherein the operation further comprises:

causing an electronic search in the information source (p. 43-44, searching for an address), by an information management program (p. 43) external to the document editing program, for the search term in order to find whether the search term is included in the information source (p. 43-44, searching for an address); and performing an action having a type (p. 43-44, inserting the address into a document),

wherein the type of action depends at least in part on whether the search term is included in the information source (p. 44, step 3), and if the search term is so included, and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (p. 44-45, inserting the address into a document).

13. A method according to claim 12, wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Not found); and

when (i) the information source does include the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (p. 64, "Harrison Schreppel").

14. A method according to claim 12, wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use insertion into the document (Not found).

15. A method according to claim 12, wherein, when the selected textual information includes information that is not in the information source, and the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Not found);

wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Not found).

16. A method according to claim 12, wherein the information source is available over a network (Not found).

17. A method according to claim 12, wherein the information source includes an e-mail address and the operation includes causing an e-mail to be sent to the e-mail address (Not found).

18. A method according to claim 12, wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (p. 44-45, inserting a name and address into a document).

19. A method according to claim 12, wherein the type of operation includes updating the document with information from the information source (Not found).

20. An apparatus for contact information handling (p. 43), comprising:

a processor (p. 9); and

a memory (p. 9) storing instructions executable by the processor to perform processes that include:

allowing a user to enter textual information into a document using the document editing program (p. 44, step 1, "Microsoft Word");

displaying the textual information in the document electronically using the document editing program (p. 44, step 1, "Microsoft Word");

allowing, in the document editing program (p. 44, "Microsoft Word"), a user to select in the document at least a portion of the textual information while the textual information is displayed (p. 44, step 2);

following user selection of textual information in the document (p. 44, step 2), analyzing (Not found), by the document editing program (p. 36, 43), the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information (Not found) and what type or types of contact information the selected textual information is (Not found);

providing an input device (p. 44, step 3, "Amate/Retrieve command") configured by the document editing program (p. 36, 43) to allow the user to initiate an operation (p. 44, step 3, insertion of an address), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Not found), the operation comprising identifying at least part of the selected textual information to use as a search term (Not found) in order to find second information, of a specific type or types, associated in an information source with the search term (p. 43-44, searching for an address in AddressMate database);

after identifying at least part of the selected information to use as a search term (Not found), and in consequence of receipt by the document editing program (p. 36, 43) of an execute command from the input device (p. 44, step 3, "Amate/Retrieve command"), performing the operation (p. 43-44, searching for an address and inserting the address into a document), wherein the operation

further comprises:

causing an electronic search in the information source (p. 43-44, searching for an address), by an information management program (p. 43) external to the document editing program, for the search term in order to find whether the search term is included in the information source (p. 43-44, searching for an address); and performing an action having a type (p. 43-44, inserting the address into a document),

wherein the type of action depends at least in part on whether the search term is included in the information source (p. 44, step 3), and if the search term is so included, and if the information source includes the second information (Not found), the action comprises causing insertion of at least part of the second information into the document (p. 44-45, inserting the address into a document).

**9(C) Detailed Explanation of Patentability:**

***Chalas U.S. Patent No. 5,392,386***

Chalas discloses a system for adding functions to an existing application program (*e.g.*, Microsoft Word) (Abstract). Chalas describes a clipboard that is used to transfer information between application programs (2:38-44). Chalas discloses a "capture mechanism" that monitors communications between the operating system and an application program (5:26-30). The capture mechanism also simulates user input commands from user input devices to the operating system, and simulates commands from the operating system to the application program (5:64-68; *See also* 15:20-25). By simulating the commands and messages, Chalas is able to 1) transfer information that has been selected by the user from the application program to the clipboard, 2) modify that information, and 3) transfer the modified information back to the application program (6:1-6).

In this manner, Chalas imparts add-on functionality to the document editing program (8:39-43). For example, Chalas can impart real-time spell checking (8: 43-54) and translation (6:54-56). Additional functions includes detecting zip codes and providing the name of town and state (12:51-53) or detecting key phrases that designated to be replaced by a picture in the text (12:57-60).

Chalas does not teach several elements or limitations of the claims. Although Chalas does disclose detecting a zip code (which is a type of contact information) and providing the name of a town, state, etc. (12:51-53), the description of this embodiment lacks many of the elements and limitations required by the claims. For example, Chalas does not disclose an "operation being of a type depending at least in part on the type or types of contact information of the selected textual information," as required by the claims. In Chalas, it appears the type of operation (such as spell checking) is selected by the user before text is processed (8:39-44). Therefore, in Chalas, the type of contact information is irrelevant to the type of operation performed.

Furthermore, the claims require "causing an electronic search in the information source … for the search term in order to find whether the search term is included in the information source." Although Chalas mentions zip codes (a type of contact information), it is unclear, based on Chalas' short description, how the zip code is used to provide town and state information. For example, Chalas does not describe "causing an electronic search in the information source for the "zip code" by an information management program external to the document editing program. Also, Chalas does not disclose or suggest "inserting the second information into the document." For at least the reasons stated above, Chalas does not meet the limitations of the claims.

***Pandit U.S. Patent No. 5,859,636***

Pandit discloses a method for recognizing textual data and performing a particular operation based on the recognized textual data (Abstract). For example, if an e-mail address is recognized within the text, then a menu appears that enables the user to either send an e-mail or add the e-mail to an address book (Fig. 1d; 2:51-63).

Pandit does not meet several elements and limitations of the claims. Among other things, Pandit does not disclose the claim requirement of "identifying at least part of the selected textual information to use as a search term in order to find second information." In fact, there is no electronic search in an information source at all, let alone a search for a search term by an information management program external to the document handling program, as required by the claims. Also, the actions performed by Pandit (*e.g.*, opening a scheduling program or opening an e-mail template), do not depend on whether the search term is included in the information source, as required by the claims. Nor is any second information inserted into the document, as required by the claims. For at least the reasons above, Pandit does not meet the limitations of the claims.

***Miller et al. U.S. Patent No. 5,946,647***

Miller discloses a method for detecting data in a document and performing a particular action based on the detected data (Abstract). For example, if a telephone number is recognized within the document, then a "mouse-down" operation is placed over the telephone number (Fig. 6; 5:38-50). When a user clicks on the mouse-down

operation, a pop-up menu appears that enables the user to either dial the number or put the electronic number in a telephone book (Fig. 7; 5:38-50).

Miller does not meet several elements and limitations of the claims. Among other things, Miller does not disclose "allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed." Miller simply does not disclose selection of a portion of the document.

Furthermore, Miller does not search in an information source in order to find second information associated with a search term. The claims require:

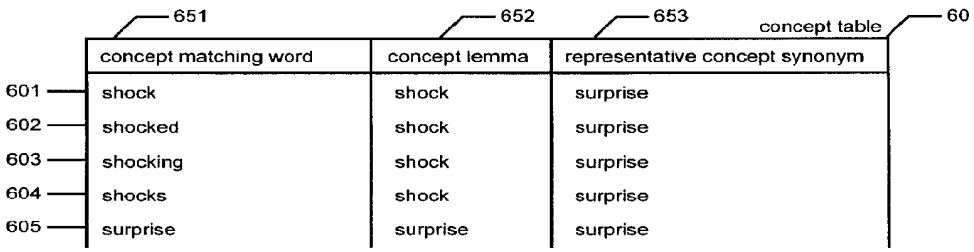
- identifying at least part of the selected textual information to use as a search term in order to find second information, of a specific type or types, associated in an information source with the search term
- causing an electronic search in the information source, by an information management program external to the document handling program, for the search term in order to find whether the search term is included in the information source

None of these limitations are met by Miller.

Since Miller does not disclose searching for a search term in an information source, then it also does not disclose an action that depends on whether "the search term is included in the information source" and "if the information source includes the second information." In fact, the action required by the claims never happens because Miller never inserts second information into the document. For at least the reasons above, Miller does not meet the limitations of the claims.

**Ho et al. U.S. Patent No. 6,021,412**

Ho discloses adding conceptually relevant graphics to a presentation document (Abstract). Ho discloses analyzing a presentation document to determine whether words in the document have any match in a list of concept matching words (Fig. 5, step 501; 3:40 – 4:2; 5:14-16). Each concept matching word is mapped to a concept lemma and a representative concept synonym that has at least one associated graphic in a graphics library. The concept table below shows the relationship:




According to this relationship, there is *always* a graphic associated with a concept matching word. Furthermore, if a word in the document is found in the concept matching word list, then it must exist in the concept table and have an associated graphic in the graphics library. Once the concept matching words are identified in the document, the associated concept lemmas are shown to the user and the user can select a particular concept lemma (5:34-42). Ho then discloses mapping the selected concept lemma to a representative concept synonym and using the representative concept synonym to search in a graphic library for graphics (Fig. 5, steps 505, 506; 6:7-12). The graphics that are associated with the concept synonym are displayed to the user and he can select one for insertion into the presentation document (6:13-34).

Ho does not meet several elements and limitations of the claims. Ho does not disclose analyzing "the selected textual information to determine if the selected textual

information is regarded by the document editing program as contact information," as required by the claims. Also, Ho does not disclose or suggest analyzing the selected textual information to determine "what type or types of contact information the selected textual information is." Identifying contact information and determining the type of contact information is not disclosed or even suggested by Ho.

Since Ho does not recognize contact information or differentiate between types of contact information, there can be no "operation [] of a type depending at least in part on the type or types of contact information of the selected textual information," as required by the claims. Ho discloses mapping concept matching words to their concept lemmas and synonyms, but this operation does not depend on contact information or a type of contact information.

Ho also does not disclose a "type of action [that] depends at least in part on whether the search term is included in the information source" and "if the information source includes the second information," as required by the claims. Although Ho does disclose displaying a graphic to the user, this action does not depend on "whether the search term is included in the information source" and "if the information source includes the second information." This is true because, according to the relationship in the concept table, there is *always* a graphic associated with a concept matching word. Indeed, a situation wherein there is no graphic for a concept matching word is not addressed or disclosed by Ho. For at least the reasons above, Ho does not meet the limitations of the claims.

**Tso U.S. Patent No. 6,085,201**

Tso discloses composing an e-mail message with the help of a context-sensitive template engine (Abstract). The method begins when a user selects a text string (4:32-35). Once the user invokes the template engine, the template engine analyzes the text string to determine an appropriate template for the e-mail message (Abstract; 2:7-21). Tso discloses "decomposing" the text string into individual words and comparing the individual words to keywords associated with predetermined templates (5:7-17). The template with the best match between keywords and individual words is chosen as the most appropriate template and that template is presented to the user (5:7-17; 5:42-44).

Tso does not meet several elements and limitations of the claims. Tso does not disclose "analyzing … the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information," as required by the claims. Also, Tso does not disclose or suggest analyzing the selected textual information to determine "what type or types of contact information the selected textual information is." Identifying contact information and determining the type of contact information is not disclosed or even suggested by Tso.

Furthermore, although Tso discloses invoking a template engine, the invocation of the template engine is not dependent on the *type* of text string selected by the user. In other words, there is no "operation [] of a type depending at least in part on the type or types of contact information of the selected textual information," as required by the claims.

Tso also does not disclose a "type of action [that] depends on at least in part on whether the search term is included in the information source," as required by the claims.

Tso discloses presenting an appropriate template to the user (5:42-44), but this action does not depend on *whether* a search term is included in an information source. Instead, it appears that a template is always presented to the user because a situation wherein there is no match between keywords and individual words from the text strings is not addressed or disclosed by Tso. Additionally, Tso does not disclose inserting second information into the document, as required by the claims. For at least the reasons above, Tso does not meet the limitations of the claims.

***Hachamovitch et al.*** ***U.S. Patent No. 6,377,965***

The Hachamovitch reference is directed to an auto-completion system (Abstract). As the user types a data entry into a document, the system searches for possible entry completions corresponding to the partial data entry (Abstract). The system then presents the user with possible entry completions for the partial data entry (Abstract). The user can select one of the possible entry completions and the system will automatically complete his data entry (Abstract). In one particular embodiment, the system uses context to limit the number of entries presented to the user (Fig 5, step 512; 5:38-54). For example, if the user entered information into a structured data field, then the system will only present possible completions that are relevant to that structured data field (11:14-35).

The system of Hachamovitch does not meet several limitations of the claims. The claims require "analyzing … the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information and what type or types of contact information the selected textual information is. Once

the text is analyzed to determine what type of contact information it is, the claims require "causing an electronic search in the information source … for the search term in order to find whether the search term is included in the information source." The Hachamovitch reference discloses the opposite approach. After the Hachamovitch system determines that enough letters have been typed into the document, it searches a list for possible completion suggestions (Fig 5, step 512; 5:38-54). Although the Applicant does not believe Hachamovitch discloses analysis, as required by the claims, if any "analysis" is performed by the Hachamovitch system, it is done only after the searching, namely in step 512, to limit the number of candidates identified in the search. Thus, the Hachamovitch system does not meet the limitations of the claims because there is no search after *an* analysis, as required by the claims.

Hachamovitch also does not meet the requirements of the claims because his system searches *before* an execute command is received, whereas the claims require searching "in consequence of receipt … of an execute command from the input device." Hachamovitch discloses receiving a completion command to replace the partial data entry with a completion. Fig. 5, steps 516, 518; 15:55-67). Receipt of this command happens after the system searches for possible completions (Fig. 5, step 506). In contrast, the claims require searching for second information "in consequence of receipt … of an execute command from the input device." Thus, the claims require a different order from what is disclosed in Hachamovitch. For at least the reasons above, Hachamovitch does not meet the limitations of the claims.

***Bocking et al. U.S. Application Pub. No. 2006/0047644***

Bocking discloses a search application for searching personal information located within various databases, applications or components of a handheld electronic device (Abstract). The search is a "Global Search" application in that it allows a user to search multiple databases or applications on the same device in a single search ([0055]). For example, a user can search messages, address book contracts, calendar appointments, tasks or memos that match a search of search criteria ([0055]). The search criteria specified include text and/or name ([0055]). Items that match the search criteria are displayed and grouped by type (*e.g.*, all messages are grouped together) ([0055]). Items from the search results can be directly viewed, directly edited or directly deleted without having to open the items in their native application ([0065]). For example, if a specific name is entered as the search criteria and the search retrieves an Address Book contact with that name, an email can be sent to the contact without the user having to open another program. The Global Search application can be initiated from a stand-alone icon or it may be initiated from within a calendar, address book, task, memo or message application on the device ([0097]).

Bocking does not meet several elements and limitations of the claims. Among other things, Bocking does not disclose contact information handling implemented by a document editing program. For example, Bocking does not disclose displaying textual information in a document using a document editing program, as required by the claims. Bocking also does not disclose allowing a user to select in that document a portion of textual information while the information is displayed. Instead, in Bocking, the user inputs certain search terms into a field in a Global Search application.

Furthermore, Bocking does not disclose "analyzing … the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information and what type or types of contact information the selected textual information is." Whether or not the search terms entered by the user constitute contact information is irrelevant to the search of Bocking.

Bocking also does not disclose "identifying at least part of the selected textual information to use as a search term." To the contrary, in Bocking, it is always the user that selects the search criteria. The user inputs the exact text to be searched and also designates the databases or applications to be searched. For at least the reasons above, Bocking does not meet the limitations of the claims.

***Apple Data Detectors User's Manual, Apple Computer Inc., © 1997 ("ADD").***

The Apple Data Detectors (ADD) User's Manual discloses a method for finding (recognizing) and acting on certain types of information (p. 1). ADD starts when a user highlights text in a word processing document (p. 5, step 1). ADD then analyzes the text to determine whether it recognizes some portion of that text as, for example, an e-mail address or web address (p. 4; p.5, step 2). The user can use a contextual menu to initiate an action that is related to the recognized data (p.5, step 2). For example, if ADD recognizes a web address, the user may use the contextual menu to initiate viewing the website associated with the web address (p.5, steps 3 and 4).

ADD does not meet several elements and limitations of the claims. Among other things, ADD does not disclose analyzing selected textual information *by the document editing program*, as required by the claims. Also, ADD does not disclose providing an

input device configured *by the document editing program*, as required by the claims. Instead, ADD itself is a part of the operating system and it analyzes the selected textual information and configures any input device. As ADD is a part of the operating system, which is separate from the document editing program, ADD does not meet the limitations of the claims.

Furthermore, ADD does not disclose "causing an electronic search in the information source … for the search term in order to find whether the search term is included in the information source," as required by the claims. ADD simply does not disclose, suggest, or teach searching for a search term (or for second information) in an information source, as required by the claims. Also, ADD does not disclose inserting at least part of the second information into a document, as required by the claims. For at least the reasons above, ADD does not meet the limitations of the claims.

***Bonnie A. Nardi, James R. Miller & David J. Wright, Collaborative, Programmable Intelligent Agents, Communications of the ACM, Vol. 41, No. 3 (Mar. 1998).***

The Nardi reference describes the functionalities of Apple Data Detectors (ADD), which is a program that finds (recognizes) and acts on certain types of information (p. 98-99). ADD starts when a user highlights text in a document (p. 97, Fig. 1, p. 98, "User interface" section). ADD then analyzes the highlighted text to determine whether it recognizes some portion of the text as, for example, an e-mail address or telephone number (p. 97, Fig. 1, p. 99, Fig. 4). The user can use a pop-up menu to initiate an action that is related to the recognized data (p. 97, Fig. 1). For example, if ADD recognizes a

date and time, the user may use the pop-up menu to place the date in an electronic calendar (p. 97, Fig. 1).

In another example, Nardi describes a functionality wherein ADD recognizes a telephone number in the selected text and uses the telephone number to search for an associated name and address in a personal information manager program (p.99, Fig. 4). Add then creates a new word processor document and uses the name and address to generate word processor letterhead (p.99, Fig. 4).

Nardi does not meet several elements and limitations of the claims. Among other things, Nardi and its description of ADD does not disclose analyzing selected textual information *by the document editing program*, as required by the claims. Instead, ADD analyzes the selected textual information and ADD is separate from the document editing program (*See* p. 99, Fig. 4, ADD opens up a word processing document). Similarly, Nardi does not disclose providing an input device configured by a document editing program, as required by the claims. Instead, it is ADD, a part of the operating system, that provides the input device.

Nardi also does not disclose performing an action that "depends at least in part on whether the search term is included in the information source." Although Nardi does disclose using a telephone number to search for names and addresses in an information source and then generating a word processor letterhead using the names and addresses, generating the letterhead does not depend on whether or not the telephone number is included in an information source and whether or not names and addresses exist in the information source. To the contrary, as disclosed, ADD carries through with an action irrespective of whether a search term is found in an information source. A situation

wherein the search terms and associated second information do not exist in the information source is not contemplated by Nardi.

Furthermore, Nardi does not disclose an action that comprises "causing insertion of at least part of the second information into the document," as required by the claims. Although Nardi does disclose generating word processor letterhead, the letterhead is not created in a document in which textual information is selected, as required by the claims.

ADD, by its design, does not have information about where the "original" selected text came from, or how to insert any second information back into the document from which the selected text came. (See action script in Fig. 4). For at least the reasons above, Nardi does not meet the limitations of the claims.

***Anind K. Dey, Gregory D. Abowd & Andrew Wood, CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services, Knowledge-Based Systems, Vol. 11 (1998).***

The Dey reference describes a tool called Cyberdesk. Cyberdesk is a framework that supports the automatic integration of various software applications (p.4, Section 2). In one example, a user highlights a URL in an e-mail message and Cyberdesk offers the user the following options: 1) search for the URL using a search engine, 2) find pages that reference the URL using the search engine, or 3) display the URL using a web browser (p.4, Section 2). The user selects the third option and, as a result, the URL is displayed in a web browser (Fig. 3, p.4, Section 2). In another example, the user highlights a name in the e-mail message and Cyberdesk offers various options that are specific to the name (*e.g.*, look up the name in a contact manager) (p.4, Section 2).

The Dey reference does not meet several elements and limitations of the claims. Among other things, the Dey reference does not disclose contact information handling implemented by a document editing program. For example, Dey does not disclose analyzing selected textual information *by the document editing program*, as required by the claims. Cyberdesk itself analyzes text highlighted by the user and Cyberdesk is *separate from* any document editing programs. Indeed, Cyberdesk is the framework that integrates various applications.

Cyberdesk also does not disclose "identifying at least part of the selected textual information to use as a search term," as required by the claims. To the contrary, it appears that Cyberdesk merely searches for what the user has highlighted and, for this reason, Cyberdesk does not identify search terms, as required by the claims.

Furthermore, Cyberdesk does not disclose "causing insertion of at least part of the second information into the document," as required by the claims. Cyberdesk simply does not disclose, teach, or suggest inserting second information into documents. For at least the reasons above, the Dey reference does not meet the limitations of the claims.

***Getting Result with Microsoft Office 97, Microsoft Corporation © 1995-1997.***

The Microsoft Office 97 Manual describes a functionality in Microsoft Word known as Mail Merger (p. 206-214). Mail Merger allows a user to insert various types of contact information into a form letter (p. 206). The process begins when a user drafts a form letter and places merger fields, or placeholders, into the document (p. 207, 210). The user can select an address book to create a list of contact information for insertion into the form letter (p. 208). When the user selects the "view merged data" button the

contact information from the address book (names and addresses) are inserted and displayed in the form letter (p. 212).

The Mail Merger functionality does not meet several elements and limitations of the claims. Among other things, Mail merger does not disclose "allowing … a user to select in the document at least a portion of the textual information while the textual information is displayed," as required by the claims. Instead, Mail Merger discloses inserting merge fields into the document. Displayed textual information is not selected.

Furthermore, Mail Merger does not disclose "analyzing … the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information and what type or types of contact information the selected textual information is," as required by the claims. Mail Merger does not determine whether text constitutes contact information because each merge field that has been placed by the user merely represents a particular field in a data source (*e.g.*, name field, address field, city field, and state field in an address book). Mail Merger simply inserts information from those fields into a document without any consideration for their type. Furthermore, the merge fields themselves do not constitute contact information, they simply point to where contact information should be inserted.

Mail Merger also does not disclose "causing an electronic search in the information source" for the search term or for second information associated with the search term. Instead, Mail Merger simply retrieves the contact information associated with each merge field from a listing of contact information that was complied previously by the user (p. 208-209). For at least the reasons above, the Microsoft Office 97 Manual does not meet the limitations of the claims.

***User Manual for AddressMate and AddressMate Plus, Addressmate Software, © 1994-1995.***

The User Manual for AddressMate and AddressMate Plus describes a functionality referred to as "Retrieving an Address" (p. 43-45). This functionality allows a user to insert address information into a document based upon a partially completed address (p. 43-44). The process begins when a user types part of the name or address into the document (p. 43-44). The user then uses a mouse to select the partially completed information (p. 44). When the user executes the "Amate/Retrieve" command, AddressMate searches for the partially completed information in the AddressMate database (p. 44). If an address matching the information is found, it is inserted into the document (p. 44).

AddressMate does not meet several elements and limitations of the claims. Among other things, AddressMate does not disclose "analyzing … the selected textual information to determine if the selected textual information is regarded by the document editing program as contact information and what type or types of contact information the selected textual information is," as required by the claims. As disclosed by the manual, AddressMate merely uses the selected partially completed information in a search. There is no further analysis of the selected information.

AddressMate also does not disclose an operation that depends "at least in part on the type or types of contact information of the selected textual information," as required by the claims, because AddressMate does not recognize contact information or differentiate between different types of contact information. Instead, AddressMate

provides input devices (*e.g.*, Amate Retrieve) to initiate operations without any consideration as to what type of information was selected by the user. For at least the reasons above, User Manual for AddressMate and AddressMate Plus does not meet the limitations of the claims.

**9(D) Concise Statement of Utility**

The invention has utility in that it provides for contact information handling in a computer, particularly in the context of a document editing program.

**9(E) Showing of Support under 35 USC 112**

***35 U.S.C 112 P1 Support in Present Application***

Each of the claim elements is supported by the present disclosure as required under 35 U.S.C. 112, first paragraph at least as indicated below:

1. At least one non-transitory computer readable medium (p. 3, l. 2) encoded with instructions which, when loaded on a computer, establish processes for contact information handling (p.1, ll. 17-19), implemented by a document editing program running in the computer (p.1, ll. 20-23), the processes comprising:

allowing a user to enter textual information into a document using the document editing program (Figs. 3 and 5; p.10, ll. 12-13, 22-23; p.18, ll. 9-13);

displaying the textual information in the document electronically using the document editing program (Figs. 3 and 5; p.10, ll. 12-13, 22-23; p.18, ll. 9-13);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (p.19, ll. 1-4);

following user selection of textual information in the document (p.19, ll. 1-4),

analyzing, by the document editing program (p. 18, ll. 4-6), the selected textual information (Fig. 1a, 4; 2a, 4; p.7, ll. 8-10) to determine if the selected textual information is regarded by the document editing program as contact information and what type or types of contact information the selected textual information is (p.7, ll. 8-23; p. 8, ll. 1-8; fig 1a, 6,; fig 2a, 6);

providing an input device configured by the document editing program to allow the user to initiate an operation (p.5, ll. 18-20, 23-27; p.18, ll. 19-23), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Fig. 1a, 1b, and 2a, 2b, steps 6, 8, 10, 12, 14; p. 7, ll. 10-13, 19-20, 21-23; p. 8, ll. 7-8), the operation comprising identifying at least part of the selected textual information to use as a search term in order to find second information, of a specific type or types, associated in an information source with the search term (Figs 1a and 1b, 2a and 2b, steps 12, 14; p. 7, ll. 21-23; p. 8, ll. 7-8);

after identifying at least part of the selected information to use as a search term (p. 8, ll. 7-8; p.9, ll. 7-8), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (p.5, l. 23 – p.6, l. 4), wherein the operation further comprises:

causing an electronic search in the information source (Figs. 1a and 1b, 2a and 2b, steps 12, 14; p.5, l. 23 – p.6, l. 4), by an information management program external to the document editing program (p.1. l. 28 – p.2, l. 4), for the search term in order to find whether the search term is included in the information source (p.7, ll. 21-23; p.8, ll. 6-7, 9, 22-23); and performing an action having a type (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p.7, l. 19 - p. 8, l. 13),

wherein the type of action depends at least in part on whether the search term is included in the information source (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p. 7, l. 23 - p. 8, l. 13), and if the search term is so included, and if the information source includes the second information (Figs. 1a and 1b, steps 20, 22, 30, 32; p.7, l. 23 - p.8, l.6; p.8, ll. 11-13), the action comprises causing insertion of at least part of the second information into the document (Figs. 1a and 1b, steps 20, 22, 30, 32; Figs . 4, 6, 7, 9-12, 15, p.7, l. 23 - p.8, l.6; p.8, ll. 11-13).

2. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Fig. 5 , 6, and 8; p.10, l. 21 – p.11, l. 7, p. 12, ll. 2-3).

3. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when (i) the information source includes the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Figs. 1a and 1b, steps 18, 22; Figs. 3 and 4; p.10, ll. 11-19).

4. At least one non-transitory computer readable medium according to claim 3, wherein the instructions further establish processes wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use for insertion into the document (Figs. 1a and 1b, steps 18, 20, 22; Figs. 3, 4, and 10; p.13, ll. 10-22).

5. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the selected textual information includes information that is not in the information source, the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Figs. 1a and 1b, steps 12, 18, 24, 16, steps 12, 14, 26, 28, 36; Fig. 6 and 7; p. 10, l. 2 – 6, p. 10, l. 21–p. 11. l. 15).

6. At least one non-transitory computer readable medium according to claim 5, wherein the instructions further establish processes wherein, when the selected textual information includes a name and an address and the information source does not include

the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Figs. 1a and 1b, steps 12, 14, 26, 28; p. 10, ll. 1- 2, p. 10, l. 21 – p. 11. l. 15).

7. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source is available over a network (p. 3, 13 – 15, p. 16, l. 20 – p. 17, l. 5; p. 17, ll. 6-10).

8. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source includes an e-mail address (Fig. 1a, between steps 6 and 10; Fig. 12, 100; p. 6, ll. 7-10) and the operation includes causing an e-mail to be sent to the e-mail address (p. 6, l. 22 – p.7, l. 3).

9. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Figs. 1a and 1b, steps 6, 12, 18, 20, steps 6, 14, 26, 30, 36; Fig. 9, 72; p. 12, ll. 10-21).

10. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the document editing program is a spreadsheet program (Figs. 14 and 15; p. 5, ll. 18-22; p. 2, l. 13 – p. 3, l.6; p. 16, ll. 1-9).

11. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the type of operation includes updating the document with information from the information source (Figs. 1a and 1b, steps 18, 22, steps 18, 20, 22; Figs. 3 and 4; p.10, ll. 11-19).

12. A method, for contact information handling (p.1, ll. 17-19), implemented by running a document editing program in a computer (p.1, ll. 20-23), the method comprising:

allowing a user to enter textual information into a document using the document editing program (Figs. 3 and 5; p.10, ll. 12-13, 22-23; p.18, ll. 9-13);

displaying the textual information in the document electronically using the document editing program (Figs. 3 and 5; p.10, ll. 12-13, 22-23; p.18, ll. 9-13);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (p.19, ll. 1-4);

following user selection of textual information in the document (p.19, ll. 1-4), analyzing, by the document editing program (p. 18, ll. 4-6), the selected textual information (Fig. 1a, 4; 2a, 4; p.7, ll. 8-10) to determine if the selected textual information is regarded by the document editing program as contact information and what type or types of contact information the selected textual information is (p.7, ll. 8-23; p. 8, ll. 1-8; fig 1a, 6,; fig 2a, 6);

providing an input device configured by the document editing program to allow the user to initiate an operation (p.5, ll. 18-20, 23-27; p.18, ll. 19-23), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Fig. 1a, 1b, and 2a, 2b, steps 6, 8, 10, 12, 14; p. 7, ll. 10-13, 19-20, 21-23; p. 8, ll. 7-8), the operation comprising identifying at least part of the selected textual information to use as a search term in order to find second information, of a specific type or types, associated in an information source with the search term (Figs 1a and 1b, 2a and 2b, steps 12, 14; p. 7, ll. 21-23; p. 8, ll. 7-8);

after identifying at least part of the selected information to use as a search term (p. 8, ll. 7-8; p.9, ll. 7-8), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (p.5, l. 23 – p.6, l. 4), wherein the operation further comprises:

causing an electronic search in the information source (Figs. 1a and 1b, 2a and 2b, steps 12, 14; p.5, l. 23 – p.6, l. 4), by an information management program external to the document editing program (p.1, l. 29 – p.2, l. 4), for the

search term in order to find whether the search term is included in the information source (p.7, ll. 21-23; p.8, ll. 6-7, 9, 22-23); and performing an action having a type (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p.7, l. 19 - p. 8, l. 13),

wherein the type of action depends at least in part on whether the search term is included in the information source (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p. 7, l. 23 - p. 8, l. 13), and if the search term is so included, and if the information source includes the second information (Figs. 1a and 1b, steps 20, 22, 30, 32; p.7, l. 23 - p.8, l.6; p.8, ll. 11-13), the action comprises causing insertion of at least part of the second information into the document (Figs. 1a and 1b, steps 20, 22, 30, 32; Figs . 4, 6, 7, 9-12, 15, p.7, l. 23 - p.8, l.6; p.8, ll. 11-13).

13. A method according to claim 12, wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Fig. 5 , 6, and 8; p.10, l. 21 – p.11, l. 7, p. 12, ll. 2-3); and

when (i) the information source does include the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Figs. 1a and 1b, steps 18, 22; Figs. 3 and 4; p.10, ll. 11-19).

14. A method according to claim 12, wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use insertion into the document (Figs. 1a and 1b, steps 18, 20, 22; Figs. 3, 4, and 10; p.13, ll. 10-22).

15. A method according to claim 12, wherein, when the selected textual information includes information that is not in the information source, and the operation further comprises:

allowing the user to cause storage of at least some of the selected textual

information in the information source (Figs. 1a and 1b, steps 12, 18, 24, 16, steps 12, 14, 26, 28, 36; Fig. 6 and 7; p. 10, l. 2 – 6, p. 10, l. 21 – p. 11. l. 15);

wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Figs. 1a and 1b, steps 12, 14, 26, 28; p. 10, ll. 1-2, p. 10, l. 21 – p. 11. l. 15).

16. A method according to claim 12, wherein the information source is available over a network (p. 3, 13 – 15, p. 16, l. 20 – p. 17, l. 5; p. 17, ll. 6-10).

17. A method according to claim 12, wherein the information source includes an e-mail address (Fig. 1a, between steps 6 and 10; Fig. 12, 100; p. 6, ll. 7-10) and the operation includes causing an e-mail to be sent to the e-mail address (p. 6, l. 22 – p.7, l. 3).

18. A method according to claim 12, wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Figs. 1a and 1b, steps 6, 12, 18, 20, steps 6, 14, 26, 30, 36; Fig. 9, 72; p. 12, ll. 10-21).

19. A method according to claim 12, wherein the type of operation includes updating the document with information from the information source (Figs. 1a and 1b, steps 18, 22, steps 18, 20, 22; Figs. 3 and 4; p.10, ll. 11-19).

20. An apparatus for contact information handling (p.1, ll.17-19), comprising:

a processor (p. 16, l. 12-p. 17, l. 5); and

a memory storing instructions executable by the processor to perform processes (p. 16, l. 12-p. 17, l. 5) that include:

allowing a user to enter textual information into a document using the document editing program (Figs. 3 and 5; p.10, ll. 12-13, 22-23; p.18, ll. 9-13);

displaying the textual information in the document electronically using the document editing program (Figs. 3 and 5; p.10, ll. 12-13, 22-23; p.18, ll. 9-13);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (p.19, ll. 1-4);

following user selection of textual information in the document (p.19, ll. 1-4), analyzing, by the document editing program (p. 18, ll. 4-6), the selected textual information (Fig. 1a, 4; 2a, 4; p.7, ll. 8-10) to determine if the selected textual information is regarded by the document editing program as contact information and what type or types of contact information the selected textual information is (p.7, ll. 8-23; p. 8, ll. 1-8; fig 1a, 6; fig 2a, 6);

providing an input device configured by the document editing program to allow the user to initiate an operation (p.5, ll. 18-20, 23-27; p.18, ll. 19-23), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Fig. 1a, 1b, and 2a, 2b, steps 6, 8, 10, 12, 14; p. 7, ll. 10-13, 19-20, 21-23; p. 8, ll. 7-8), the operation comprising identifying at least part of the selected textual information to use as a search term in order to find second information, of a specific type or types, associated in an information source with the search term (Figs 1a and 1b, 2a and 2b, steps 12, 14; p. 7, ll. 21-23; p. 8, ll. 7-8);

after identifying at least part of the selected information to use as a search term (p. 8, ll. 7-8; p.9, ll. 7-8), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (p.5, l. 23 – p.6, l. 4), wherein the operation further comprises:

causing an electronic search in the information source (Figs. 1a and 1b, 2a and 2b, steps 12, 14; p.5, l. 23 – p.6, l. 4), by an information management program external to the document editing program (p.1, l. 29 – p.2, l. 4), for the search term in order to find whether the search term is included in the information source (p.7, ll. 21-23; p.8, ll. 6-7, 9, 22-23);

and performing an action having a type (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p.7, l. 19 - p. 8, l. 13),

wherein the type of action depends at least in part on whether the search term is included in the information source (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p. 7, l. 23 - p. 8, l. 13), and if the search term is so included, and if the information source includes the second information (Figs. 1a and 1b, steps 20, 22, 30, 32; p.7, l. 23 - p.8, l.6; p.8, ll. 11-13), the action comprises causing insertion of at least part of the second information into the document (Figs. 1a and 1b, steps 20, 22, 30, 32; Figs. 4, 6, 7, 9-12, 15, p.7, l. 23 - p.8, l.6; p.8, ll. 11-13).

***35 U.S.C 112 P1 Support in Application No. 12/182,048***

The present application claims priority to U.S. Application No. 12/182,048. Each of the claim elements is supported by U.S. Application No. 12/182,048 as required under 35 U.S.C. 112, first paragraph at least as indicated below:

1. At least one non-transitory computer readable medium (p. 3, l. 2) encoded with instructions which, when loaded on a computer, establish processes for contact information handling (p.1, ll. 17-19), implemented by a document editing program running in the computer (p.1, ll. 20-23), the processes comprising:

allowing a user to enter textual information into a document using the document editing program (Figs. 3 and 5; p.10, ll. 12-13, 22-23; p.18, ll. 9-13);

displaying the textual information in the document electronically using the document editing program (Figs. 3 and 5; p.10, ll. 12-13, 22-23; p.18, ll. 9-13);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (p.19, ll. 1-4);

following user selection of textual information in the document (p.19, ll. 1-4),

analyzing, by the document editing program (p. 18, ll. 4-6), the selected textual information (Fig. 1a, 4; 2a, 4; p.7, ll. 8-10) to determine if the selected textual information is regarded by the document editing program as contact information and what type or types of contact information the selected textual information is (p.7, ll. 8-23; p. 8, ll. 1-8; fig 1a, 6,; fig 2a, 6);

providing an input device configured by the document editing program to allow the user to initiate an operation (p.5, ll. 18-20, 23-27; p.18, ll. 19-23), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Fig. 1a, 1b, and 2a, 2b, steps 6, 8, 10, 12, 14; p. 7, ll. 10-13, 19-20, 21-23; p. 8, ll. 7-8), the operation comprising identifying at least part of the selected textual information to use as a search term in order to find second information, of a specific type or types, associated in an information source with the search term (Figs 1a and 1b, 2a and 2b, steps 12, 14; p. 7, ll. 21-23; p. 8, ll. 7-8);

after identifying at least part of the selected information to use as a search term (p. 8, ll. 7-8; p.9, ll. 7-8), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (p.5, l. 23 – p.6, l. 4), wherein the operation further comprises:

causing an electronic search in the information source (Figs. 1a and 1b, 2a and 2b, steps 12, 14; p.5, l. 23 – p.6, l. 4), by an information management program external to the document editing program (p.1, l. 28 – p.2, l. 4), for the search term in order to find whether the search term is included in the information source (p.7, ll. 21-23; p.8, ll. 6-7, 9, 22-23); and performing an action having a type (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p.7, l. 19 - p. 8, l. 13),

wherein the type of action depends at least in part on whether the search term is included in the information source (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p. 7, l. 23 - p. 8, l. 13), and if the search term is so included, and if the information source includes the second information (Figs. 1a and 1b, steps 20, 22, 30, 32; p.7, l. 23 - p.8, l.6; p.8, ll. 11-13), the action comprises causing insertion of at least part of the second information into the document (Figs. 1a and 1b, steps 20, 22, 30, 32; Figs . 4, 6, 7, 9-12, 15, p.7, l. 23 - p.8, l.6; p.8, ll. 11-13).

2. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Fig. 5 , 6, and 8; p.10, l. 21 – p.11, l. 7, p. 12, ll. 2-3).

3. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when (i) the information source includes the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Figs. 1a and 1b, steps 18, 22; Figs. 3 and 4; p.10, ll. 11-19).

4. At least one non-transitory computer readable medium according to claim 3, wherein the instructions further establish processes wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use for insertion into the document (Figs. 1a and 1b, steps 18, 20, 22; Figs. 3, 4, and 10; p.13, ll. 10-22).

5. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the selected textual information includes information that is not in the information source, the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Figs. 1a and 1b, steps 12, 18, 24, 16, steps 12, 14, 26, 28, 36; Fig. 6 and 7; p. 10, l. 2 – 6, p. 10, l. 21–p. 11. l. 15).

6. At least one non-transitory computer readable medium according to claim 5, wherein the instructions further establish processes wherein, when the selected textual information includes a name and an address and the information source does not include

the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Figs. 1a and 1b, steps 12, 14, 26, 28; p. 10, ll. 1- 2, p. 10, l. 21 – p. 11. l. 15).

7. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source is available over a network (p. 3, 13 – 15, p. 16, l. 20 – p. 17, l. 5; p. 17, ll. 6-10).

8. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source includes an e-mail address (Fig. 1a, between steps 6 and 10; Fig. 12, 100; p. 6, ll. 7-10) and the operation includes causing an e-mail to be sent to the e-mail address (p. 6, l. 22 – p.7, l. 3).

9. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Figs. 1a and 1b, steps 6, 12, 18, 20, steps 6, 14, 26, 30, 36; Fig. 9, 72; p. 12, ll. 10-21).

10. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the document editing program is a spreadsheet program (Figs. 14 and 15; p. 5, ll. 18-22; p. 2, l. 13 – p. 3, l.6; p. 16, ll. 1-9).

11. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the type of operation includes updating the document with information from the information source (Figs. 1a and 1b, steps 18, 22, steps 18, 20, 22; Figs. 3 and 4; p.10, ll. 11-19).

12. A method, for contact information handling (p.1, ll. 17-19), implemented by running a document editing program in a computer (p.1, ll. 20-23), the method comprising:

allowing a user to enter textual information into a document using the document editing program (Figs. 3 and 5; p.10, ll. 12-13, 22-23; p.18, ll. 9-13);

displaying the textual information in the document electronically using the document editing program (Figs. 3 and 5; p.10, ll. 12-13, 22-23; p.18, ll. 9-13);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (p.19, ll. 1-4);

following user selection of textual information in the document (p.19, ll. 1-4), analyzing, by the document editing program (p. 18, ll. 4-6), the selected textual information (Fig. 1a, 4; 2a, 4; p.7, ll. 8-10) to determine if the selected textual information is regarded by the document editing program as contact information and what type or types of contact information the selected textual information is (p.7, ll. 8-23; p. 8, ll. 1-8; fig 1a, 6,; fig 2a, 6);

providing an input device configured by the document editing program to allow the user to initiate an operation (p.5, ll. 18-20, 23-27; p.18, ll. 19-23), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Fig. 1a, 1b, and 2a, 2b, steps 6, 8, 10, 12, 14; p. 7, ll. 10-13, 19-20, 21-23; p. 8, ll. 7-8), the operation comprising identifying at least part of the selected textual information to use as a search term in order to find second information, of a specific type or types, associated in an information source with the search term (Figs 1a and 1b, 2a and 2b, steps 12, 14; p. 7, ll. 21-23; p. 8, ll. 7-8);

after identifying at least part of the selected information to use as a search term (p. 8, ll. 7-8; p.9, ll. 7-8), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (p.5, l. 23 – p.6, l. 4), wherein the operation further comprises:

causing an electronic search in the information source (Figs. 1a and 1b, 2a and 2b, steps 12, 14; p.5, l. 23 – p.6, l. 4), by an information management program external to the document editing program (p.1. l. 28 – p.2, l. 4), for the

search term in order to find whether the search term is included in the information source (p.7, ll. 21-23; p.8, ll. 6-7, 9, 22-23); and performing an action having a type (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p.7, l. 19 - p. 8, l. 13),

wherein the type of action depends at least in part on whether the search term is included in the information source (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p. 7, l. 23 - p. 8, l. 13), and if the search term is so included, and if the information source includes the second information (Figs. 1a and 1b, steps 20, 22, 30, 32; p.7, l. 23 - p.8, l.6; p.8, ll. 11-13), the action comprises causing insertion of at least part of the second information into the document (Figs. 1a and 1b, steps 20, 22, 30, 32; Figs . 4, 6, 7, 9-12, 15, p.7, l. 23 - p.8, l.6; p.8, ll. 11-13).

13. A method according to claim 12, wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Fig. 5 , 6, and 8; p.10, l. 21 – p.11, l. 7, p. 12, ll. 2-3); and

when (i) the information source does include the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Figs. 1a and 1b, steps 18, 22; Figs. 3 and 4; p.10, ll. 11-19).

14. A method according to claim 12, wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use insertion into the document (Figs. 1a and 1b, steps 18, 20, 22; Figs. 3, 4, and 10; p.13, ll. 10-22).

15. A method according to claim 12, wherein, when the selected textual information includes information that is not in the information source, and the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Figs. 1a and 1b, steps 12, 18, 24, 16, steps 12, 14,

26, 28, 36; Fig. 6 and 7; p. 10, l. 2 – 6, p. 10, l. 21 – p. 11. l. 15);

wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Figs. 1a and 1b, steps 12, 14, 26, 28; p. 10, ll. 1-2, p. 10, l. 21 – p. 11. l. 15).

16. A method according to claim 12, wherein the information source is available over a network (p. 3, 13 – 15, p. 16, l. 20 – p. 17, l. 5; p. 17, ll. 6-10).

17. A method according to claim 12, wherein the information source includes an e-mail address (Fig. 1a, between steps 6 and 10; Fig. 12, 100; p. 6, ll. 7-10) and the operation includes causing an e-mail to be sent to the e-mail address (p. 6, l. 22 – p.7, l. 3).

18. A method according to claim 12, wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Figs. 1a and 1b, steps 6, 12, 18, 20, steps 6, 14, 26, 30, 36; Fig. 9, 72; p. 12, ll. 10-21).

19. A method according to claim 12, wherein the type of operation includes updating the document with information from the information source (Figs. 1a and 1b, steps 18, 22, steps 18, 20, 22; Figs. 3 and 4; p.10, ll. 11-19).

20. An apparatus for contact information handling (p.1, ll. 17-19), comprising:

a processor (p. 16, l. 12-p. 17, l. 5); and

a memory storing instructions executable by the processor to perform processes (p. 16, l. 12-p. 17, l. 5) that include:

allowing a user to enter textual information into a document using the document editing program (Figs. 3 and 5; p.10, ll. 12-13, 22-23; p.18, ll. 9-13);

displaying the textual information in the document electronically using the document editing program (Figs. 3 and 5; p.10, ll. 12-13, 22-23; p.18, ll. 9-13);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (p.19, ll. 1-4);

following user selection of textual information in the document (p.19, ll. 1-4), analyzing, by the document editing program (p. 18, ll. 4-6), the selected textual information (Fig. 1a, 4; 2a, 4; p.7, ll. 8-10) to determine if the selected textual information is regarded by the document editing program as contact information and what type or types of contact information the selected textual information is (p.7, ll. 8-23; p. 8, ll. 1-8; fig 1a, 6,; fig 2a, 6);

providing an input device configured by the document editing program to allow the user to initiate an operation (p.5, ll. 18-20, 23-27; p.18, ll. 19-23), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Fig. 1a, 1b, and 2a, 2b, steps 6, 8, 10, 12, 14; p. 7, ll. 10-13, 19-20, 21-23; p. 8, ll. 7-8), the operation comprising identifying at least part of the selected textual information to use as a search term in order to find second information, of a specific type or types, associated in an information source with the search term (Figs 1a and 1b, 2a and 2b, steps 12, 14; p. 7, ll. 21-23; p. 8, ll. 7-8);

after identifying at least part of the selected information to use as a search term (p. 8, ll. 7-8; p.9, ll. 7-8), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (p.5, l. 23 – p.6, l. 4), wherein the operation further comprises:

causing an electronic search in the information source (Figs. 1a and 1b, 2a and 2b, steps 12, 14; p.5, l. 23 – p.6, l. 4), by an information management program external to the document editing program (p.1, l. 28 – p.2, l. 4), for the search term in order to find whether the search term is included in the information source (p.7, ll. 21-23; p.8, ll. 6-7, 9, 22-23); and performing an action having a type (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p.7, l. 19 - p. 8, l. 13),

wherein the type of action depends at least in part on whether the search term is included in the information source (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p. 7, l. 23 - p. 8, l. 13), and if the search term is so included, and if the information source includes the second information (Figs. 1a and 1b, steps 20, 22, 30, 32; p.7, l. 23 - p.8, l.6; p.8, ll. 11-13), the action comprises causing insertion of at least part of the second information into the document (Figs. 1a and 1b, steps 20, 22, 30, 32; Figs . 4, 6, 7, 9-12, 15, p.7, l. 23 - p.8, l.6; p.8, ll. 11-13).

***35 U.S.C 112 P1 Support in Application No. 09/923,134***

The present application claims priority to U.S. Application No. 09/923,134. Each of the claim elements is supported by U.S. Application No. 09/923,134 as required under 35 U.S.C. 112, first paragraph at least as indicated below:

1. At least one non-transitory computer readable medium (p. 3, l. 6) encoded with instructions which, when loaded on a computer, establish processes for contact information handling (p.1, ll. 7-12), implemented by a document editing program running in the computer (p.1, ll. 7-12), the processes comprising:

allowing a user to enter textual information into a document using the document editing program (Figs. 3 and 5; p.10, ll. 16-17, p. 11, 4-5; p.18, ll. 14-16);

displaying the textual information in the document electronically using the document editing program (Figs. 3 and 5; p.10, ll. 16-17, p. 11, 4-5; p.18, ll. 14-16);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (p.19, ll. 6-10);

following user selection of textual information in the document (p.19, ll. 6-10), analyzing, by the document editing program (p. 18, ll. 9-11), the selected textual information (Fig. 1a, 4; 2a, 4; p.7, ll. 16-18) to determine if the selected textual information is regarded by the document editing program as contact information and what type or types of contact information the selected textual information is (p. 7, l. 16 –

p.8, l. 16; fig 1a, 6; fig 2a, 6);

providing an input device configured by the document editing program to allow the user to initiate an operation (p.5, ll. 21 – p. 6, 8; p.19, ll. 1-5), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Fig. 1a, 1b, and 2a, 2b, steps 6, 8, 10, 12, 14; p. 7, ll. 18-21; p. 8, ll. 4-8; p. 8, ll. 15-16), the operation comprising identifying at least part of the selected textual information to use as a search term in order to find second information, of a specific type or types, associated in an information source with the search term (Figs 1a and 1b, 2a and 2b, steps 12, 14; p. 8, ll. 6-8; p. 8, ll. 15-16);

after identifying at least part of the selected information to use as a search term (p. 8, ll. 15-16; p.9, ll. 14-16), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (p.6, ll. 4-16), wherein the operation further comprises:

causing an electronic search in the information source (Figs. 1a and 1b, 2a and 2b, steps 12, 14; p.6, ll. 4-16), by an information management program external to the document editing program (p.1, ll. 17-20), for the search term in order to find whether the search term is included in the information source (p.8, ll. 6-8; p.8, ll. 15-17; p. 9, ll. 6-7); and performing an action having a type (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p.8, ll. 4-21),

wherein the type of action depends at least in part on whether the search term is included in the information source (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p. 8, ll. 8-21), and if the search term is so included, and if the information source includes the second information (Figs. 1a and 1b, steps 20, 22, 30, 32; p.8, ll. 8-13; p. 8, ll. 18-20), the action comprises causing insertion of at least part of the second information into the document (Figs. 1a and 1b, steps 20, 22, 30, 32; Figs . 4, 6, 7, 9-12, 15, p.8, ll. 8-13; p.8, ll. 18-20).

2. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the

search term (Fig. 5 , 6, and 8; p.11, ll. 3-12; p. 12, ll. 6-7).

3. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when (i) the information source includes the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Figs. 1a and 1b, steps 18, 22; Figs. 3 and 4; p.10, ll. 15-21).

4. At least one non-transitory computer readable medium according to claim 3, wherein the instructions further establish processes wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use for insertion into the document (Figs. 1a and 1b, steps 18, 20, 22; Figs. 3, 4, and 10; p.13, l. 13 - p. 14, l. 2).

5. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the selected textual information includes information that is not in the information source, the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Figs. 1a and 1b, steps 12, 18, 24, 16, steps 12, 14, 26, 28, 36; Fig. 6 and 7; p. 10, l. 7-11; p. 11, ll. 3-20).

6. At least one non-transitory computer readable medium according to claim 5, wherein the instructions further establish processes wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Figs. 1a and 1b, steps 12, 14, 26, 28; p. 10, ll. 6-7; p. 11, ll. 3-20).

7. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source is available over a network (p. 3, 11-13, p. 17, ll. 4-16).

8. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source includes an e-mail address (Fig. 1a, between steps 6 and 10; Fig. 12, 100; p. 6, ll. 15-18) and the operation includes causing an e-mail to be sent to the e-mail address (p. 7, ll. 7-11).

9. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Figs. 1a and 1b, steps 6, 12, 18, 20, steps 6, 14, 26, 30, 36; Fig. 9, 72; p. 12, l. 13 – p. 13, l. 2).

10. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the document editing program is a spreadsheet program (Figs. 14 and 15; p. 5, l. 20 – p. 6, l. 3; p. 2, l. 9 – p. 3, l. 4; p. 16, ll. 7-14).

11. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the type of operation includes updating the document with information from the information source (Figs. 1a and 1b, steps 18, 22, steps 18, 20, 22; Figs. 3 and 4; p.10, ll. 15-21).

12. A method, for contact information handling (p.1, ll. 7-12), implemented by running a document editing program in a computer (p.1, ll. 7-12), the method comprising:

allowing a user to enter textual information into a document using the document editing program (Figs. 3 and 5; p.10, ll. 16-17, p. 11, 4-5; p.18, ll. 14-16);

displaying the textual information in the document electronically using the document editing program (Figs. 3 and 5; p.10, ll. 16-17, p. 11, 4-5; p.18, ll. 14-16);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (p.19, ll. 6-10);

following user selection of textual information in the document (p.19, ll. 6-10), analyzing, by the document editing program (p. 18, ll. 9-11), the selected textual information (Fig. 1a, 4; 2a, 4; p.7, ll. 16-18) to determine if the selected textual information is regarded by the document editing program as contact information and what type or types of contact information the selected textual information is (p. 7, l. 16 – p.8, l. 16; fig 1a, 6; fig 2a, 6);

providing an input device configured by the document editing program to allow the user to initiate an operation (p.5, ll. 21 – p. 6, 8; p.19, ll. 1-5), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Fig. 1a, 1b, and 2a, 2b, steps 6, 8, 10, 12, 14; p. 7, ll. 18-21; p. 8, ll. 4-8; p. 8, ll. 15-16), the operation comprising identifying at least part of the selected textual information to use as a search term in order to find second information, of a specific type or types, associated in an information source with the search term (Figs 1a and 1b, 2a and 2b, steps 12, 14; p. 8, ll. 6-8; p. 8, ll. 15-16);

after identifying at least part of the selected information to use as a search term (p. 8, ll. 15-16; p.9, ll. 14-16), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (p.6, ll. 4-16), wherein the operation further comprises:

causing an electronic search in the information source (Figs. 1a and 1b, 2a and 2b, steps 12, 14; p.6, ll. 4-16), by an information management program external to the document editing program (p.1, ll. 17-20), for the search term in order to find whether the search term is included in the information source (p.8, ll. 6-8; p.8, ll. 15-17; p. 9, ll. 6-7); and performing an action having a type (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p.8, ll. 4-21),

wherein the type of action depends at least in part on whether the search term is included in the information source (Figs. 1a and 1b, steps 20, 22, 24, 28,

30, 32; p. 8, ll. 8-21), and if the search term is so included, and if the information source includes the second information (Figs. 1a and 1b, steps 20, 22, 30, 32; p.8, ll. 8-13; p. 8, ll. 18-20), the action comprises causing insertion of at least part of the second information into the document (Figs. 1a and 1b, steps 20, 22, 30, 32; Figs . 4, 6, 7, 9-12, 15, p.8, ll. 8-13; p.8, ll. 18-20).

13. A method according to claim 12, wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Fig. 5 , 6, and 8; p.11, ll. 3-12; p. 12, ll. 6-7); and

when (i) the information source does include the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Figs. 1a and 1b, steps 18, 22; Figs. 3 and 4; p.10, ll. 15-21).

14. A method according to claim 12, wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use insertion into the document (Figs. 1a and 1b, steps 18, 20, 22; Figs. 3, 4, and 10; p.13, l. 13 - p. 14, l. 2).

15. A method according to claim 12, wherein, when the selected textual information includes information that is not in the information source, and the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Figs. 1a and 1b, steps 12, 18, 24, 16, steps 12, 14, 26, 28, 36; Fig. 6 and 7; p. 10, l. 2 – 6, p. 10, l. 21 – p. 11. l. 15);

wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Figs. 1a and 1b, steps 12, 18, 24, 16, steps 12,

14, 26, 28, 36; Fig. 6 and 7; p. 10, l. 7-11; p. 11, ll. 3-20).

16. A method according to claim 12, wherein the information source is available over a network (Figs. 1a and 1b, steps 12, 14, 26, 28; p. 10, ll. 6-7; p. 11, ll. 3-20).

17. A method according to claim 12, wherein the information source includes an e-mail address (Fig. 1a, between steps 6 and 10; Fig. 12, 100; p. 6, ll. 15-18) and the operation includes causing an e-mail to be sent to the e-mail address (p. 7, ll. 7-11).

18. A method according to claim 12, wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Figs. 1a and 1b, steps 6, 12, 18, 20, steps 6, 14, 26, 30, 36; Fig. 9, 72; p. 12, l. 13 – p. 13, l. 2).

19. A method according to claim 12, wherein the type of operation includes updating the document with information from the information source (Figs. 1a and 1b, steps 18, 22, steps 18, 20, 22; Figs. 3 and 4; p.10, ll. 15-21).

20. An apparatus for contact information handling (p.1, ll. 7-12), comprising:

a processor (p. 16, ll. 17 – p. 17, ll. 11); and

a memory storing instructions executable by the processor to perform processes (p. 16, ll. 17 – p. 17, ll. 11) that include:

allowing a user to enter textual information into a document using the document editing program (Figs. 3 and 5; p.10, ll. 16-17, p. 11, 4-5; p.18, ll. 14-16);

displaying the textual information in the document electronically using the document editing program (Figs. 3 and 5; p.10, ll. 16-17, p. 11, 4-5; p.18, ll. 14-16);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual

information is displayed (p.19, ll. 6-10);

following user selection of textual information in the document (p.19, ll. 6-10), analyzing, by the document editing program (p. 18, ll. 9-11), the selected textual information (Fig. 1a, 4; 2a, 4; p.7, ll. 16-18) to determine if the selected textual information is regarded by the document editing program as contact information and what type or types of contact information the selected textual information is (p. 7, l. 16 – p.8, l. 16; fig 1a, 6; fig 2a, 6);

providing an input device configured by the document editing program to allow the user to initiate an operation (p.5, ll. 21 – p. 6, 8; p.19, ll. 1-5), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Fig. 1a, 1b, and 2a, 2b, steps 6, 8, 10, 12, 14; p. 7, ll. 18-21; p. 8, ll. 4-8; p. 8, ll. 15-16), the operation comprising identifying at least part of the selected textual information to use as a search term in order to find second information, of a specific type or types, associated in an information source with the search term (Figs 1a and 1b, 2a and 2b, steps 12, 14; p. 8, ll. 6-8; p. 8, ll. 15-16);

after identifying at least part of the selected information to use as a search term (p. 8, ll. 15-16; p.9, ll. 14-16), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (p.6, ll. 4-16), wherein the operation further comprises:

causing an electronic search in the information source (Figs. 1a and 1b, 2a and 2b, steps 12, 14; p.6, ll. 4-16), by an information management program external to the document editing program (p.1, ll. 17-20), for the search term in order to find whether the search term is included in the information source (p.8, ll. 6-8; p.8, ll. 15-17; p. 9, ll. 6-7); and performing an action having a type (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p.8, ll. 4-21),

wherein the type of action depends at least in part on whether the search term is included in the information source (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p. 8, ll. 8-21), and if the search term is so included, and if the information source includes the second information (Figs. 1a

and 1b, steps 20, 22, 30, 32; p.8, ll. 8-13; p. 8, ll. 18-20), the action comprises causing insertion of at least part of the second information into the document (Figs. 1a and 1b, steps 20, 22, 30, 32; Figs . 4, 6, 7, 9-12, 15, p.8, ll. 8-13; p.8, ll. 18-20).

***35 U.S.C 112 P1 Support in Application No. 09/189,626***

The present application claims priority to U.S. Application No. 09/189,626. Each of the claim elements is supported by U.S. Application No. 09/189,626 as required under 35 U.S.C. 112, first paragraph at least as indicated below:

1. At least one non-transitory computer readable medium (p. 3, l. 6) encoded with instructions which, when loaded on a computer, establish processes for contact information handling (p.1, ll. 7-12), implemented by a document editing program running in the computer (p.1, ll. 7-12), the processes comprising:

allowing a user to enter textual information into a document using the document editing program (Figs. 3 and 5; p.10, ll. 16-17, p. 11, 4-5; p.18, ll. 14-16);

displaying the textual information in the document electronically using the document editing program (Figs. 3 and 5; p.10, ll. 16-17, p. 11, 4-5; p.18, ll. 14-16);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (p.19, ll. 6-10);

following user selection of textual information in the document (p.19, ll. 6-10), analyzing, by the document editing program (p. 18, ll. 9-11), the selected textual information (Fig. 1a, 4; 2a, 4; p.7, ll. 16-18) to determine if the selected textual information is regarded by the document editing program as contact information and what type or types of contact information the selected textual information is (p. 7, l. 16 – p.8, l. 16; fig 1a, 6; fig 2a, 6);

providing an input device configured by the document editing program to allow the user to initiate an operation (p.5, ll. 21 – p. 6, 8; p.19, ll. 1-5), such operation being of a type depending at least in part on the type or types of contact information of the

selected textual information (Fig. 1a, 1b, and 2a, 2b, steps 6, 8, 10, 12, 14; p. 7, ll. 18-21; p. 8, ll. 4-8; p. 8, ll. 15-16), the operation comprising identifying at least part of the selected textual information to use as a search term in order to find second information, of a specific type or types, associated in an information source with the search term (Figs 1a and 1b, 2a and 2b, steps 12, 14; p. 8, ll. 6-8; p. 8, ll. 15-16);

after identifying at least part of the selected information to use as a search term (p. 8, ll. 15-16; p.9, ll. 14-16), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (p.6, ll. 4-16), wherein the operation further comprises:

causing an electronic search in the information source (Figs. 1a and 1b, 2a and 2b, steps 12, 14; p.6, ll. 4-16), by an information management program external to the document editing program (p.1, ll. 17-20), for the search term in order to find whether the search term is included in the information source (p.8, ll. 6-8; p.8, ll. 15-17; p. 9, ll. 6-7); and performing an action having a type (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p.8, ll. 4-21),

wherein the type of action depends at least in part on whether the search term is included in the information source (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p. 8, ll. 8-21), and if the search term is so included, and if the information source includes the second information (Figs. 1a and 1b, steps 20, 22, 30, 32; p.8, ll. 8-13; p. 8, ll. 18-20), the action comprises causing insertion of at least part of the second information into the document (Figs. 1a and 1b, steps 20, 22, 30, 32; Figs . 4, 6, 7, 9-12, 15, p.8, ll. 8-13; p.8, ll. 18-20).

2. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Fig. 5 , 6, and 8; p.11, ll. 3-12; p. 12, ll. 6-7).

3. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein:

when (i) the information source includes the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Figs. 1a and 1b, steps 18, 22; Figs. 3 and 4; p.10, ll. 15-21).

4. At least one non-transitory computer readable medium according to claim 3, wherein the instructions further establish processes wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use for insertion into the document (Figs. 1a and 1b, steps 18, 20, 22; Figs. 3, 4, and 10; p.13, l. 13 - p. 14, l. 2).

5. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the selected textual information includes information that is not in the information source, the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Figs. 1a and 1b, steps 12, 18, 24, 16, steps 12, 14, 26, 28, 36; Fig. 6 and 7; p. 10, l. 7-11; p. 11, ll. 3-20).

6. At least one non-transitory computer readable medium according to claim 5, wherein the instructions further establish processes wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Figs. 1a and 1b, steps 12, 14, 26, 28; p. 10, ll. 6-7; p. 11, ll. 3-20).

7. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source is available over a network (p. 3, 11-13, p. 17, ll. 4-16).

8. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the information source includes an e-mail address (Fig. 1a, between steps 6 and 10; Fig. 12, 100; p. 6, ll. 15-18) and the operation includes causing an e-mail to be sent to the e-mail address (p. 7, ll. 7-11).

9. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Figs. 1a and 1b, steps 6, 12, 18, 20, steps 6, 14, 26, 30, 36; Fig. 9, 72; p. 12, l. 13 – p. 13, l. 2).

10. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the document editing program is a spreadsheet program (Figs. 14 and 15; p. 5, l. 20 – p. 6, l. 3; p. 2, l. 9 – p. 3, l. 4; p. 16, ll. 7-14).

11. At least one non-transitory computer readable medium according to claim 1, wherein the instructions further establish processes wherein the type of operation includes updating the document with information from the information source (Figs. 1a and 1b, steps 18, 22, steps 18, 20, 22; Figs. 3 and 4; p.10, ll. 15-21).

12. A method, for contact information handling (p.1, ll. 7-12), implemented by running a document editing program in a computer (p.1, ll. 7-12), the method comprising:

allowing a user to enter textual information into a document using the document editing program (Figs. 3 and 5; p.10, ll. 16-17, p. 11, 4-5; p.18, ll. 14-16);

displaying the textual information in the document electronically using the document editing program (Figs. 3 and 5; p.10, ll. 16-17, p. 11, 4-5; p.18, ll. 14-16);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (p.19,

ll. 6-10);

following user selection of textual information in the document (p.19, ll. 6-10), analyzing, by the document editing program (p. 18, ll. 9-11), the selected textual information (Fig. 1a, 4; 2a, 4; p.7, ll. 16-18) to determine if the selected textual information is regarded by the document editing program as contact information and what type or types of contact information the selected textual information is (p. 7, l. 16 – p.8, l. 16; fig 1a, 6; fig 2a, 6);

providing an input device configured by the document editing program to allow the user to initiate an operation (p.5, ll. 21 – p. 6, 8; p.19, ll. 1-5), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Fig. 1a, 1b, and 2a, 2b, steps 6, 8, 10, 12, 14; p. 7, ll. 18-21; p. 8, ll. 4-8; p. 8, ll. 15-16), the operation comprising identifying at least part of the selected textual information to use as a search term in order to find second information, of a specific type or types, associated in an information source with the search term (Figs 1a and 1b, 2a and 2b, steps 12, 14; p. 8, ll. 6-8; p. 8, ll. 15-16);

after identifying at least part of the selected information to use as a search term (p. 8, ll. 15-16; p.9, ll. 14-16), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (p.6, ll. 4-16), wherein the operation further comprises:

causing an electronic search in the information source (Figs. 1a and 1b, 2a and 2b, steps 12, 14; p.6, ll. 4-16), by an information management program external to the document editing program (p.1, ll. 17-20), for the search term in order to find whether the search term is included in the information source (p.8, ll. 6-8; p.8, ll. 15-17; p. 9, ll. 6-7); and performing an action having a type (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p.8, ll. 4-21),

wherein the type of action depends at least in part on whether the search term is included in the information source (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p. 8, ll. 8-21), and if the search term is so included, and if the information source includes the second information (Figs. 1a and 1b, steps 20, 22, 30, 32; p.8, ll. 8-13; p. 8, ll. 18-20), the action comprises causing insertion of at least part of the second information into the document (Figs. 1a and 1b, steps 20, 22, 30, 32;

Figs . 4, 6, 7, 9-12, 15, p.8, ll. 8-13; p.8, ll. 18-20).

13. A method according to claim 12, wherein:

when the information source does not include the search term, the action comprises causing indication to the user that the information source does not include the search term (Fig. 5 , 6, and 8; p.11, ll. 3-12; p. 12, ll. 6-7); and

when (i) the information source does include the search term, (ii) the selected textual information includes a name, (iii) the information source further includes the second information, and (iv) the second information includes an address, the action further comprises causing insertion of at least part of the address into the document (Figs. 1a and 1b, steps 18, 22; Figs. 3 and 4; p.10, ll. 15-21).

14. A method according to claim 12, wherein, when the second information includes a plurality of addresses, the operation further comprises allowing the user to choose one of the plurality of addresses to use insertion into the document (Figs. 1a and 1b, steps 18, 20, 22; Figs. 3, 4, and 10; p.13, l. 13 - p. 14, l. 2).

15. A method according to claim 12, wherein, when the selected textual information includes information that is not in the information source, and the operation further comprises:

allowing the user to cause storage of at least some of the selected textual information in the information source (Figs. 1a and 1b, steps 12, 18, 24, 16, steps 12, 14, 26, 28, 36; Fig. 6 and 7; p. 10, l. 2 – 6, p. 10, l. 21 – p. 11. l. 15);

wherein, when the selected textual information includes a name and an address and the information source does not include the name, allowing the user to cause storage comprises allowing the user to cause storage of at least part of the selected information in the information source as a new contact (Figs. 1a and 1b, steps 12, 18, 24, 16, steps 12, 14, 26, 28, 36; Fig. 6 and 7; p. 10, l. 7-11; p. 11, ll. 3-20).

16. A method according to claim 12, wherein the information source is available over a network (Figs. 1a and 1b, steps 12, 14, 26, 28; p. 10, ll. 6-7; p. 11, ll. 3-20).

17. A method according to claim 12, wherein the information source includes an e-mail address (Fig. 1a, between steps 6 and 10; Fig. 12, 100; p. 6, ll. 15-18) and the operation includes causing an e-mail to be sent to the e-mail address (p. 7, ll. 7-11).

18. A method according to claim 12, wherein, when the type or types of contact information includes a name, the operation includes causing display by the information management program of at least part of a contact information record in the information source corresponding to the name (Figs. 1a and 1b, steps 6, 12, 18, 20, steps 6, 14, 26, 30, 36; Fig. 9, 72; p. 12, l. 13 – p. 13, l. 2).

19. A method according to claim 12, wherein the type of operation includes updating the document with information from the information source (Figs. 1a and 1b, steps 18, 22, steps 18, 20, 22; Figs. 3 and 4; p.10, ll. 15-21).

20. An apparatus for contact information handling (p.1, ll. 7-12), comprising:

a processor (p. 16, ll. 17 – p. 17, ll. 11); and

a memory storing instructions executable by the processor to perform processes (p. 16, ll. 17 – p. 17, ll. 11) that include:

allowing a user to enter textual information into a document using the document editing program (Figs. 3 and 5; p.10, ll. 16-17, p. 11, 4-5; p.18, ll. 14-16);

displaying the textual information in the document electronically using the document editing program (Figs. 3 and 5; p.10, ll. 16-17, p. 11, 4-5; p.18, ll. 14-16);

allowing, in the document editing program, a user to select in the document at least a portion of the textual information while the textual information is displayed (p.19, ll. 6-10);

following user selection of textual information in the document (p.19, ll. 6-10), analyzing, by the document editing program (p. 18, ll. 9-11), the selected textual information (Fig. 1a, 4; 2a, 4; p.7, ll. 16-18) to determine if the selected

textual information is regarded by the document editing program as contact information and what type or types of contact information the selected textual information is (p. 7, l. 16 – p.8, l. 16; fig 1a, 6; fig 2a, 6);

providing an input device configured by the document editing program to allow the user to initiate an operation (p.5, ll. 21 – p. 6, 8; p.19, ll. 1-5), such operation being of a type depending at least in part on the type or types of contact information of the selected textual information (Fig. 1a, 1b, and 2a, 2b, steps 6, 8, 10, 12, 14; p. 7, ll. 18-21; p. 8, ll. 4-8; p. 8, ll. 15-16), the operation comprising identifying at least part of the selected textual information to use as a search term in order to find second information, of a specific type or types, associated in an information source with the search term (Figs 1a and 1b, 2a and 2b, steps 12, 14; p. 8, ll. 6-8; p. 8, ll. 15-16);

after identifying at least part of the selected information to use as a search term (p. 8, ll. 15-16; p.9, ll. 14-16), and in consequence of receipt by the document editing program of an execute command from the input device, performing the operation (p.6, ll. 4-16), wherein the operation further comprises:

causing an electronic search in the information source (Figs. 1a and 1b, 2a and 2b, steps 12, 14; p.6, ll. 4-16), by an information management program external to the document editing program (p.1, ll. 17-20), for the search term in order to find whether the search term is included in the information source (p.8, ll. 6-8; p.8, ll. 15-17; p. 9, ll. 6-7); and performing an action having a type (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p.8, ll. 4-21),

wherein the type of action depends at least in part on whether the search term is included in the information source (Figs. 1a and 1b, steps 20, 22, 24, 28, 30, 32; p. 8, ll. 8-21), and if the search term is so included, and if the information source includes the second information (Figs. 1a and 1b, steps 20, 22, 30, 32; p.8, ll. 8-13; p. 8, ll. 18-20), the action comprises causing insertion of at least part of the second information into the document (Figs. 1a and 1b, steps 20, 22, 30, 32; Figs . 4, 6, 7, 9-12, 15, p.8, ll. 8-13; p.8, ll. 18-20).

35 U.S.C. 112, Sixth Paragraph:

None of the pending claims utilize the means plus function format provided by 35 U.S.C. §112 ¶6. Accordingly, the corresponding rules of 71 Fed. Reg. 36323 do not apply to this application.

**9(F) Identification of References Disqualified as Prior Art under 35 USC 103(c):**

None of the cited references are disqualified as prior art under 35 U.S.C. 103(c).

Respectfully submitted,

/Jakub M. Michna, #61,033/
Jakub M. Michna
Registration No. 61,033
Attorney for Applicant

Sunstein Kann Murphy & Timbers LLP
125 Summer Street
Boston, MA 02110-1618
(617) 443-9292

03324/00106 1284118.1

# Exhibit 10

US006323853B1

(12) **United States Patent**
**Hedloy**

(10) **Patent No.: US 6,323,853 B1**
(45) **Date of Patent: Nov. 27, 2001**

(54) **METHOD, SYSTEM AND COMPUTER READABLE MEDIUM FOR ADDRESSING HANDLING FROM A COMPUTER PROGRAM**

(75) Inventor: **Atle Hedloy**, Stabekk (NO)

(73) Assignee: **Arendi AS**, Stabekk (NO)

( * ) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/189,626**

(22) Filed: **Nov. 10, 1998**

(30) **Foreign Application Priority Data**

Sep. 3, 1998 (NO) ..... 984066

(51) **Int. Cl.**[7] ..... **G06F 17/00**
(52) **U.S. Cl.** ..... **345/339**; 345/968; 707/530
(58) **Field of Search** ..... 345/326, 333, 345/335, 336, 339, 347, 348, 968, 354, 356; 707/1, 3, 500, 503, 505, 507, 530, 501, 513, 515

(56) **References Cited**

U.S. PATENT DOCUMENTS

| | | | |
|---|---|---|---|
| 5,226,117 | 7/1993 | Miklos | 345/356 |
| 5,331,555 | 7/1994 | Hashimoto et al. | 707/531 |
| 5,416,901 | 5/1995 | Torres | 345/348 |
| 5,491,783 | 2/1996 | Douglas et al. | 345/335 |
| 5,491,784 | 2/1996 | Douglas et al. | 345/352 |
| 5,500,859 | 3/1996 | Sharma et al. | 370/468 |
| 5,530,853 | 6/1996 | Schell et al. | 701/1 |
| 5,546,447 | 8/1996 | Skarbo et al. | 379/142 |
| 5,606,712 | 2/1997 | Hidaka | 712/1 |
| 5,640,565 | 6/1997 | Dickinson | 707/103 |
| 5,666,502 | 9/1997 | Capps | 345/352 |
| 5,708,804 | 1/1998 | Goodwin et al. | 707/3 |
| 5,724,597 * | 3/1998 | Cuthbertson et al. | 707/531 |
| 5,732,229 * | 3/1998 | Dickinson | 345/334 |
| 5,761,656 | 6/1998 | Ben-Shachar | 707/4 |
| 5,799,302 * | 8/1998 | Johnson et al. | 707/7 |
| 5,805,886 | 9/1998 | Skarbo et al. | 709/318 |
| 5,826,257 | 10/1998 | Snellling, Jr. | 707/4 |
| 5,835,089 | 11/1998 | Skarbo et al. | 345/335 |
| 5,859,636 | 1/1999 | Pandit | 345/335 |
| 5,873,107 * | 2/1999 | Borovoy et al. | 707/501 |
| 6,026,398 * | 2/2000 | Brown et al. | 707/5 |
| 6,085,201 * | 7/2000 | Tso | 707/505 |

* cited by examiner

*Primary Examiner*—Crescelle N. dela Torre
(74) *Attorney, Agent, or Firm*—Oblon, Spivak, McClelland, Maier & Neustadt, P.C.

(57) **ABSTRACT**

A method, system and computer readable medium for providing for providing a function item, such as a key, button, icon, or menu, tied to a user operation in a computer, whereby a single click on the function item in a window or program on a computer screen, or one single selection in a menu in a program, initiates retrieval of name and addresses and/or other person or company related information, while the user works simultaneously in another program, e.g., a word processor. The click on the function item initiates a program connected to the button to search a database or file available on or through the computer, containing the person, company or address related data, in order to look up data corresponding to what the user types, or partly typed, e.g., name and/or address in the word processor, the correct data from the database, data related to the typed data, e.g., the name of the person, company, or the traditional or electronic address, or other person, or company, or address related data, and alternatively the persons, companies, or addresses, are displayed and possibly entered into the word processor, if such related data exists.

**79 Claims, 17 Drawing Sheets**

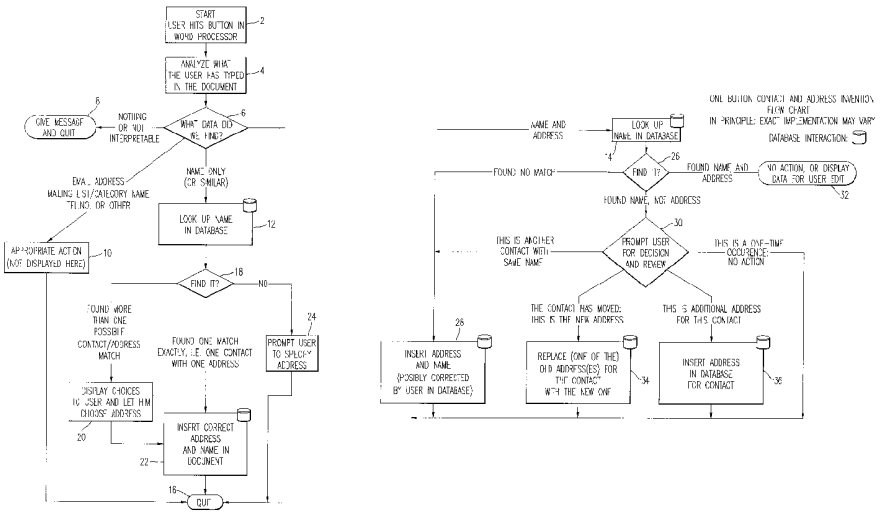
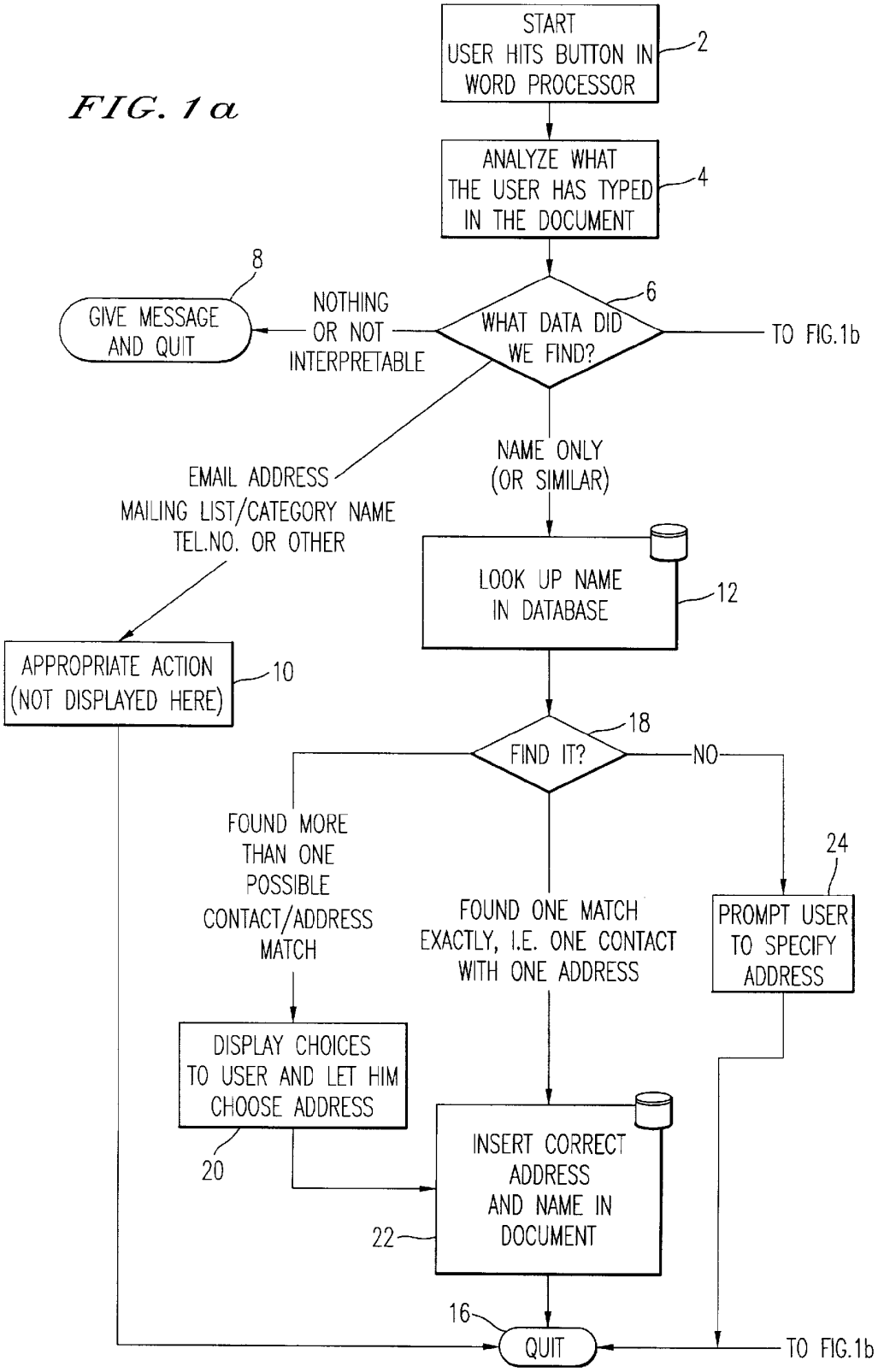


FIG. 1a


START
USER HITS BUTTON IN WORD PROCESSOR
2
ANALYZE WHAT THE USER HAS TYPED IN THE DOCUMENT
4
WHAT DATA DID WE FIND?
6
NOTHING OR NOT INTERPRETABLE
GIVE MESSAGE AND QUIT
8
TO FIG.1b
EMAIL ADDRESS
MAILING LIST/CATEGORY NAME
TEL.NO. OR OTHER
NAME ONLY (OR SIMILAR)
LOOK UP NAME IN DATABASE
12
APPROPRIATE ACTION (NOT DISPLAYED HERE)
10
FIND IT?
18
NO
FOUND MORE THAN ONE POSSIBLE CONTACT/ADDRESS MATCH
FOUND ONE MATCH EXACTLY, I.E. ONE CONTACT WITH ONE ADDRESS
PROMPT USER TO SPECIFY ADDRESS
24
DISPLAY CHOICES TO USER AND LET HIM CHOOSE ADDRESS
20
INSERT CORRECT ADDRESS AND NAME IN DOCUMENT
22
16
QUIT
TO FIG.1b

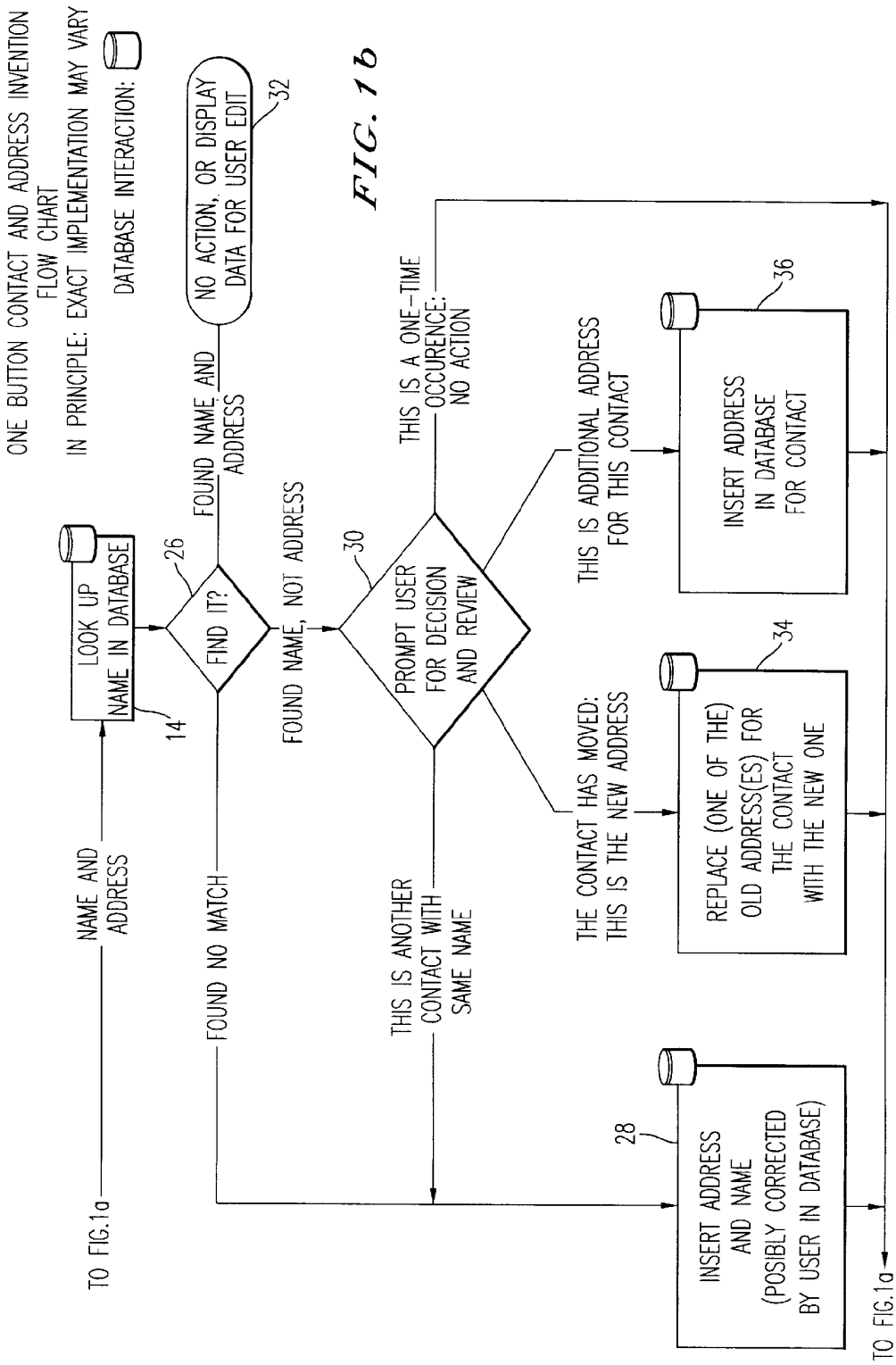
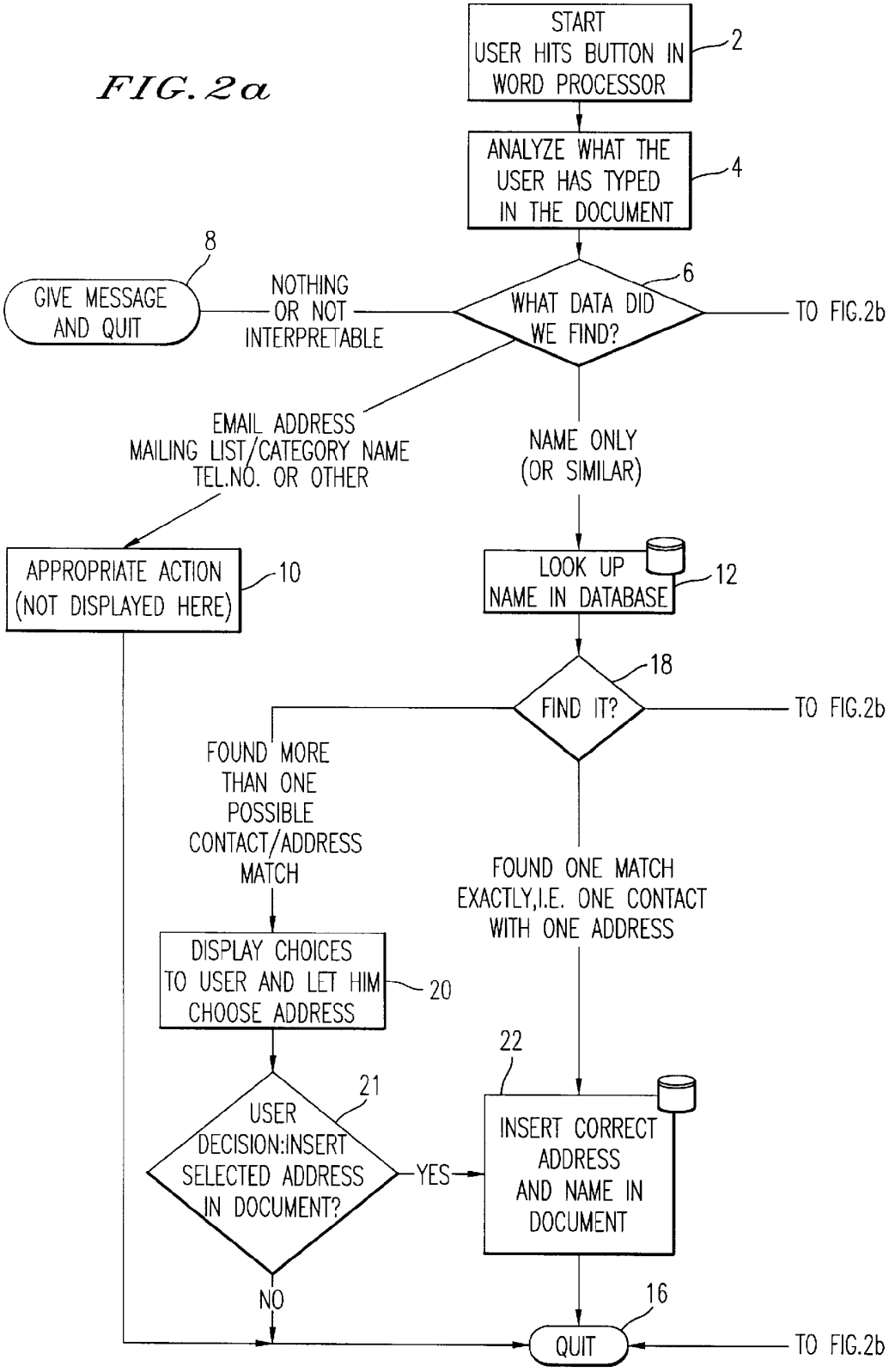

ONE BUTTON CONTACT AND ADDRESS INVENTION
FLOW CHART
IN PRINCIPLE: EXACT IMPLEMENTATION MAY VARY
DATABASE INTERACTION:
TO FIG.1a
NAME AND ADDRESS
LOOK UP NAME IN DATABASE
14
FIND IT?
26
FOUND NAME AND ADDRESS
NO ACTION, OR DISPLAY DATA FOR USER EDIT
32
FOUND NO MATCH
FOUND NAME, NOT ADDRESS
PROMPT USER FOR DECISION AND REVIEW
30
THIS IS A ONE-TIME OCCURENCE: NO ACTION
THIS IS ADDITIONAL ADDRESS FOR THIS CONTACT
INSERT ADDRESS IN DATABASE FOR CONTACT
36
THIS IS ANOTHER CONTACT WITH SAME NAME
THE CONTACT HAS MOVED: THIS IS THE NEW ADDRESS
REPLACE (ONE OF THE) OLD ADDRESS(ES) FOR THE CONTACT WITH THE NEW ONE
34
28
INSERT ADDRESS AND NAME (POSIBLY CORRECTED BY USER IN DATABASE)
TO FIG.1a

FIG. 1b

FIG. 2a


START
USER HITS BUTTON IN WORD PROCESSOR
2
ANALYZE WHAT THE USER HAS TYPED IN THE DOCUMENT
4
WHAT DATA DID WE FIND?
6
TO FIG.2b
NOTHING OR NOT INTERPRETABLE
8
GIVE MESSAGE AND QUIT
EMAIL ADDRESS MAILING LIST/CATEGORY NAME TEL.NO. OR OTHER
NAME ONLY (OR SIMILAR)
APPROPRIATE ACTION (NOT DISPLAYED HERE)
10
LOOK UP NAME IN DATABASE
12
FIND IT?
18
TO FIG.2b
FOUND MORE THAN ONE POSSIBLE CONTACT/ADDRESS MATCH
FOUND ONE MATCH EXACTLY,I.E. ONE CONTACT WITH ONE ADDRESS
DISPLAY CHOICES TO USER AND LET HIM CHOOSE ADDRESS
20
22
USER DECISION:INSERT SELECTED ADDRESS IN DOCUMENT?
21
YES
INSERT CORRECT ADDRESS AND NAME IN DOCUMENT
NO
16
QUIT
TO FIG.2b

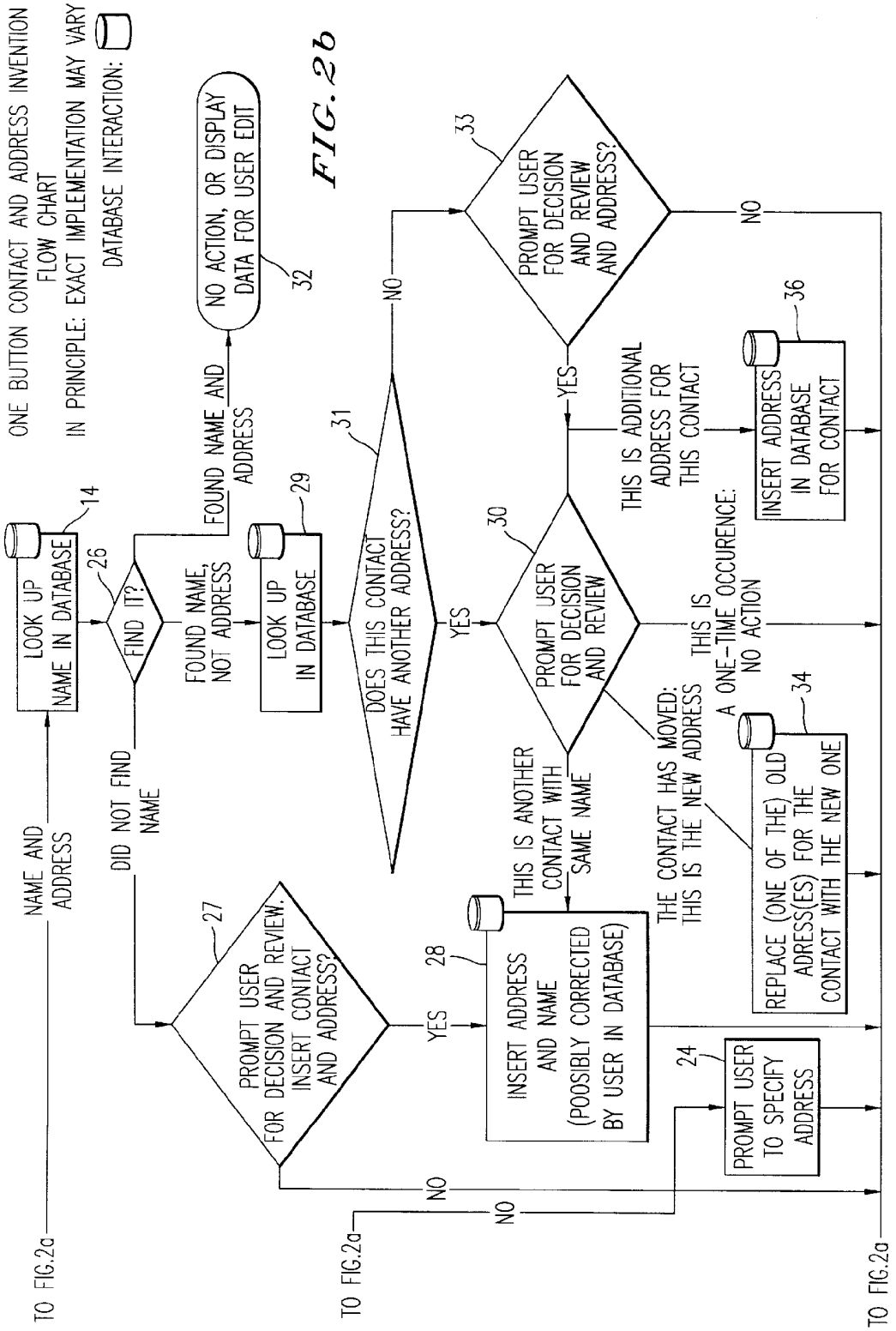

ONE BUTTON CONTACT AND ADDRESS INVENTION
FLOW CHART
IN PRINCIPLE: EXACT IMPLEMENTATION MAY VARY
DATABASE INTERACTION:
NAME AND ADDRESS
LOOK UP NAME IN DATABASE
14
FIND IT?
26
DID NOT FIND NAME
FOUND NAME, NOT ADDRESS
LOOK UP IN DATABASE
29
FOUND NAME AND ADDRESS
NO ACTION, OR DISPLAY DATA FOR USER EDIT
32
DOES THIS CONTACT HAVE ANOTHER ADDRESS?
31
NO
YES
PROMPT USER FOR DECISION AND REVIEW
30
PROMPT USER FOR DECISION AND REVIEW AND ADDRESS?
33
NO
YES
THIS IS ADDITIONAL ADDRESS FOR THIS CONTACT
INSERT ADDRESS IN DATABASE FOR CONTACT
36
THIS IS A ONE-TIME OCCURENCE: NO ACTION
THE CONTACT HAS MOVED: THIS IS THE NEW ADDRESS
REPLACE (ONE OF THE) OLD ADRESS(ES) FOR THE CONTACT WITH THE NEW ONE
34
THIS IS ANOTHER CONTACT WITH SAME NAME
PROMPT USER FOR DECISION AND REVIEW. INSERT CONTACT AND ADDRESS?
27
YES
28
INSERT ADDRESS AND NAME (POSSIBLY CORRECTED BY USER IN DATABASE)
NO
NO
PROMPT USER TO SPECIFY ADDRESS
24
TO FIG.2a
TO FIG.2a
TO FIG.2a

FIG. 2b

FIG. 3

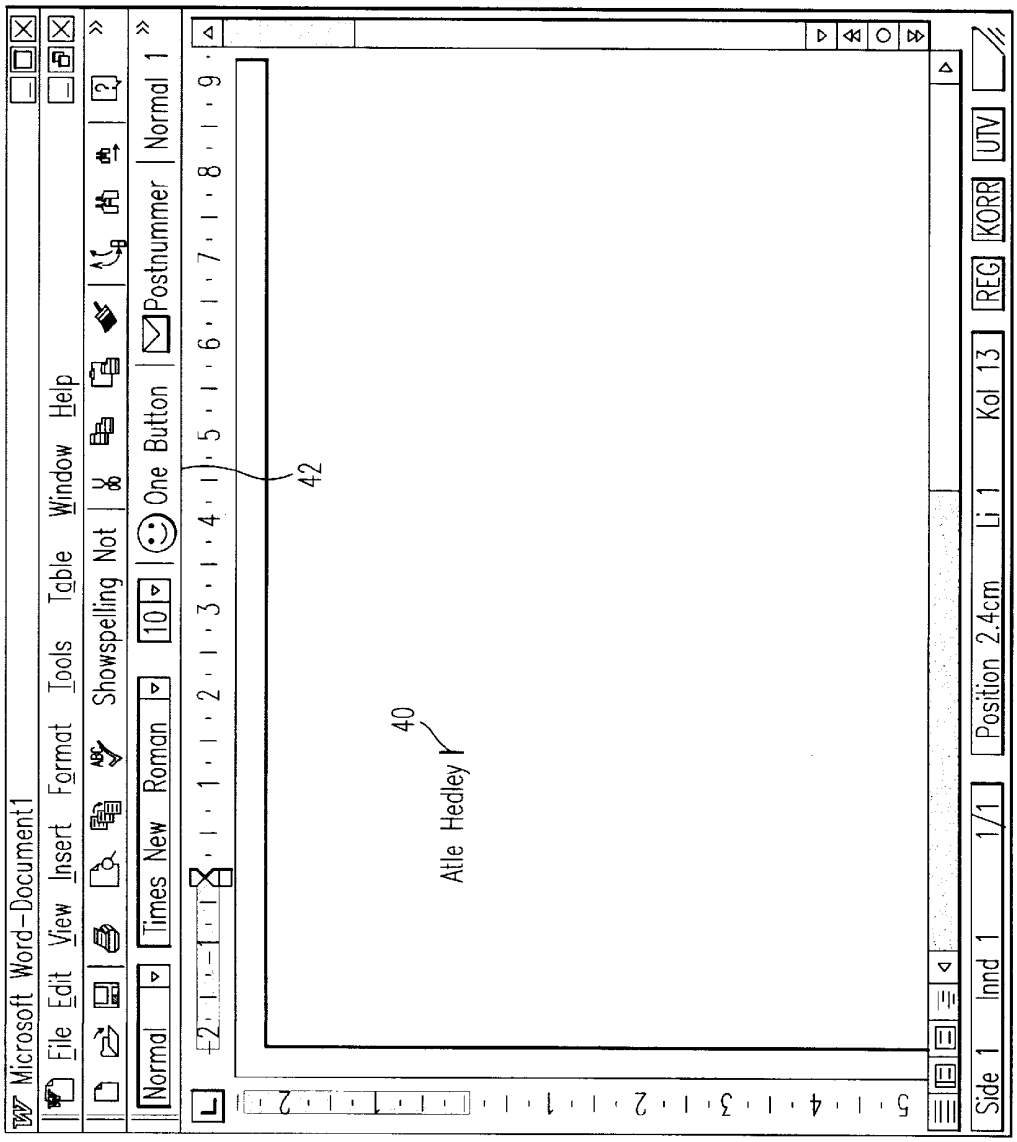

Microsoft Word-Document1
File Edit View Insert Format Tools Table Window Help
Normal
Times New Roman
10
Showspelling Not
One Button
Postnummer
Normal 1
42
40
Atle Hedley
Side 1
Innd 1
1/1
Position 2.4cm
Li 1
Kol 13
REG
KORR
UTV

FIG. 4

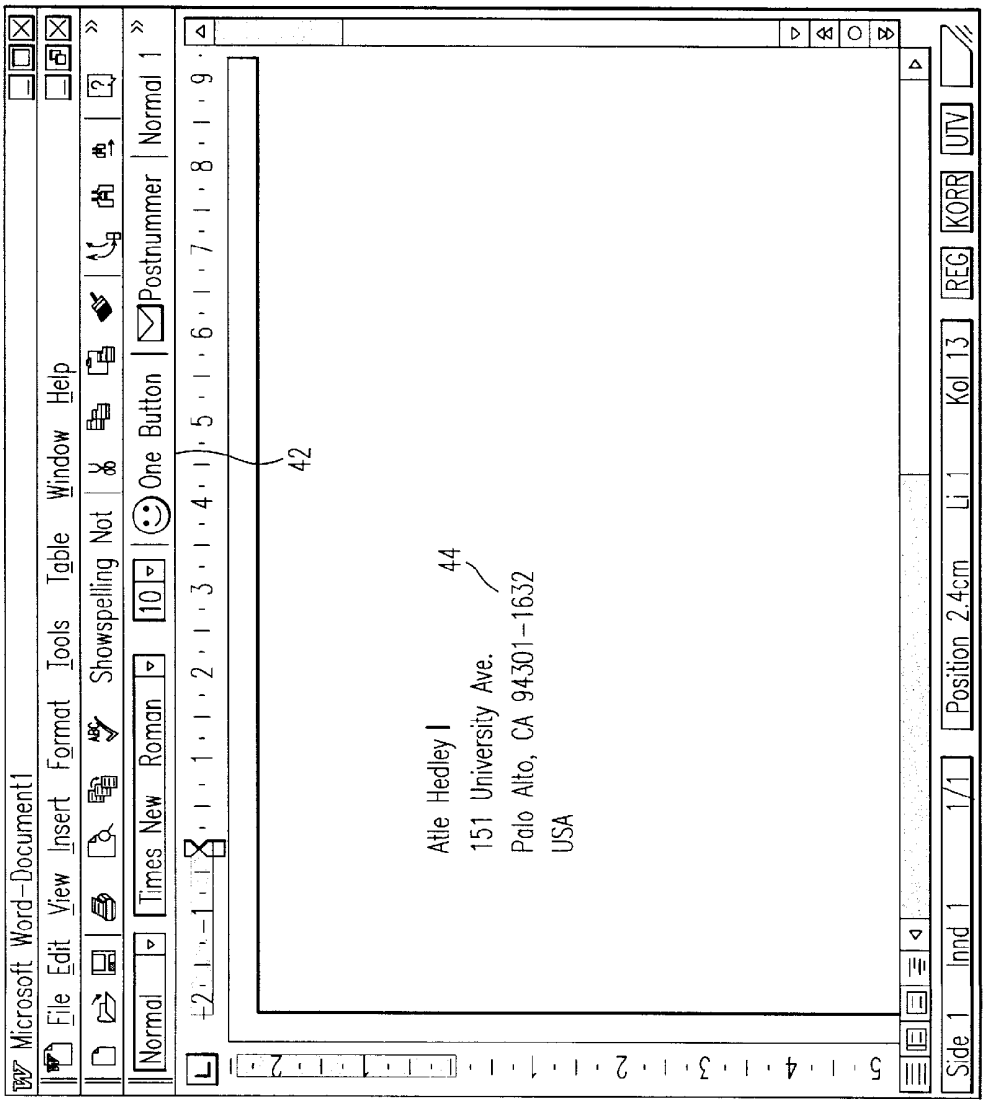

Microsoft Word-Document1
File Edit View Insert Format Tools Table Window Help
Normal
Times New Roman
10
Showspelling Not
One Button
Postnummer
Normal 1
42
44
Atle Hedley
151 University Ave.
Palo Alto, CA 94301-1632
USA
Side 1
Innd 1
1/1
Position 2.4cm
Li 1
Kol 13
REG
KORR
UTV

FIG. 5

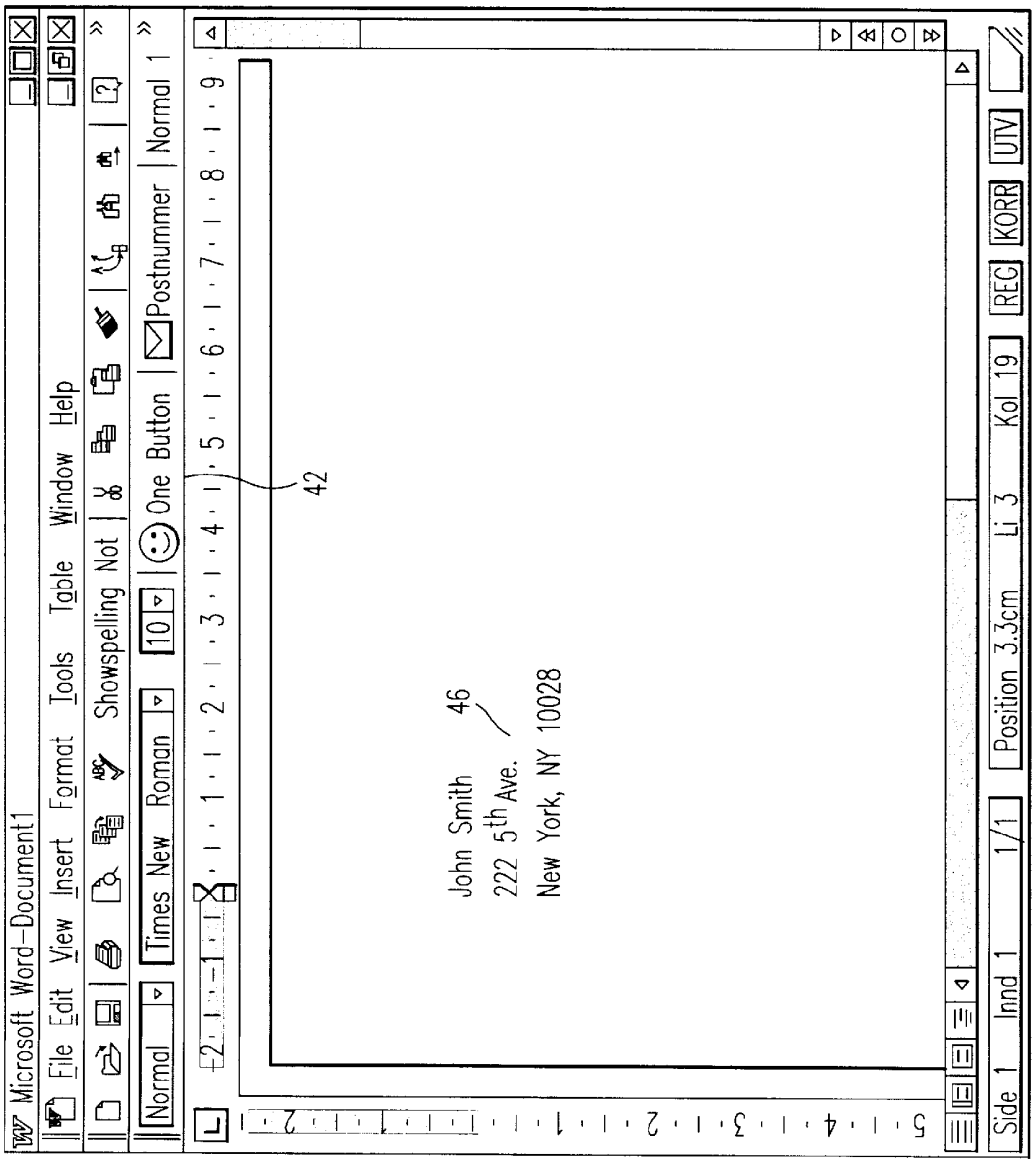

Microsoft Word-Document1
File Edit View Insert Format Tools Table Window Help
Normal
Times New Roman
10
One Button
Postnummer
Normal 1
Showspelling Not
42
John Smith 46
222 5th Ave.
New York, NY 10028
Side 1
Innd 1
1/1
Position 3.3cm
Li 3
Kol 19
REC
KORR
UTV

FIG. 6

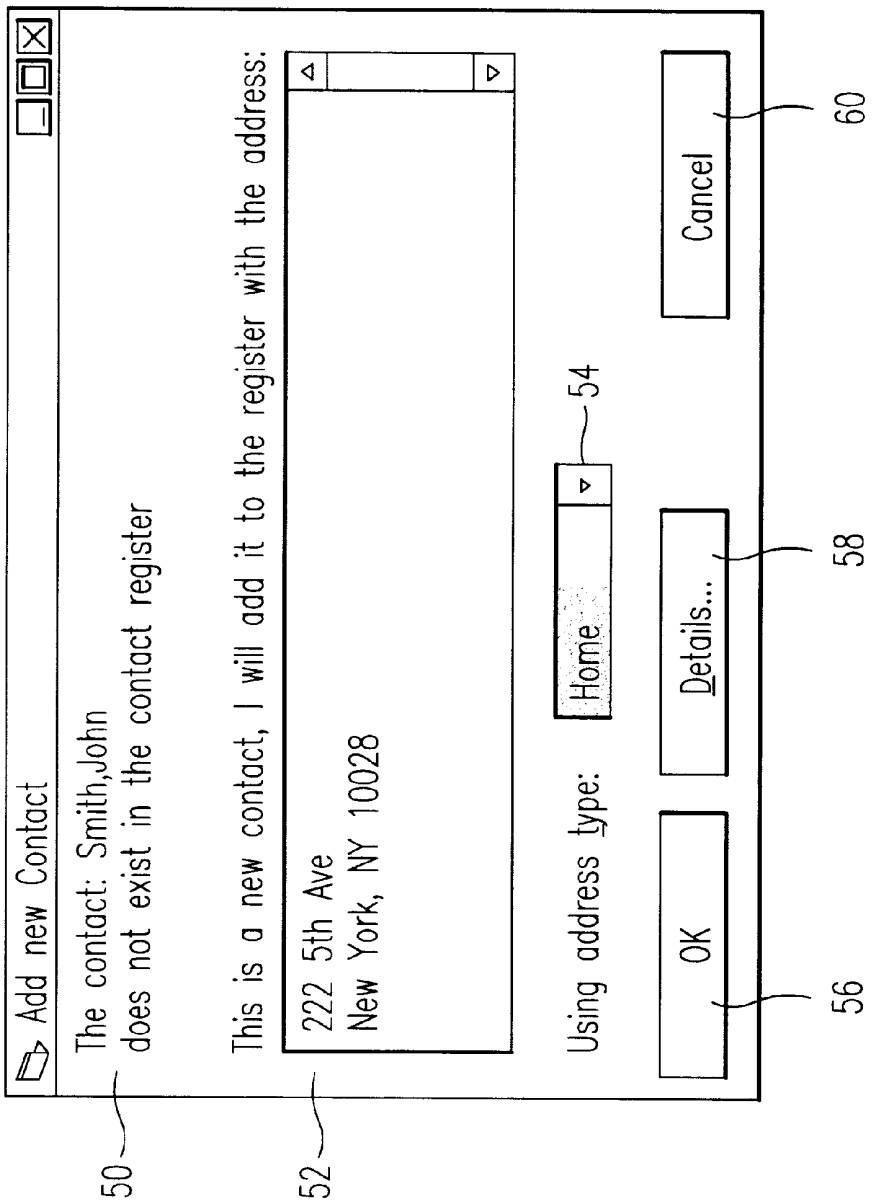

Add new Contact
The contact: Smith,John
does not exist in the contact register
This is a new contact, I will add it to the register with the address:
222 5th Ave
New York, NY 10028
Using address type:
Home
OK
Details...
Cancel
50
52
54
56
58
60

FIG. 7

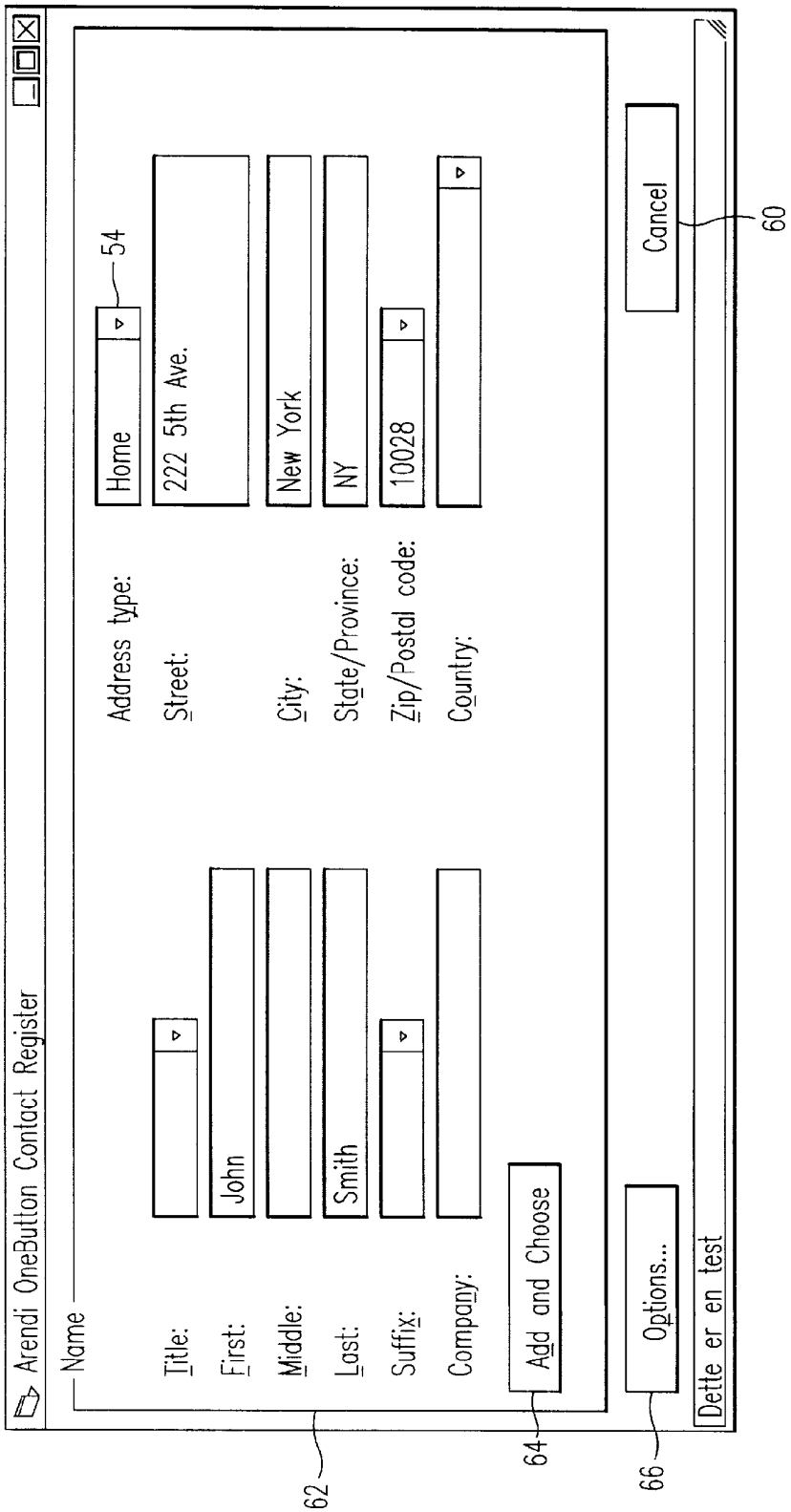

Arendi OneButton Contact Register
Name
Title:
First:
John
Middle:
Last:
Smith
Suffix:
Company:
Address type:
Home
54
Street:
222 5th Ave.
City:
New York
State/Province:
NY
Zip/Postal code:
10028
Country:
Add and Choose
Options...
Cancel
Dette er en test
62
64
66
60

FIG. 8

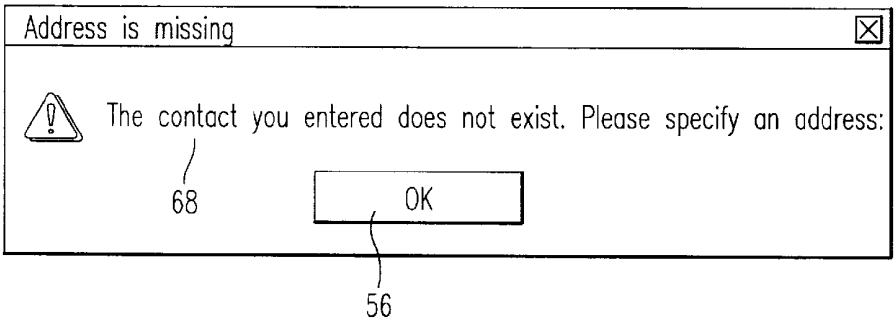

Address is missing
The contact you entered does not exist. Please specify an address:
68
OK
56

FIG. 9

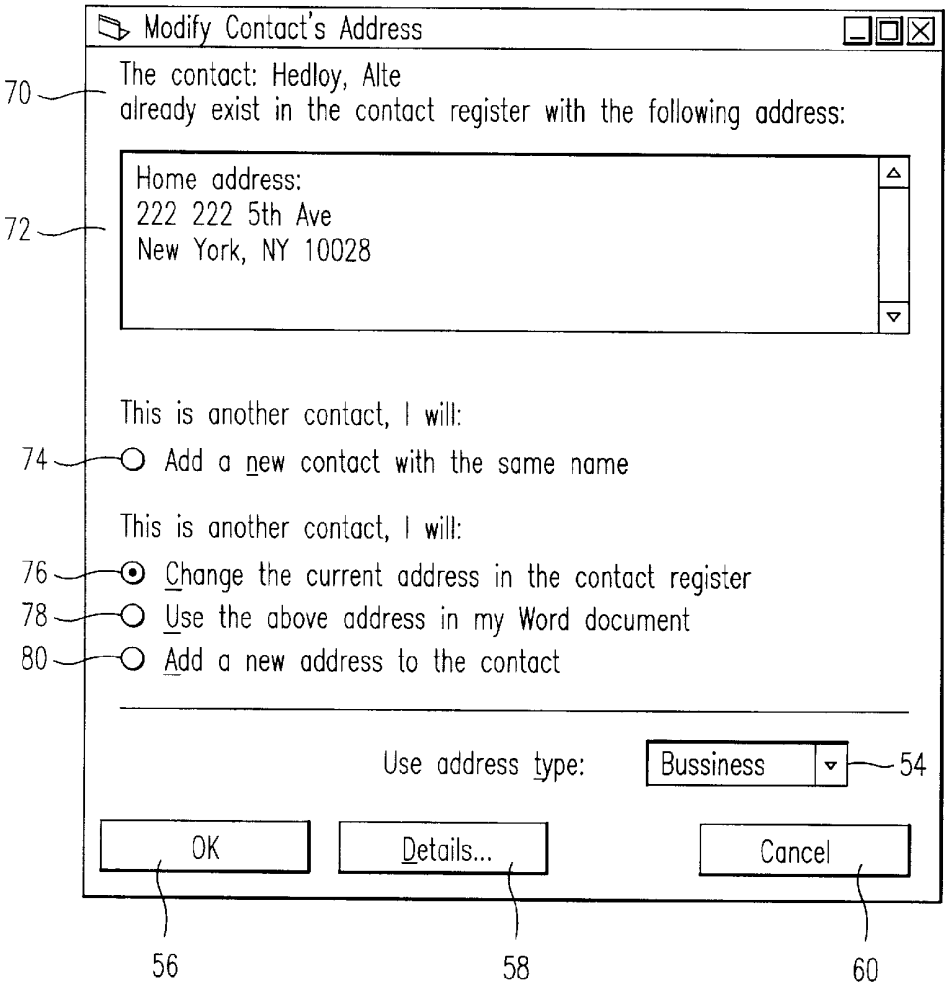

Modify Contact's Address
70
The contact: Hedloy, Alte
already exist in the contact register with the following address:
72
Home address:
222 222 5th Ave
New York, NY 10028
This is another contact, I will:
74
Add a new contact with the same name
This is another contact, I will:
76
Change the current address in the contact register
78
Use the above address in my Word document
80
Add a new address to the contact
Use address type:
Bussiness
54
OK
Details...
Cancel
56
58
60

FIG. 10

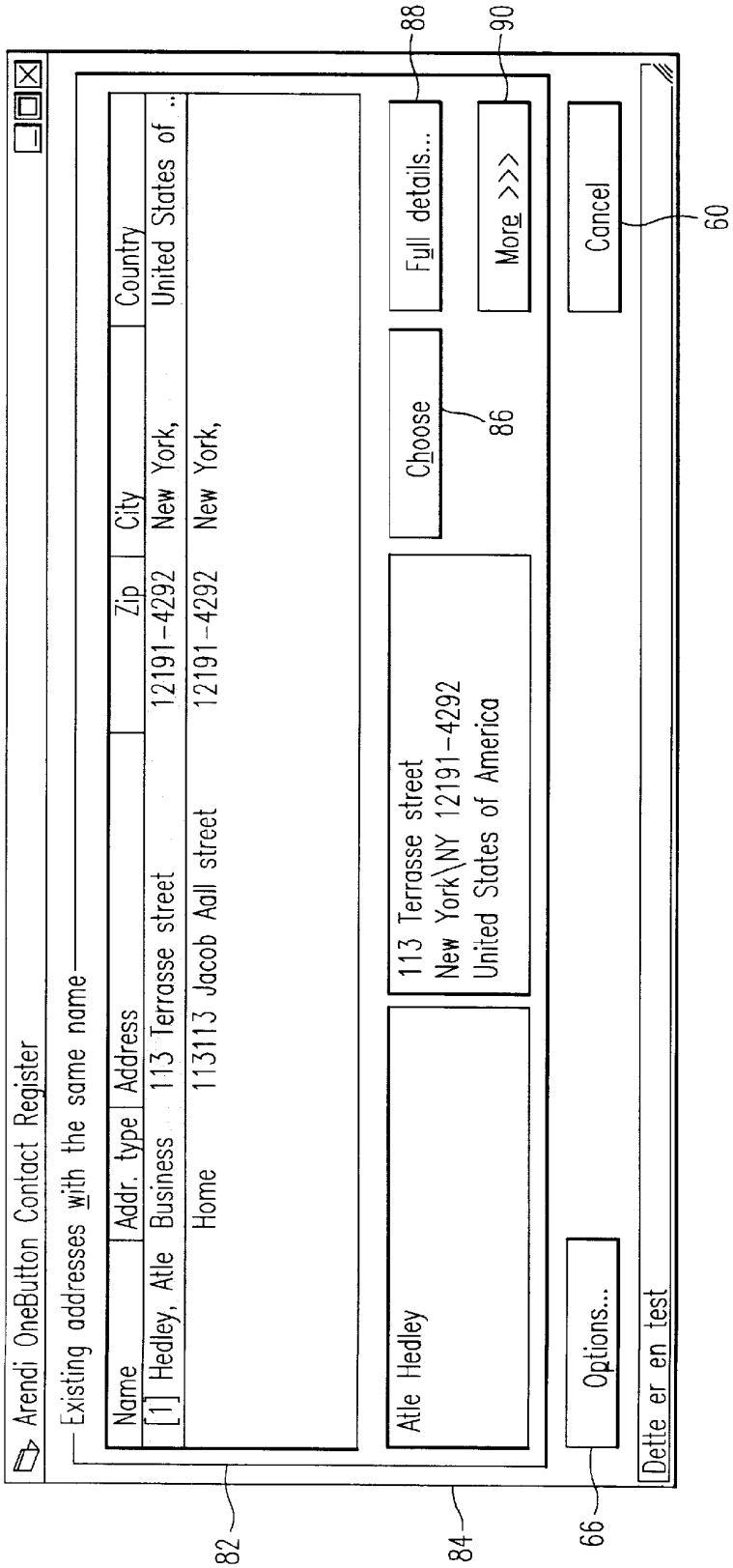

Arendi OneButton Contact Register
Existing addresses with the same name
Name
Addr. type
Address
Zip
City
Country
[1] Hedley, Atle
Business
113 Terrasse street
12191-4292
New York,
United States of ..
Home
113113 Jacob Aall street
12191-4292
New York,
Choose
Full details...
More >>>
Cancel
Atle Hedley
113 Terrasse street
New York\NY 12191-4292
United States of America
Options...
Dette er en test
82
84
66
86
88
90
60

FIG. 11

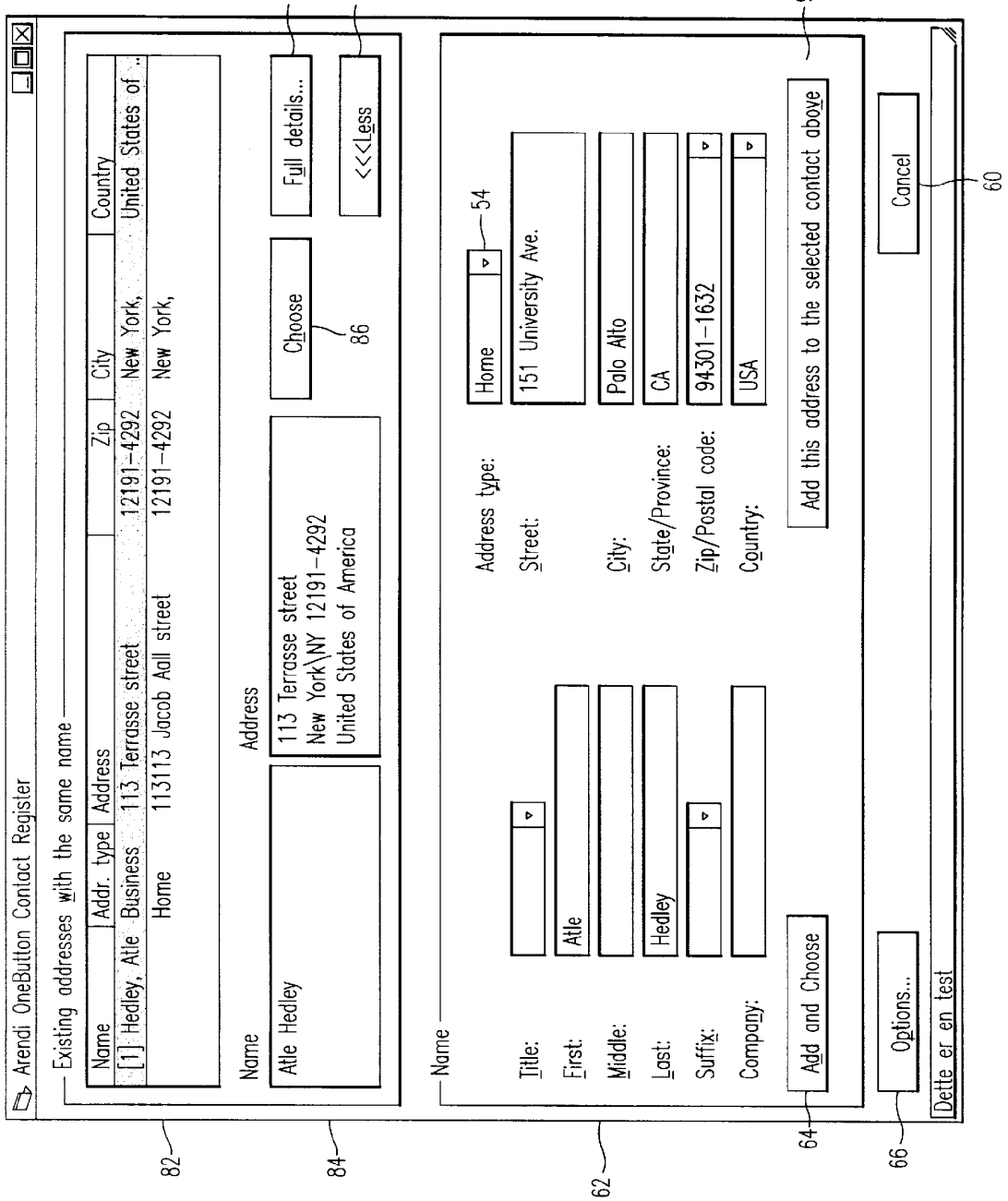

Arendi OneButton Contact Register
Existing addresses with the same name
Name
Addr. type
Address
Zip
City
Country
[1] Hedley, Atle
Business
113 Terrasse street
12191-4292
New York,
United States of ..
Home
113113 Jacob Aall street
12191-4292
New York,
Name
Address
Atle Hedley
113 Terrasse street
New York\NY 12191-4292
United States of America
Choose
86
88
Full details...
90
<<Less
Name
Address type:
Home
54
Street:
151 University Ave.
City:
Palo Alto
State/Province:
CA
Zip/Postal code:
94301-1632
Country:
USA
Add this address to the selected contact above
92
Title:
First:
Atle
Middle:
Last:
Hedley
Suffix:
Company:
Add and Choose
Options...
Cancel
60
Dette er en test
82
84
62
64
66

FIG. 12

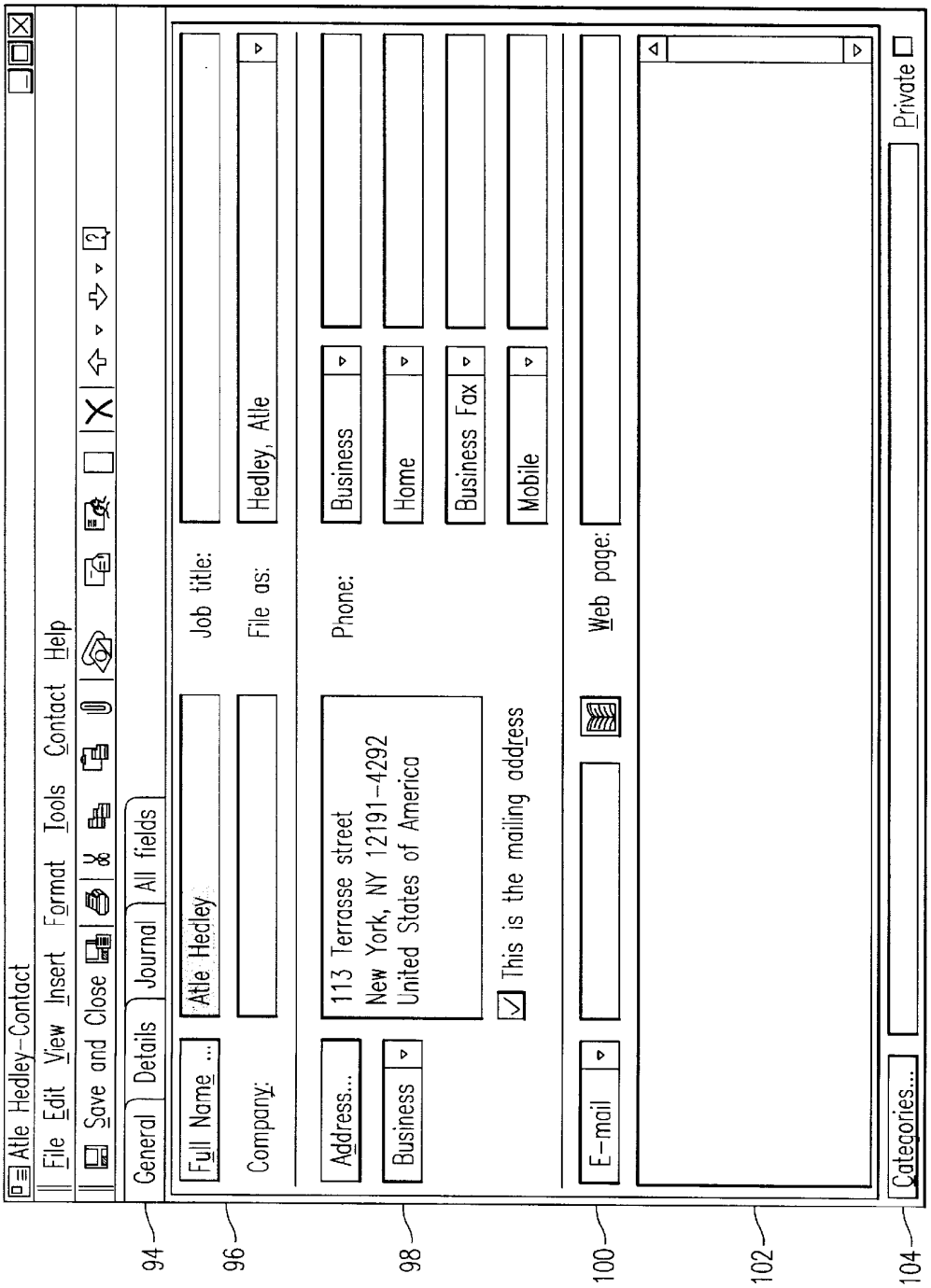

Atle Hedley - Contact
File Edit View Insert Format Tools Contact Help
Save and Close
General Details Journal All fields
Full Name ...
Atle Hedley
Job title:
Company:
File as:
Hedley, Atle
Address...
113 Terrasse street
New York, NY 12191-4292
United States of America
Business
This is the mailing address
Phone:
Business
Home
Business Fax
Mobile
E-mail
Web page:
Categories...
Private
94
96
98
100
102
104

*FIG. 13*

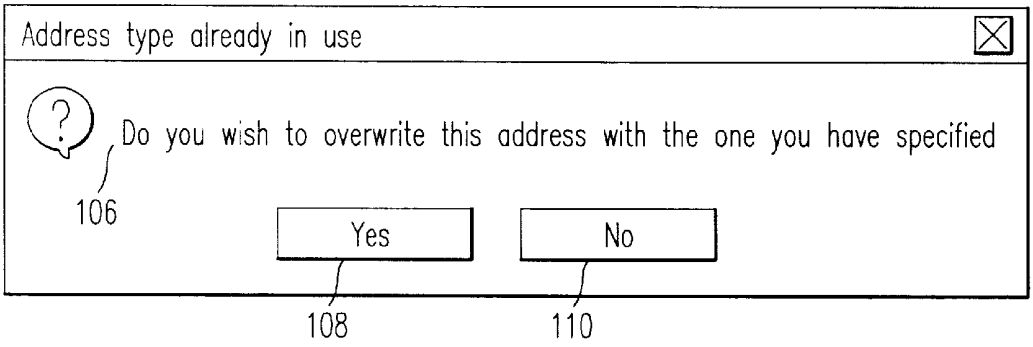

Address type already in use
Do you wish to overwrite this address with the one you have specified
106
Yes
No
108
110

FIG. 14

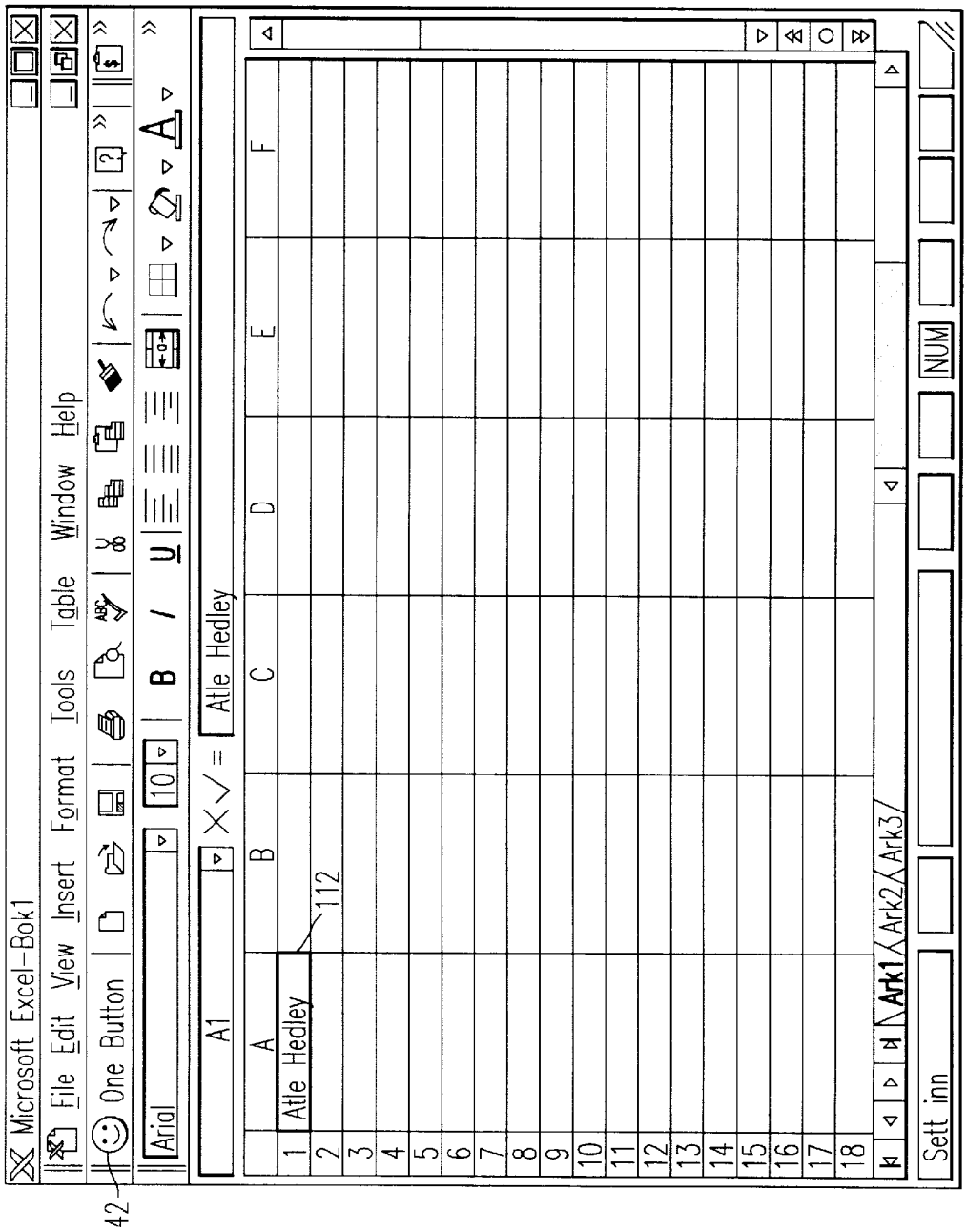

Microsoft Excel-Bok1
File Edit View Insert Format Tools Table Window Help
One Button
Arial
10
A1
Atle Hedley
A B C D E F
1 Atle Hedley
112
42
Ark1 Ark2 Ark3
Sett inn
NUM

FIG. 15

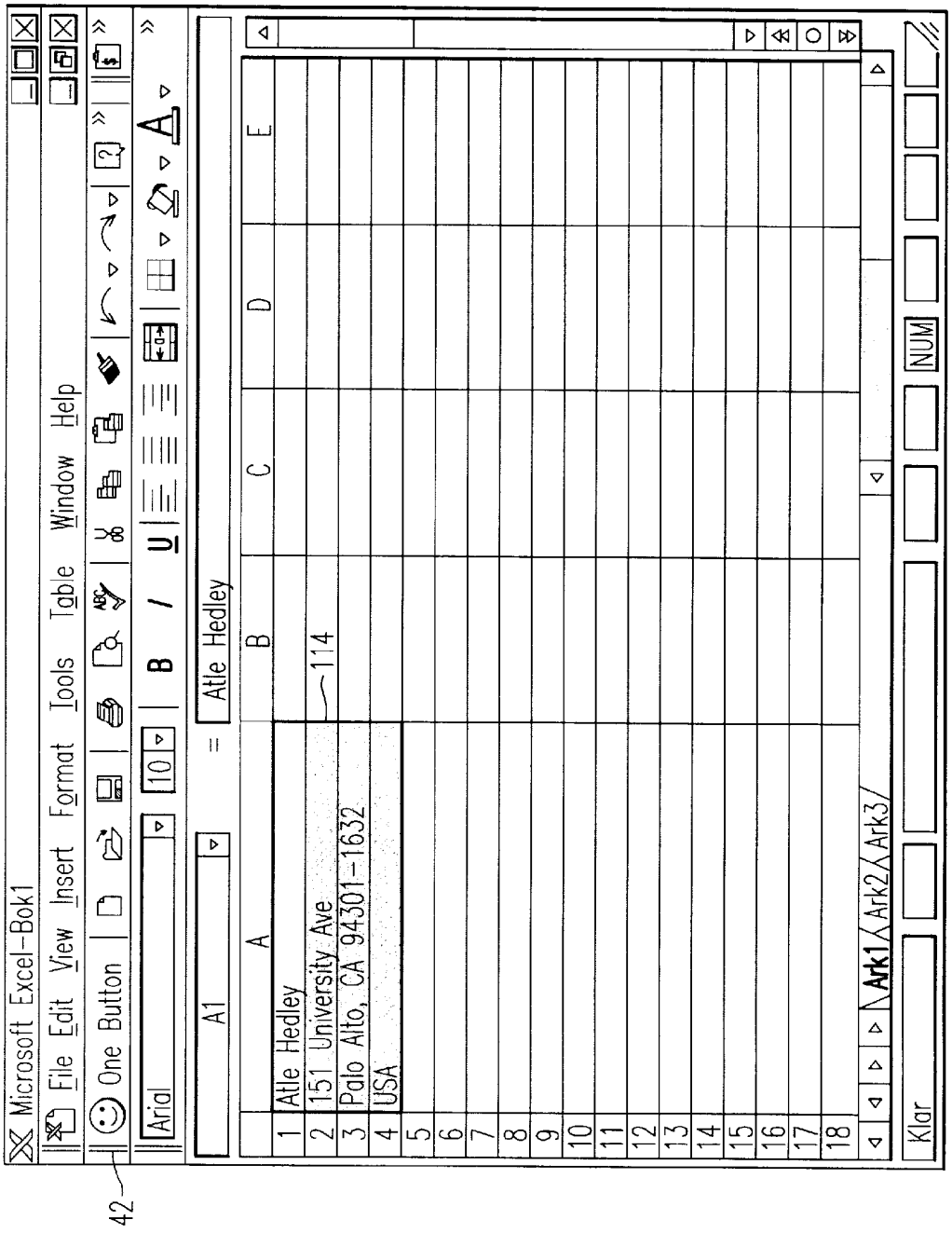
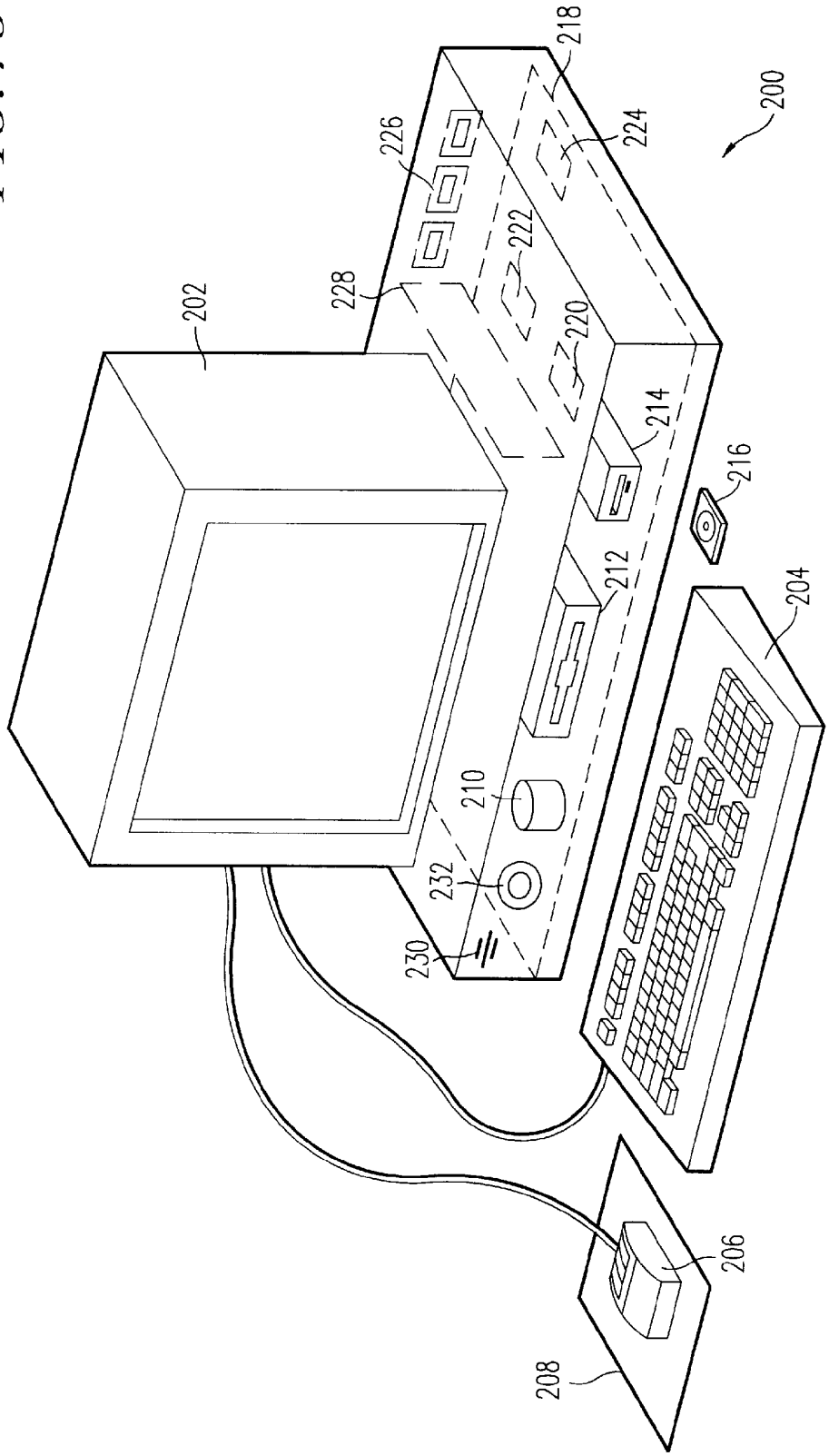

Microsoft Excel-Bok1
File Edit View Insert Format Tools Table Window Help
One Button
Arial
10
A1
= Atle Hedley
A
B
C
D
E
1 Atle Hedley
2 151 University Ave
3 Palo Alto, CA 94301-1632
4 USA
114
Ark1 Ark2 Ark3
Klar
NUM
42

FIG. 16


200
202
204
206
208
210
212
214
216
218
220
222
224
226
228
230
232

# METHOD, SYSTEM AND COMPUTER READABLE MEDIUM FOR ADDRESSING HANDLING FROM A COMPUTER PROGRAM

## BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to a method, system and computer readable medium for name and address handling (hereinafter called "address handling"), and more particularly to a touch screen, keyboard button, icon, menu, voice command device, etc. (hereinafter called "button") provided in a computer program, such as a word processing program, spreadsheet program, etc., and coupled to an information management source for providing address handling within a document created by the computer program.

2. Discussion of the Background

In recent years, with the advent of programs, such as word processors, spreadsheets, etc. (hereinafter called "word processors") users may require retrieval of information, such as name and address information, etc., for insertion into a document, such a letter, fax, etc., created with the word processor. Typically, the information is retrieved by the user from an information management source external to the word processor, such as a database program, contact management program, etc., or from the word processor itself, for insertion into the document. Examples of such word processors are WORD™, NOTEPAD™, EXCEL™, WORDPAD™, WORDPERFECT™, QUATROPRO™, AMIPRO™, etc., and examples of such information management sources are ACCESS™, OUTLOOK™, ORACLE™, DBASE™, RBASE™, CARDFILE™, etc.

However, the information in the database must constantly be updated by the user. This requires the user to learn how to use and have access to the database. In this case, a change in the information, such as change in an address or a name, etc., requires the user of the word processor to implement this change in the database, or alternatively, the change is made to the database centrally by a database administrator.

## SUMMARY OF THE INVENTION

Accordingly, an object of the present invention is to provide a method, system and computer readable medium for address handling within a computer program.

Another object of the present invention is to provide a method, system and computer readable medium for address handling within a computer program, such as a word processing program, spreadsheet program, etc.

Another object of the present invention is to provide a method, system and computer readable medium for address handling within a computer program, such as a word processing program, spreadsheet program, etc., using an input device provided in the computer program.

Another object of the present invention is to provide a method, system and computer readable medium for address handling within a computer program, such as a word processing program, spreadsheet program, etc., using an input device, such as a touch screen, keyboard button, icon, menu, voice command device, etc., provided in the computer program and coupled to an information management source.

Another object of the present invention is to provide a method, system and computer readable medium for address handling within a computer program, such as a word processing program, spreadsheet program, etc., using an input device, such as a touch screen, keyboard button, icon, menu, voice command device, etc., provided in the computer program and coupled to an information management source, such as a database program, contact management program, etc.



The above and other objects are achieved according to the present invention by providing a novel method, system and computer readable medium for providing a function item, such as a key, button, icon, or menu, tied to a user operation in a computer, whereby a single click on the function item in a window or program on a computer screen, or one single selection in a menu in a program, initiates retrieval of name and addresses and/or other person or company related information, while the user works simultaneously in another program, e.g., a word processor. The click on the function item initiates a program connected to the button to search a database or file available on or through the computer, containing the person, company or address related data, in order to look up data corresponding to what the user types, or partly typed, e.g., name and/or address in the word processor, the correct data from the database, data related to the typed data, e.g., the name of the person, company, or the traditional or electronic address, or other person, or company, or address related data, and alternatively the persons, companies, or addresses, are displayed and possibly entered into the word processor, if such related data exists.

The present invention also includes a computer readable medium storing program instructions by which the method of the invention can be performed when the stored program instructions are appropriately loaded into a computer, and a system for implementing the method of the invention.

## BRIEF DESCRIPTION OF THE DRAWINGS

A more complete appreciation of the invention and many of the attendant advantages thereof will be readily obtained as the same becomes better understood by reference to the following detailed description when considered in connection with the accompanying drawings wherein:

FIG. **1** is a flow chart illustrating a method for address handling within a computer program, according to an exemplary embodiment of the present invention;

FIG. **2** is a flow chart illustrating a method for address handling within a computer program, according to another exemplary embodiment of the present invention;

FIG. **3** is a screen shot illustrating the inputting of a name to be searched and an address handling button within a word processor, according to an exemplary embodiment of the present invention;

FIG. **4** is a screen shot illustrating a retrieved address in a word processor, according to an exemplary embodiment of the present invention;

FIG. **5** is a screen shot illustrating the inputting of a name and address to be searched and an address handling button within a word processor, according to an exemplary embodiment of the present invention;

FIG. **6** is a screen shot illustrating an add new contact message window, according to an exemplary embodiment of the present invention;

FIG. **7** is a screen shot illustrating a contact register message window, according to an exemplary embodiment of the present invention;

FIG. **8** is a screen shot illustrating an address missing message window, according to an exemplary embodiment of the present invention;

FIG. **9** is a screen shot illustrating a modify contact's address message window, according to an exemplary embodiment of the present invention;

FIG. **10** is a screen shot illustrating a select a contact address register message window, according to an exemplary embodiment of the present invention;

FIG. **11** is a screen shot illustrating a more detailed mode of registering an additional address for the contact register of FIG. **9**, according to an exemplary embodiment of the present invention;

FIG. **12** is a screen shot illustrating a contact management program window in a full detailed mode, according to an exemplary embodiment of the present invention;

FIG. **13** is a screen shot illustrating an address already in use message window, according to an exemplary embodiment of the present invention;

FIG. **14** is a screen shot illustrating the inputting of a name to be searched and an address handling button within a spreadsheet, according to an exemplary embodiment of the present invention;

FIG. **15** is a screen shot illustrating a retrieved address in a spreadsheet, according to an exemplary embodiment of the present invention; and

FIG. **16** is a schematic illustration of a general purpose computer for performing the processes of the present invention, according to an exemplary embodiment of the present invention.

## DESCRIPTION OF THE PREFERRED EMBODIMENTS

In an embodiment of the present invention, single button addressing is achieved by providing an input device, such as a touch screen, keyboard, icon, menu, voice command device etc. (hereinafter called "button"), in a computer program, such as a word processing program, spreadsheet program, etc. (hereinafter called "word processor"), for executing address handling therein.

Accordingly, in a word processor, the button is added and a user types information, such as an addressee's name, or a part of the name, etc. in a document created with the word processor, such as a letter, fax, etc., and then clicks, selects, commands, etc. the button via the appropriate input device, such as a touch screen button, keyboard button, icon, menu choice, voice command device, etc. A program then executes and retrieves the typed information from the document, and searches an information management source, such as a database, file, database program, contact management program, etc. (hereinafter called "database") to determine if the information, such as the name or part of the name typed and searched by the program exists in the database. If the program does not find stored information, such as a name, corresponding to the name or part of the name typed, the user is asked by the program whether the information, such as the name that was not found, should be added to the database. In addition, the user may enter any other information besides the name, such as addresses, businesses, telephone numbers, fax numbers, e-mail address, etc., so that this other information can be stored in the database for later use.

If the program finds name(s) and address(es) corresponding to the part of the addressee's name typed, this additional information is automatically entered into the user's word processor, optionally with a confirmation from the user that this is the correct data. If the typed address information does not correspond to data already stored in the database, after clicking on the button, the program, for example, lets the user decide: (1) if this is new data (e.g., a new address) for an existing contact; (2) if the stored data should be changed to what the user just typed; (3) if this is a new contact with the same name as one already entered into the database; or (4) if the typed address is only to be used once, and therefore not to be stored in the database at all. If, later, for example, a name with several address stored in the database is recalled, all addresses for this contact will be displayed, so that the correct address can be selected by the user.

The program may be extended to also store and retrieve other information, such as telephone numbers, fax numbers, e-mail addresses, etc. Once the program recalls the telephone numbers, fax numbers, e-mail addresses, etc., the user can command the program to send e-mails, faxes, etc. Similarly, if the user types in the name of a mailing list, the program create merge letters, group e-mails, etc.

Referring now to the drawings, wherein like reference numerals designate identical or corresponding parts throughout the several views, and more particularly to FIGS. **1** and **2** thereof, there is illustrated flow charts of single button addressing, according to exemplary embodiments of the present invention.

In FIG. **1**, after the user has inserted the address in the word processor, the user commands the button at step **2** and the program analyzes what the user has typed in the document at step **4**. At step **6**, the program decides what was found in the document and if the program found nothing in the document or what it found was un-interpretable the program goes to step **8** and outputs an appropriate message to the user and then quits at step **16**. The program analyzes what the user has typed in the document at step **4**, for example, by analyzing (i) paragraph/line separations/formatting, etc.; (ii) street, avenue, drive, lane, boulevard, city, state, zip code, country designators and abbreviations, etc.; (iii) Mr., Mrs., Sir, Madam, Jr., Sr. designators and abbreviations, etc.; (iv) Inc., Ltd., P.C., L.L.C, designators and abbreviations, etc.; and (v) a database of common male/female names, etc.

If the program finds an e-mail address mailing list/category name telephone number or other information, at step **10** an appropriate action is performed by the program and then the program execution quits at step **16**. If the program only finds a name or initials, or the like, the program looks up the name in the database at step **12** and at step **18** the program determines what was found. If the program finds more than one possible contact/address match, at step **20** the program displays menu choices to the user to let him choose an appropriate answer. Then at step **22** the program inserts a correct address and name in the document and then at step **16** the program quits execution. If the program finds one match exactly, i.e., one contact with one address, the program inserts the correct address and name in the document at step **22** and then quits execution at step **16**. If the program does not find a name in the database, at step **24** the program prompts the user to specify an address and then quits execution at step **16**. If the program at step **6** finds a name and an address, at step **14** the name is looked up in the database. Then, at step **26**, if no match is found, at step **28** the program inserts an address and a name which are possibly corrected by the user into the database and then quits execution at step **16**. If at step **26**, the name and address is found, at step **32** the program either takes no action or displays the data for the user to edit. If at step **26**, the name is found but not the address, the program prompts the user for a decision at step **30**. If the user decides that this another contact with a same name, the program goes to step **28**. If the user decides that this is a one time occurrence, no action is taken and the program quits at step **16**. If the user decides

that the contact has, for example, moved and that this is a new address, at step **34** one of the old addresses for the contact is replaced with the new one and the program quits at step **16**. If the user decides that this is an additional address for the contact, at step **36** the additional address is inserted into the database for that contact and execution quits at step **16**.

The flowchart shown in FIG. **2** is similar to the flowchart in FIG. **1**, except for some additional steps which will now be discussed. At step **6**, if the program only finds a name or a similar name then the name is looked up in the database at step **12**, then at step **18** if the program found more than one possible contact/address match, the program displays choices to the user to let him choose an address at step **20**. Then at step **21** the user decides whether to insert the selected address into the document. If the user does not decide to select the address into the document the program quits execution at step **16**. If the user decides to insert the selected address into the document, the program inserts the address and name into the document at step **22** and then quits at step **16**.

If the program finds a name and address in the database at step **6**, then at step **14** the program looks up the name in the database and at step **26** the program determines what it has found. If the program does not find the name at step **26**, at step **27** the program prompts the user for a decision and review and whether to insert the contact and address. If the user does not decide to insert the contact address, the program quits at step **16**. If the user decides to insert the contact address, at step **28** the program inserts the address and name which may be possibly corrected by the user or program in the database and then execution quits at step **16**.

If at step **26** the program finds a name and not an address, then at step **29** the name is looked up in the database. Then at step **31** the program decides whether this contact has another address. If the contact does not have another address, at step **33** the program prompts the user for a decision and review and whether to add the address. If the user does not want to add the address at step **33**, the program quits at step **16**. If the user wants to add the address at step **33** because this is an additional address for the contact, at step **36** the address is inserted in the database for the contact and execution quits at step **16**.

At step **30**, if the user decides that this is another contact with a same name, then the program goes to step **28**. If at step **30** the user decides that this is a one time occurrence, then the program quits at step **16**. If at step **30**, the user decides that the contact has, for example, moved, the program goes to step **34**. If at step **30**, the user decides that this is an additional address for the contact, at step **36** the program inserts the address in the database for the contact and then quits at step **16**.

Various exemplary screen shots which are generated during execution of the program, according to the present invention, will now be described with reference to FIGS. **3–15** and examples 1–7 as follows.

EXAMPLE 1

Retrieving an Existing Address from the Database:

FIG. **3** illustrates a starting point in word processor document, such as a WORD™ document, wherein the user has typed a name **40**. The user hits the button **42**, for example, marked "OneButton" and the program according to the present invention retrieves the name **40** from the document, searches a database for the name **40**, and inserts the retrieved address **44** associated with the name **40** into the document as shown in, for example, FIG. **4**.

The above example corresponds to steps **2**, **4**, **6**, **12**, **18**, **22** and **16** in the flow charts of FIGS. **1** and **2**.

EXAMPLE 2

Adding a New Contact to the Database:

FIG. **5** illustrates a starting point in word processor document, such as a WORD™ document, wherein the user has typed a name and address of a new contact **46**. The user commands the button **42**, for example, marked "OneButton," and the program according to the invention retrieves the new contact **46** from the document, searches a database for the name of the new contact **46** and generates a screen as shown in, for example, FIG. **6**. This screen includes a message **50** informing the user that the new contact does not exist in the database, a message **52** including the address retrieved from the document, an address type selection **54**, such as home, business, etc., and "OK," "Details," and "Cancel" buttons **56**, **58**, and **60**, respectively.

At this point, the user can cancel the operation by commanding the Cancel button **60**, ask the program to store data in the database and return to the document by commanding the OK button **56**, or check details before storing data into the database by commanding the Details button **58**. If the user commands the Details button **58**, as shown in, for example, FIG. **7**, a message screen is provided so that the user can review and edit data **62** and the selection **54**, store the data **62** and **54** in the database by commanding a "Add and Choose" button **64**, see more options by commanding an "Options" button **66**, or cancel the operation by commanding the Cancel button **60**.

The above example corresponds to steps **2**, **4**, **6**, **14**, **26**, **28** and **16** in the flow chart of FIG. **1** and steps **2**, **4**, **6**, **14**, **26**, **27**, **28** and **16** in the flow chart of FIG. **2**.

EXAMPLE 3

Try to Retrieve Existing Address, but Contact is not in Database:

FIG. **3** illustrates a starting point in word processor document, such as a WORD™ document, wherein the user has typed a name of a contact **40**. The user commands the button **42**, for example, marked "OneButton," and the program according to the present invention retrieves the name **40** from the document, searches a database for the name of the contact **40** and generates a screen as shown in, for example, FIG. **8**. This screen includes a message **68** informing the user that the contact does not exist in the database and to specify an address, and "OK" buttons **56**. At this point when the user commands the OK button **56**, the user returns to the document so that the contact's address can be included as in Example 2 above.

The above example corresponds to steps **2**, **4**, **6**, **12**, **18**, **24** and **16** in the flow charts of FIGS. **1** and **2**.

EXAMPLE 4

Adding a New Address for an Existing Contact (short version):

FIG. **4** illustrates a starting point in word processor document, such as a WORD™ document, wherein the user has typed a name and new address of an existing contact **44**. The user commands the button **42**, for example, marked "OneButton," and the program according to the present invention retrieves the existing contact **44** from the document, searches a database for the name of the existing contact **44** and generates a screen as shown in, for example, FIG. **9**. This screen includes a message **70** informing the user that the contact already exits in the database with an existing address, a message **72** including the existing address, add new contact with same name selection **74**, change existing

address selection **76**, use existing address in document selection **78**, add the new address to contact selection **80**, the address type selection **54**, such as home, business, etc., and the "OK," "Details," and "Cancel" buttons **56**, **58**, and **60** respectively. At this point, the user may select one of the four options **74–80**, and command the OK button **56** to execute the selected options. The user can also cancel the operation by commanding the Cancel button **60**, or check details before storing data into the database by commanding the Details button **58**.

The above example corresponds to steps **2**, **4**, **6**, **14**, **26**, **28**, **30**, **34**, **36**, and **16** in the flow chart of FIG. **1** and steps **2**, **4**, **6**, **14**, **26**, **29**, **31**, **30**, **28**, **34**, **36**, and **16** in the flow chart of FIG. **2**.

EXAMPLE 5

Selecting Between Several Possible Matching Addresses:

FIG. **3** illustrates a starting point in word processor document, such as a WORD™ document, wherein the user has typed a name and possibly address of at least one existing contact **40**. The user commands the button **42**, for example, marked "OneButton," and the program according to the present invention retrieves the existing contact **40** from the document, searches a database for the name of the existing contact **40** and generates a screen as shown in, for example, FIG. **10**. This screen includes a message informing the user that the name corresponds to several addresses and possible contacts which already exist in the database, with existing contacts and addresses for selection **82**, a message **84** including the full name and address for the contact that the user selects in **82**, the Options button **66**, a "Choose" button **86**, a "Full details" button **88**, a "More>>>" button **90**, and the Cancel button **60**. The above screen indicates to the user that at least one contact with the same name exists, and that there are more than one addresses and/or contacts that match.

At this point, the user may command the Choose button **86** to use the selected address and return to the document, or the user may command the More>>> button **90** to view how the program interpreted what the user typed in the word processor, and possibly change this data, wherein the program generates an updated screen as shown in, for example, FIG. **11**. The updated screen includes the data **62** which displays the name typed in the word processor as interpreted by the program, address fields, and the fields for the address type selection **54**, such as home, business, etc., which may be changed by the user before the program stores it in the database, the Add and Choose button **64**, a "<<<Less" button **90** corresponding to the More>>> button **90** for returning to the screen of FIG. **10**, and an "Add this address to the selected contact above" button **92**. The user might then command the Add this address to the selected contact above button **92** and the result in the word processor is illustrated in FIG. **4**. The user can also cancel the operation by commanding the Cancel button **60**, or command the add choose button **64** to add this name and address as a new contact and address, or open the database before storing data into the database by commanding a "Full details" button **88** as will be later described.

The above example corresponds to steps **2**, **4**, **6**, **12**, **18**, **20**, **22**, and **16** in the flow chart of FIG. **1** and steps **2**, **4**, **6**, **12**, **18**, **20**, **21**, **22**, and **16** in the flow chart of FIG. **2**.

EXAMPLE 6

Adding a New Address for an Existing Contact (long version):

FIG. **4** illustrates a starting point in word processor document, such as a WORD™ document, wherein the user has typed a name and new address of an existing contact **44**. The user commands the button **42**, for example, marked "OneButton," and the program according to the present invention retrieves the existing contact **44** from the document, searches a database for the name of the existing contact **44** and generates a screen as shown in, for example, FIG. **9**. As previously described, the screen includes a message **70** informing the user that the contact already exits in the database with an existing address, and the user may command the Details button **58** to see the details of the new address for potentially modify the details before they are stored in the database and the program generates a screen as shown in, for example, FIG. **10**. From this screen, the user may choose to use another address than the one he typed, and return to the document, or the user may command the "Full details" button **88** to enter a database program, such as OUTLOOK™, directly as shown in, for example, FIG. **12**. In FIG. **12**, the database program, such as OUTLOOK™, may include portions **94–104** for allowing the user to modify various pieces of data before they are stored in the database.

Alternatively, in the screen shown in FIG. **10**, the user may command the More>>> button **90** at which time the program generates the screen as shown in, for example, FIG. **11** and as previously described. In this screen, the user might then command the Add this address to the selected contact above button **92**. If the address typed is already in use, the program generates a screen including a message **106**, and "Yes" and "No" buttons, **108** and **110**, respectively, as shown in, for example, FIG. **13**. If the user hits the Yes button **108** the program overwrites the contact address with the address specified by the user (e.g., if the contact has moved) and the result in the word processor is shown in, for example, FIG. **4**.

The above example corresponds to steps **2**, **4**, **6**, **12**, **14**, **26**, **28**, **30**, **34**, **36**, and **16** in the flow chart of FIG. **1** and steps **2**, **4**, **6**, **12**, **14**, **26**, **29**, **31**, **30**, **28**, **34**, **36**, and **16** in the flow chart of FIG. **2**.

EXAMPLE 7

Spreadsheet Application:

FIG. **14** illustrates a starting point in word processor document, such as an EXCEL™ spreadsheet, wherein the user has typed a name **112**. The user hits the button **42**, for example, marked "OneButton," and the program according to the present invention retrieves the name **112** from the spreadsheet, searches a database for the name **112**, and inserts the retrieved address **114** into the spreadsheet as shown in, for example, FIG. **15**. Accordingly, the examples 1–6 apply not only to word processor documents, such as WORD™ documents, etc., but to other word processor documents, and spread sheets, such as EXCEL™ spreadsheets, etc.

The above example corresponds to steps **2**, **4**, **6**, **12**, **18**, **22** and **16** in the flow charts of FIGS. **1** and **2**.

FIG. **16** is a schematic illustration of a computer system for implementing the single button addressing according to the present invention. A computer **200** implements the method of the present invention, wherein the computer includes, for example, a display device **202**, such as a conventional display device or a touch screen monitor with a touch-screen interface, etc., a keyboard **204**, a pointing device **206**, a mouse pad or digitizing pad **208**, a hard disk **210**, or other fixed, high density media drives, connected using an appropriate device bus (e.g., a SCSI bus, an Enhanced IDE bus, an Ultra DMA bus, a PCI bus, etc.), a floppy drive **212**, a tape or CD ROM drive **214** with tape or CD media **216**, or other removable media devices, such as magneto-optical media, etc., and a mother board **218**. The

mother board **218** includes, for example, a processor **220**, a RAM **222**, and a ROM **224** (e.g., DRAM, ROM, EPROM, EEPROM, SRAM, SDRAM, and Flash RAM, etc.), I/O ports **226** which may be used to couple to external devices, networks, etc., (not shown), and optional special purpose logic devices (e.g., ASICs) or configurable logic devices (e.g., GAL and re-programmable FPGA) **228** for performing specialized hardware/software functions, such as sound processing, image processing, signal processing, neural network processing, object character recognition (OCR) processing, etc., a microphone **230**, and a speaker or speakers **232**.

As stated above, the system includes at least one computer readable medium, or alternatively, the computer readable medium may be accessed through various paths, such as networks, internet, drives, etc. Examples of computer readable media are compact discs, hard disks, floppy disks, tape, magneto-optical disks, PROMs (EPROM, EEPROM, Flash EPROM), DRAM, SRAM, SDRAM, etc. Stored on any one or on a combination of computer readable media, the present invention includes software for controlling both the hardware of the computer **200** and for enabling the computer **200** to interact with a human user. Such software may include, but is not limited to, device drivers, operating systems and user applications, such as development tools. Such computer readable media further includes the computer program product of the present invention for performing any of the processes according to the present invention, described above (see, e.g., FIGS. **1–15**). The computer code devices of the present invention can be any interpreted or executable code mechanism, including but not limited to scripts, interpreters, dynamic link libraries, Java classes, and complete executable programs, etc.

The invention may also be implemented by the preparation of application specific integrated circuits or by interconnecting an appropriate network of conventional component circuits, as will be readily apparent to those skilled in the art.

Address handling, according to this invention, is a significant simplification relative to existing methods, and requires little or no training on the part of a user, as correct addresses are retrieved with a minimal number of user commands, "clicks", keystrokes, etc. In addition, a program according to the present invention, can be programmed and created in most existing programming languages and be connected to most modern word processors. Therefore, according to the present invention, the process of creating and updating records in an address database is significantly simplified, since this may now be performed directly from the word processor.

Although the present invention is defined in terms of word processing documents, such as WORD™ documents and EXCEL™ spreadsheets, the present invention is applicable to all types of word processing documents, such as NOTEPAD™, WORDPAD™, WORDPERFECT™, QUATROPRO™, AMIPRO™, etc., as will be readily apparent to those skilled in the art.

Although the present invention is defined in terms of information management or database programs, such as OUTLOOK™, etc., the present invention is applicable to all types of information management or database programs, such as ACCESS™, ORACLE™, DBASE™, RBASE™, CARDFILE™, including "flat files," etc., as will be readily apparent to those skilled in the art.

Although the present invention is defined in terms of providing an input device, such as a button **42** in a word processor for address handling therein, the present invention may be practiced with all types of input devices, such as a touch screen, keyboard button, icon, menu, voice command device, etc., as will be readily apparent to those skilled in the art.

Although the present invention is defined in terms of a program retrieving information from a document before searching a database, the user may select the information in the document to be searched by the program in the database (e.g., by highlighting, selecting, italicizing, underlining, etc.), as will be readily apparent to those skilled in the art.

Although the present invention is defined in terms of a program retrieving a name or portion thereof from a document before searching a database, the program may retrieve an address or portion thereof from the document before searching the database and insert, correct, complete, etc., the retrieved address based on the information found in the database corresponding to the retrieved address or portion thereof, as will be readily apparent to those skilled in the art.

Obviously, numerous modifications and variations of the present invention are possible in light of the above teachings. It is therefore to be understood that within the scope of the appended claims, the invention may be practiced otherwise than as specifically described herein.

This application claims priority and contains subject matter related to Norwegian patent application No. 984066 filed on Sep. 3, 1998, the entire contents of which are hereby incorporated by reference.

What is claimed as new and desired to be secured by Letters Patent of the United States is:

**1**. A computerized method for information handling within a document created using an application program, the document including first information provided therein, the method comprising:

providing a record retrieval program;

providing an input device configured to enter an execute command which initiates a record retrieval from an information source using the record retrieval program;

upon a single entry of the execute command by means of the input device:

analyzing the document to determine if the first information is contained therein, and

if the first information is contained in the document, searching, using the record retrieval program, the information source for second information associated with the first information; and

when the information source includes second information associated with the first information, performing at least one of,

(a) displaying the second information,

(b) inserting the second information in the document, and

(c) completing the first information in the document based on the second information.

**2**. The method of claim **1**, further comprising one of the following steps:

storing the first information in the information source if no second information associated with the first information is found in the information source during said searching step,

changing the second information in the information source based on one of differences and similarities between the first information and the second information,

completing the first information in the document using the second information,

adding one of all and part of the first information to an existing record in the information source associated with one of all and part of the first information,

correcting the first information in the document using the second information,

adding information about said document to said information source, and

adding information about said document to said information source, said added information associated with said second information.

**3**. The method of claim **1**, wherein said second information includes at least one of a zip code, a city, a state, a county, a country, a street name, a house number, an apartment number, a telephone number, an email address and abbreviations or misspellings thereof, further comprising:

performing at least one of completing and correcting at least one of a zip code, a city, a state, a county, a country, a street name, a house number, an apartment number, a telephone number, an email address and abbreviations or misspellings thereof in the first information based on the second information.

**4**. The method of claim **1**, where in said second information includes at least one of a zip code, a city, a state, a county, a country, a street name, a house number, an apartment number, a telephone number, an email address and abbreviations or misspellings thereof, further comprising:

performing at least one of completing and correcting at least one of a zip code, a city, a state, a county, a country, a street name, a house number, an apartment number, a telephone number, an email address and abbreviations or misspellings thereof in the first information based on the second information automatically.

**5**. The method of claim **1**, where in said second information includes at least one of a zip code, a city, a state, a county, a country, a street name, a house number, an apartment number, a telephone number, an email address and abbreviations or misspellings thereof, further comprising:

performing at least one of completing and correcting at least one of a zip code, a city, a state, a county, a country, a street name, a house number, an apartment number, a telephone number, an email address and abbreviations or misspellings thereof in the first information based on the second information with assistance from a user.

**6**. The method of claim **1**, wherein said first information includes an identification of a list of addressees, further comprising:

addressing said document to all of said addressees based on the second information associated with said identification of said list of addressees.

**7**. The method of claim **1**, further comprising:

providing a user the option of making changes to the second information directly in the information source.

**8**. The method of claim **1**, wherein:

the step of using said application program comprises using said application program to enter first information comprising one of a person's name, a person's title, a person's name and address, a business name, a business name and address, a telephone number, and an email address, or a part thereof, into said document; and

the step of searching comprises searching, using the record retrieval program, the information source for second information comprising one of a person's name, a person's title, a person's name and address, a business name, a business name and address, a telephone number, and an email address, associated with the first information.

**9**. The method of claim **1**, wherein:

the step of providing an input device comprises providing an input device comprising one of a touch screen, a keyboard button, an icon, a menu and a voice command device, and configured to enter an execute command which initiates a record retrieval from an information source using the record retrieval program; and

the step of displaying the second information comprises displaying the second information comprising one of displaying a message screen with the second information and providing a voiced response of the second information.

**10**. The method of claim **1**, wherein the step of using said application program comprises:

using one of a word processing program and a spreadsheet program to enter first information into a respective one of a word processing document and a spreadsheet document.

**11**. The method of claim **1**, wherein the step of providing an input device comprises:

providing an input device configured to enter an execute command which initiates a record retrieval from an information source comprising at least one of a file, a database, a database program, a computer network, and a contact management program, using the record retrieval program.

**12**. The method of claim **1**, wherein said first information includes an identification of a list of addressees, further comprising:

creating copies of said document, each addressed to one of addressees in said list identified by said first information, based on said second information associated with said identification of said list of addressees.

**13**. The method of claim **1**, further comprising the step of indicating which part of information in said document is said first information.

**14**. The method of claim **1**, further comprising the step of automatically interpreting which part of information in said document is said first information.

**15**. A computer system configured to perform the steps recited in one of claims **1**–**14**.

**16**. A storage medium storing a program for performing the steps recited in one of claims **1**–**14**.

**17**. The method of claim **2**, wherein:

the step of using said application program comprises using said application program to enter first information comprising one of a person's name, a person's title, a person's name and address, a business name, a business name and address, a telephone number, and an email address, or a part thereof, into said document; and

the step of searching comprises searching, using the record retrieval program, the information source for second information comprising one of a person's name, a person's title, a person's name and address, a business name, a business name and address, a telephone number, and an email address, associated with the first information.

**18**. The method of claim **3**, wherein:

the step of using said application program comprises using said application program to enter first information comprising one of a person's name, a person's title, a person's name and address, a business name, a business name and address, a telephone number, and an email address, or a part thereof, into said document; and

the step of searching comprises searching, using the record retrieval program, the information source for

second information comprising one of a person's name, a person's title, a person's name and address, a business name, a business name and address, a telephone number, and an email address, associated with the first information.

**19**. The method of claim **4**, wherein:

the step of using said application program comprises using said application program to enter first information comprising one of a person's name, a person's title, a person's name and address, a business name, a business name and address, a telephone number, and an email address, or a part thereof, into said document; and

the step of searching comprises searching, using the record retrieval program, the information source for second information comprising one of a person's name, a person's title, a person's name and address, a business name, a business name and address, a telephone number, and an email address, associated with the first information.

**20**. The method of claim **5**, wherein:

the step of using said application program comprises using said application program to enter first information comprising one of a person's name, a person's title, a person's name and address, a business name, a business name and address, a telephone number, and an email address, or a part thereof, into said document; and

the step of searching comprises searching, using the record retrieval program, the information source for second information comprising one of a person's name, a person's title, a person's name and address, a business name, a business name and address, a telephone number, and an email address, associated with the first information.

**21**. The method of claim **6**, wherein:

the step of using said application program comprises using said application program to enter first information comprising one of a person's name, a person's title, a person's name and address, a business name, a business name and address, a telephone number, and an email address, or a part thereof, into said document; and

the step of searching comprises searching, using the record retrieval program, the information source for second information comprising one of a person's name, a person's title, a person's name and address, a business name, a business name and address, a telephone number, and an email address, associated with the first information.

**22**. The method of claim **7**, wherein:

the step of using said application program comprises using said application program to enter first information comprising one of a person's name, a person's title, a person's name and address, a business name, a business name and address, a telephone number, and an email address, or a part thereof, into said document; and

the step of searching comprises searching, using the record retrieval program, the information source for second information comprising one of a person's name, a person's title, a person's name and address, a business name, a business name and address, a telephone number, and an email address, associated with the first information.

**23**. The method of claim **2**, wherein:

the step of providing an input device comprises providing an input device comprising one of a touch screen, a keyboard button, an icon, a menu and a voice command device, and configured to enter an execute command which initiates a record retrieval from an information source using the record retrieval program; and

the step of displaying the second information comprises displaying the second information comprising one of displaying a message screen with the second information and providing a voiced response of the second information.

**24**. The method of claim **3**, wherein:

the step of providing an input device comprises providing an input device comprising one of a touch screen, a keyboard button, an icon, a menu and a voice command device, and configured to enter an execute command which initiates a record retrieval from an information source using the record retrieval program; and

the step of displaying the second information comprises displaying the second information comprising one of displaying a message screen with the second information and providing a voiced response of the second information.

**25**. The method of claim **4**, wherein:

the step of providing an input device comprises providing an input device comprising one of a touch screen, a keyboard button, an icon, a menu and a voice command device, and configured to enter an execute command which initiates a record retrieval from an information source using the record retrieval program; and

the step of displaying the second information comprises displaying the second information comprising one of displaying a message screen with the second information and providing a voiced response of the second information.

**26**. The method of claim **5**, wherein:

the step of providing an input device comprises providing an input device comprising one of a touch screen, a keyboard button, an icon, a menu and a voice command device, and configured to enter an execute command which initiates a record retrieval from an information source using the record retrieval program; and

the step of displaying the second information comprises displaying the second information comprising one of displaying a message screen with the second information and providing a voiced response of the second information.

**27**. The method of claim **6**, wherein:

the step of providing an input device comprises providing an input device comprising one of a touch screen, a keyboard button, an icon, a menu and a voice command device, and configured to enter an execute command which initiates a record retrieval from an information source using the record retrieval program; and

the step of displaying the second information comprises displaying the second information comprising one of displaying a message screen with the second information and providing a voiced response of the second information.

**28**. The method of claim **7**, wherein:

the step of providing an input device comprises providing an input device comprising one of a touch screen, a keyboard button, an icon, a menu and a voice command device, and configured to enter an execute command which initiates a record retrieval from an information source using the record retrieval program; and

the step of displaying the second information comprises displaying the second information comprising one of displaying a message screen with the second information and providing a voiced response of the second information.

**29**. The method of claim **8**, wherein:

the step of providing an input device comprises providing an input device comprising one of a touch screen, a keyboard button, an icon, a menu and a voice command device, and configured to enter an execute command which initiates a record retrieval from an information source using the record retrieval program; and

the step of displaying the second information comprises displaying the second information comprising one of displaying a message screen with the second information and providing a voiced response of the second information.

**30**. The method of claim **2**, wherein the step of using said application program comprises:

using one of a word processing program and a spreadsheet program to enter first information into a respective one of a word processing document and a spreadsheet document.

**31**. The method of claim **3**, wherein the step of using said application program comprises:

using one of a word processing program and a spreadsheet program to enter first information into a respective one of a word processing document and a spreadsheet document.

**32**. The method of claim **4**, wherein the step of using said application program comprises:

using one of a word processing program and a spreadsheet program to enter first information into a respective one of a word processing document and a spreadsheet document.

**33**. The method of claim **5**, wherein the step of using said application program comprises:

using one of a word processing program and a spreadsheet program to enter first information into a respective one of a word processing document and a spreadsheet document.

**34**. The method of claim **6**, wherein the step of using said application program comprises:

using one of a word processing program and a spreadsheet program to enter first information into a respective one of a word processing document and a spreadsheet document.

**35**. The method of claim **7**, wherein the step of using said application program comprises:

using one of a word processing program and a spreadsheet program to enter first information into a respective one of a word processing document and a spreadsheet document.

**36**. The method of claim **8**, wherein the step of using said application program comprises:

using one of a word processing program and a spreadsheet program to enter first information into a respective one of a word processing document and a spreadsheet document.

**37**. The method of claim **9**, wherein the step of using said application program comprises:

using one of a word processing program and a spreadsheet program to enter first information into a respective one of a word processing document and a spreadsheet document.

**38**. The method of claim **2**, wherein the step of providing an input device comprises:

providing an input device configured to enter an execute command which initiates a record retrieval from an information source comprising at least one of a file, a database, a database program, a computer network, and a contact management program, using the record retrieval program.

**39**. The method of claim **3**, wherein the step of providing an input device comprises:

providing an input device configured to enter an execute command which initiates a record retrieval from an information source comprising at least one of a file, a database, a database program, a computer network, and a contact management program, using the record retrieval program.

**40**. The method of claim **4**, wherein the step of providing an input device comprises:

providing an input device configured to enter an execute command which initiates a record retrieval from an information source comprising at least one of a file, a database, a database program, a computer network, and a contact management program, using the record retrieval program.

**41**. The method of claim **5**, wherein the step of providing an input device comprises:

providing an input device configured to enter an execute command which initiates a record retrieval from an information source comprising at least one of a file, a database, a database program, a computer network, and a contact management program, using the record retrieval program.

**42**. The method of claim **6**, wherein the step of providing an input device comprises:

providing an input device configured to enter an execute command which initiates a record retrieval from an information source comprising at least one of a file, a database, a database program, a computer network, and a contact management program, using the record retrieval program.

**43**. The method of claim **7**, wherein the step of providing an input device comprises:

providing an input device configured to enter an execute command which initiates a record retrieval from an information source comprising at least one of a file, a database, a database program, a computer network, and a contact management program, using the record retrieval program.

**44**. The method of claim **8**, wherein the step of providing an input device comprises:

providing an input device configured to enter an execute command which initiates a record retrieval from an information source comprising at least one of a file, a database, a database program, a computer network, and a contact management program, using the record retrieval program.

**45**. The method of claim **9**, wherein the step of providing an input device comprises:

providing an input device configured to enter an execute command which initiates a record retrieval from an information source comprising at least one of a file, a database, a database program, a computer network, and a contact management program, using the record retrieval program.

**46**. The method of claim **10**, wherein the step of providing an input device comprises:

providing an input device configured to enter an execute command which initiates a record retrieval from an information source comprising at least one of a file, a database, a database program, a computer network, and a contact management program, using the record retrieval program.

**47**. The method of claim **2**, wherein said first information includes an identification of a list of addressees, further comprising:

creating copies of said document, each addressed to one of addressees in said list identified by said first information, bascd on said second information associated with said identification of said list of addressees.

**48**. The method of claim **3**, wherein said first information includes an identification of a list of addressees, further comprising:

creating copies of said document, each addressed to one of addressees in said list identified by said first information, based on said second information associated with said identification of said list of addressees.

**49**. The method of claim **4**, wherein said first information includes an identification of a list of addressees, further comprising:

creating copies of said document, each addressed to one of addressees in said list identified by said first information, based on said second information associated with said identification of said list of addressees.

**50**. The method of claim **5**, wherein said first information includes an identification of a list of addressees, further comprising:

creating copies of said document, each addressed to one of addressees in said list identified by said first information, based on said second information associated with said identification of said list of addressees.

**51**. The method of claim **6**, wherein said first information includes an identification of a list of addressees, further comprising:

creating copies of said document, each addressed to one of addressees in said list identified by said first information, based on said second information associated with said identification of said list of addressees.

**52**. The method of claim **7**, wherein said first information includes an identification of a list of addressees, further comprising:

creating copies of said document, each addressed to one of addressees in said list identified by said first information, based on said second information associated with said identification of said list of addressees.

**53**. The method of claim **8**, wherein said first information includes an identification of a list of addressees, further comprising:

creating copies of said document, each addressed to one of addressees in said list identified by said first information, based on said second information associated with said identification of said list of addressees.

**54**. The method of claim **9**, wherein said first information includes an identification of a list of addressees, further comprising:

creating copies of said document, each addressed to one of addressees in said list identified by said first information, based on said second information associated with said identification of said list of addressees.

**55**. The method of claim **10**, wherein said first information includes an identification of a list of addressees, further comprising:

creating copies of said document, each addressed to one of addressees in said list identified by said first information, based on said second information associated with said identification of said list of addressees.

**56**. The method of claim **11**, wherein said first information includes an identification of a list of addressees, further comprising:

creating copies of said document, each addressed to one of addressees in said list identified by said first information, based on said second information associated with said identification of said list of addressees.

**57**. The method of claim **2**, further comprising the step of indicating which part of information in said document is said first information.

**58**. The method of claim **3**, further comprising the step of indicating which part of information in said document is said first information.

**59**. The method of claim **4**, further comprising the step of indicating which part of information in said document is said first information.

**60**. The method of claim **5**, further comprising the step of indicating which part of information in said document is said first information.

**61**. The method of claim **6**, further comprising the step of indicating which part of information in said document is said first information.

**62**. The method of claim **7**, further comprising the step of indicating which part of information in said document is said first information.

**63**. The method of claim **8**, further comprising the step of indicating which part of information in said document is said first information.

**64**. The method of claim **9**, further comprising the step of indicating which part of information in said document is said first information.

**65**. The method of claim **10**, further comprising the step of indicating which part of information in said document is said first information.

**66**. The method of claim **11**, further comprising the step of indicating which part of information in said document is said first information.

**67**. The method of claim **12**, further comprising the step of indicating which part of information in said document is said first information.

**68**. The method of claim **2**, further comprising the step of automatically interpreting which part of information in said document is said first information.

**69**. The method of claim **3**, further comprising the step of automatically interpreting which part of information in said document is said first information.

**70**. The method of claim **4**, further comprising the step of automatically interpreting which part of information in said document is said first information.

**71**. The method of claim **5**, further comprising the step of automatically interpreting which part of information in said document is said first information.

**72**. The method of claim **6**, further comprising the step of automatically interpreting which part of information in said document is said first information.

**73**. The method of claim **7**, further comprising the step of automatically interpreting which part of information in said document is said first information.

**74**. The method of claim **8**, further comprising the step of automatically interpreting which part of information in said document is said first information.

**75**. The method of claim **9**, further comprising the step of automatically interpreting which part of information in said document is said first information.

**76**. The method of claim **10**, further comprising the step of automatically interpreting which part of information in said document is said first information.

**77**. The method of claim **11**, further comprising the step of automatically interpreting which part of information in said document is said first information.

**78**. The method of claim **12**, further comprising the step of automatically interpreting which part of information in said document is said first information.

**79**. The method of claim **13**, further comprising the step of automatically interpreting which part of information in said document is said first information.

* * * * *

# Exhibit 11

US005946647A

# United States Patent [19]

**Miller et al.**

[11] **Patent Number: 5,946,647**

[45] **Date of Patent: Aug. 31, 1999**

[54] **SYSTEM AND METHOD FOR PERFORMING AN ACTION ON A STRUCTURE IN COMPUTER-GENERATED DATA**

[75] Inventors: **James R. Miller**, Mountain View; **Thomas Bonura**, Capitola; **Bonnie Nardi**, Mountain View; **David Wright**, Santa Clara, all of Calif.

[73] Assignee: **Apple Computer, Inc.**, Cupertino, Calif.

[21] Appl. No.: **08/595,257**

[22] Filed: **Feb. 1, 1996**

[51] **Int. Cl.**[6] ..... **G06F 17/27**

[52] **U.S. Cl.** ..... **704/9**; 704/1

[58] **Field of Search** ..... 704/1, 7, 9–10, 704/243; 707/513, 101–104

[56] **References Cited**

U.S. PATENT DOCUMENTS

| | | | |
|---|---|---|---|
| 5,115,390 | 5/1992 | Fukuda et al. | 364/146 |
| 5,130,924 | 7/1992 | Barker et al. | 704/1 |
| 5,164,899 | 11/1992 | Sobotka et al. | 704/9 |
| 5,202,828 | 4/1993 | Vertelney et al. | 364/419 |
| 5,247,437 | 9/1993 | Vale et al. | 704/1 |
| 5,369,575 | 11/1994 | Lamberti et al. | 704/1 |
| 5,574,843 | 11/1996 | Gerlach et al. | 395/118 |

OTHER PUBLICATIONS

TerryMorse Software "What is Myrmidon" Downloaded from the Internet at URL http://www.terrymorse.com (Publication Date Unknown), 2 pages.

Shoens, K. et al. "Rufus System: Information Organization for Semi–Structured Data," Proceedings of the 19th VLDB Conference (Dublin, Ireland 1993), pp. 1–12.

Schwarz, Peter and Shoens, Kurt. "Managing Change in the Rufus System," Abstract from the IBM Almaden Research Center, pp. 1–16.

Myers, Brad A. "Tourmaline: Text Formatting by Demonstration," (Chapter 14) in *Watch What I Do: Programming by Demonstration,* edited by Allen Cypher, MIT Press, (Cambridge, MA 1993), pp. 309–321.

Maulsby, David. "Instructible Agents," Dissertation from the Department of Computer Science at The University of Calgary (Calgary, Alberta—Jun. 1994), pp. 178, 181–188, 193–196 (from Chapter 5).

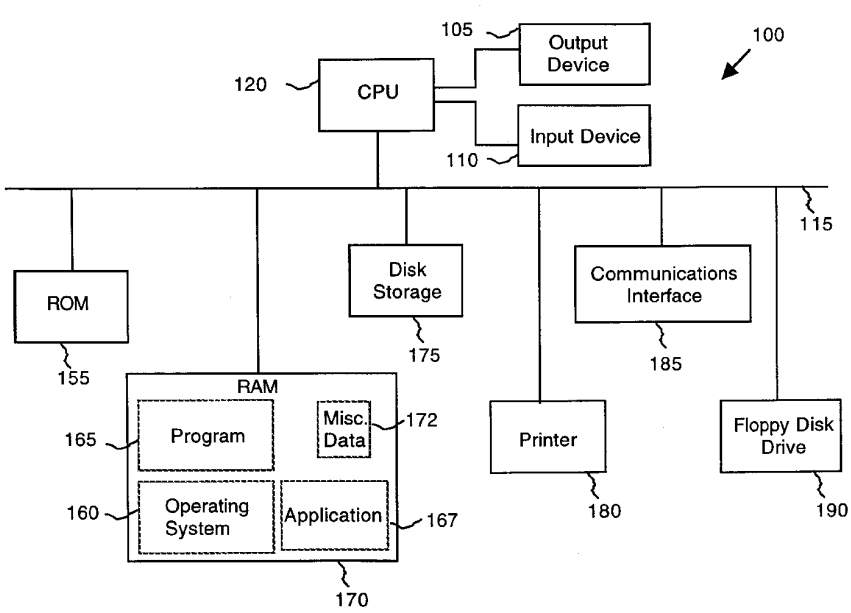
Rus, Daniela and Subramanian, Devika. "Designing Structure–Based Information Agents," AAAI Symposium (Mar. 1994), pp. 79–86.

*Primary Examiner*—Forester W. Isen
*Assistant Examiner*—Patrick N. Edouard
*Attorney, Agent, or Firm*—Carr & Ferrell LLP

[57] **ABSTRACT**

A system and method causes a computer to detect and perform actions on structures identified in computer data. The system provides an analyzer server, an application program interface, a user interface and an action processor. The analyzer server receives from an application running concurrently data having recognizable structures, uses a pattern analysis unit, such as a parser or fast string search function, to detect structures in the data, and links relevant actions to the detected structures. The application program interface communicates with the application running concurrently, and transmits relevant information to the user interface. Thus, the user interface can present and enable selection of the detected structures, and upon selection of a detected structure, present the linked candidate actions. Upon selection of an action, the action processor performs the action on the detected structure.

**24 Claims, 10 Drawing Sheets**



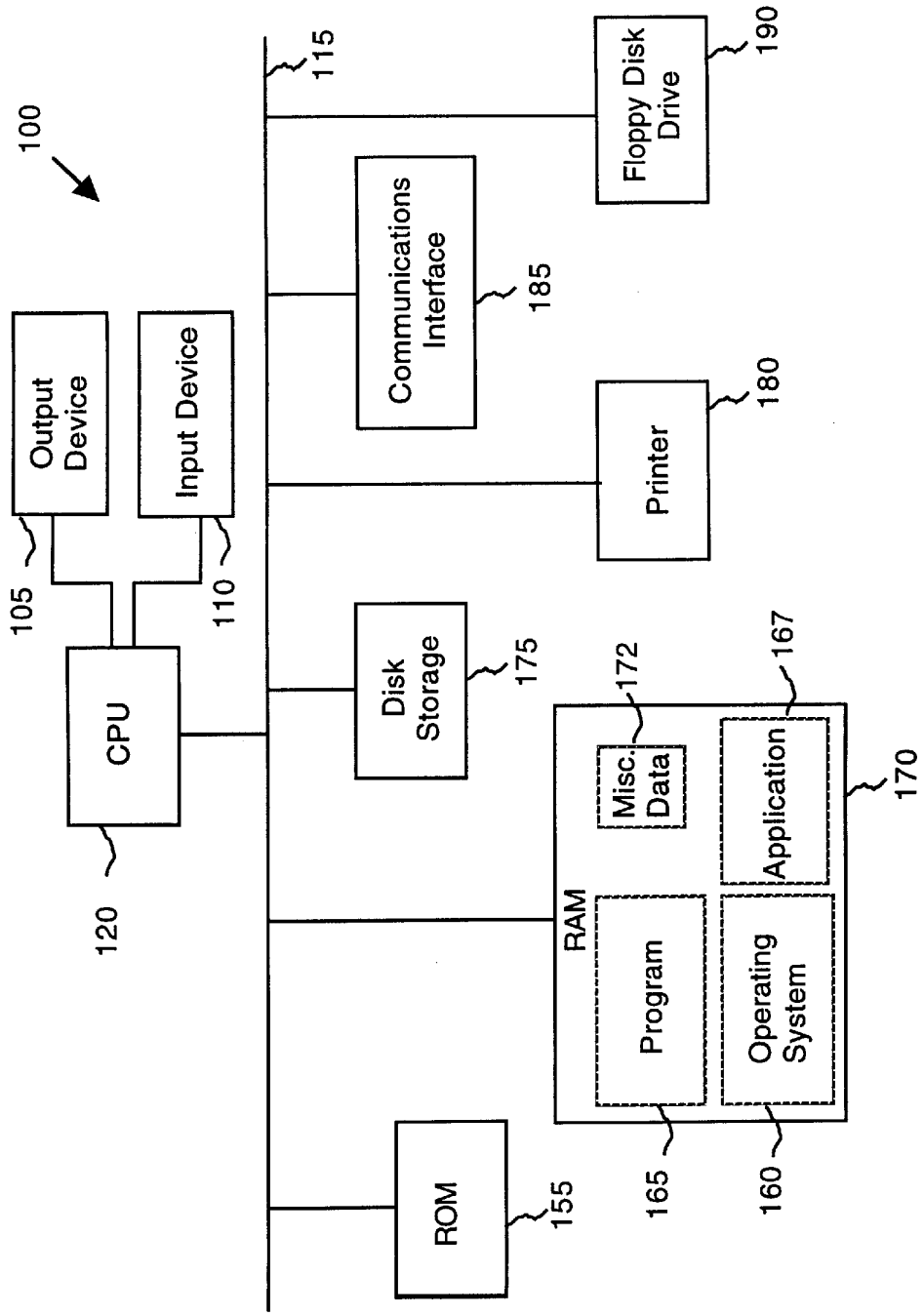

100
115
Floppy Disk Drive
190
Communications Interface
185
Output Device
Input Device
Printer
180
105
110
Disk Storage
175
172
167
CPU
Misc. Data
Application
170
120
RAM
Program
Operating System
ROM
155
165
160

FIG. 1

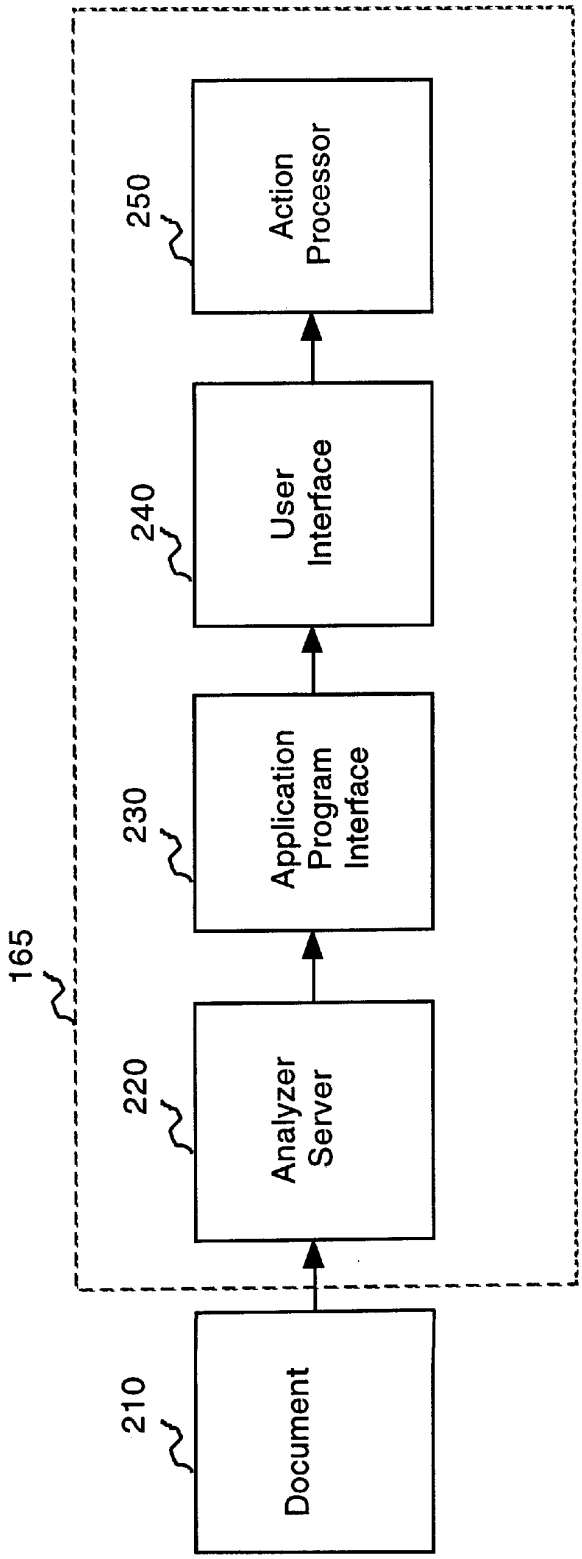

165
210
Document
220
Analyzer Server
230
Application Program Interface
240
User Interface
250
Action Processor

FIG. 2

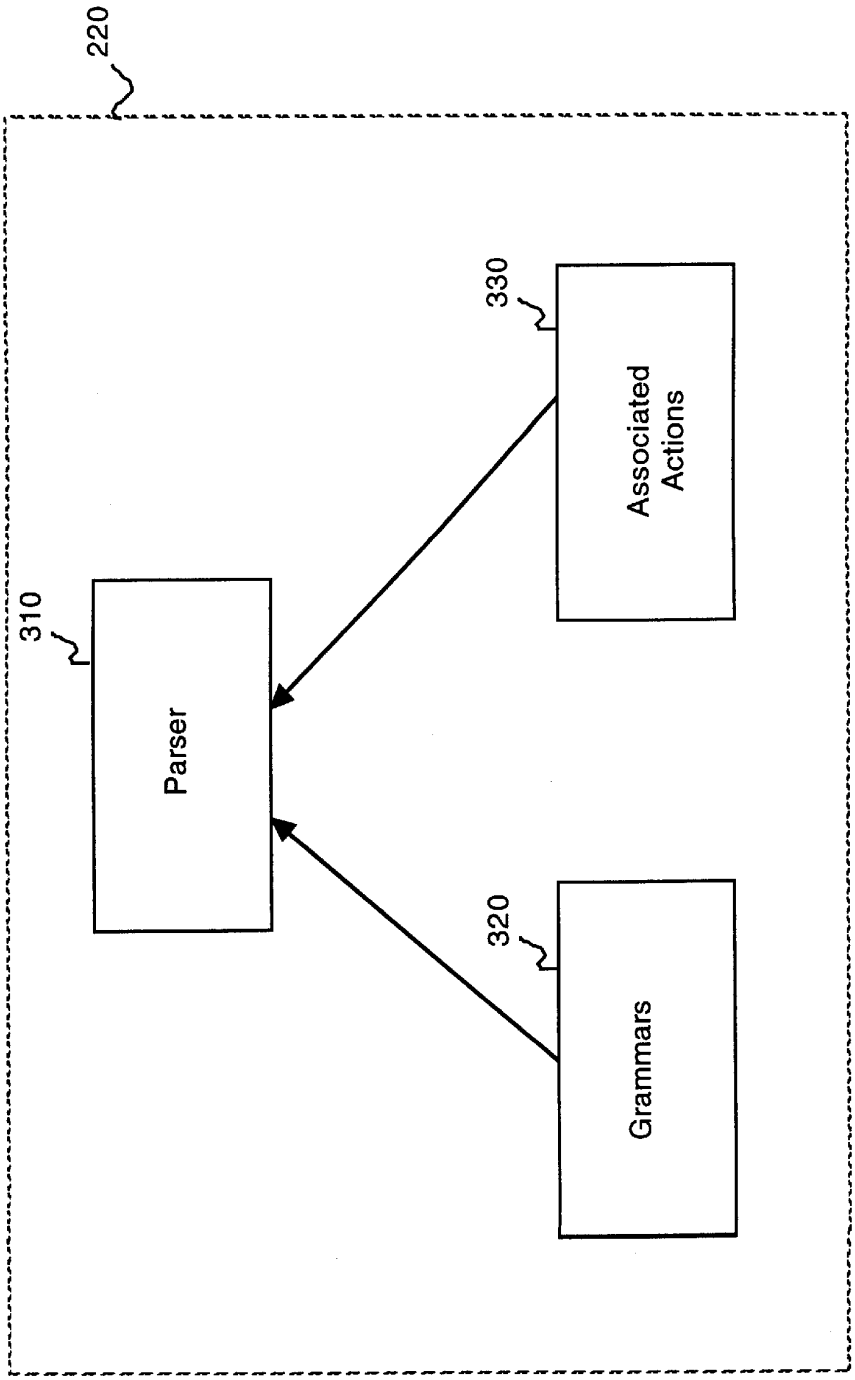



FIG. 3

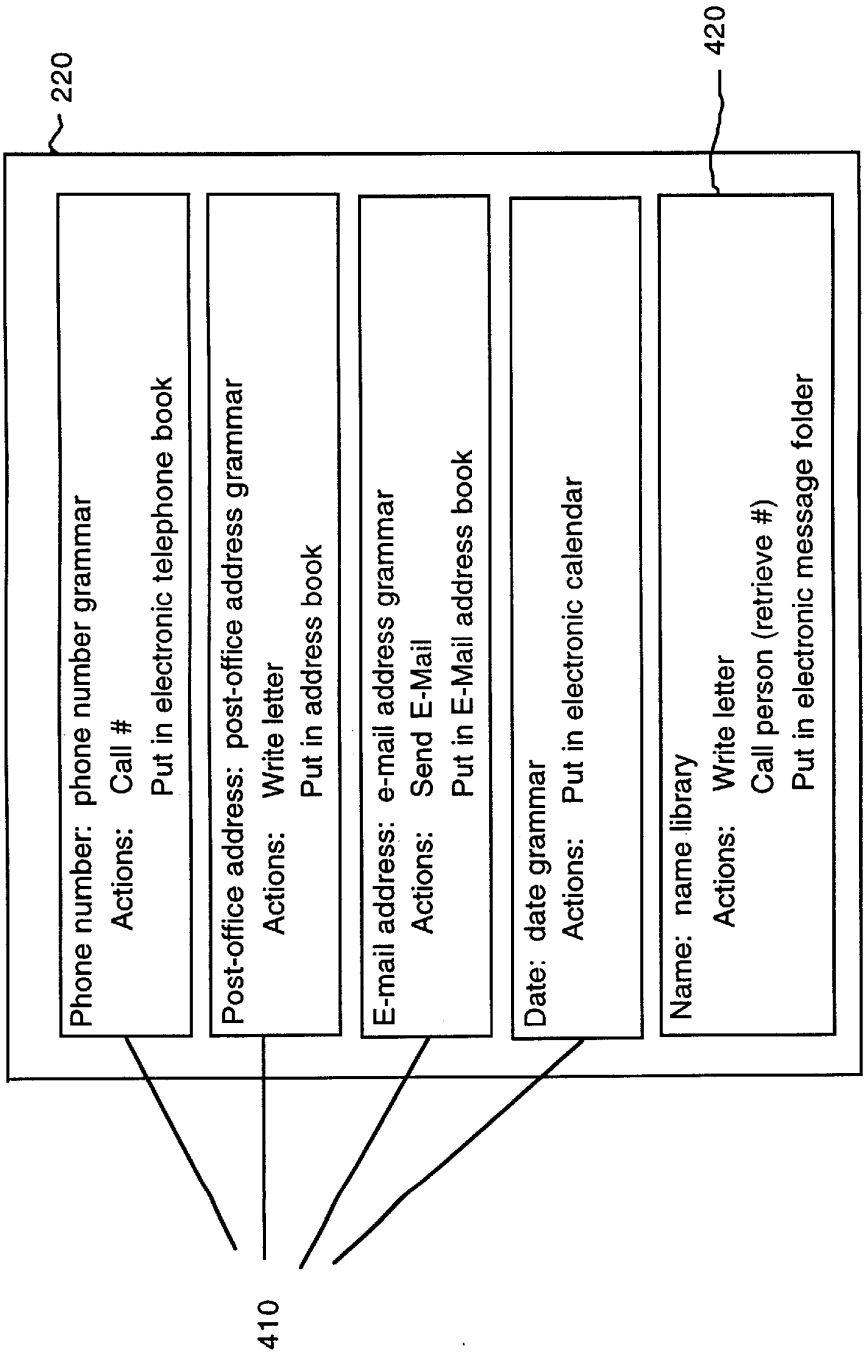

220
420
Phone number: phone number grammar
Actions: Call #
Put in electronic telephone book
Post-office address: post-office address grammar
Actions: Write letter
Put in address book
E-mail address: e-mail address grammar
Actions: Send E-Mail
Put in E-Mail address book
Date: date grammar
Actions: Put in electronic calendar
Name: name library
Actions: Write letter
Call person (retrieve #)
Put in electronic message folder
410

FIG. 4

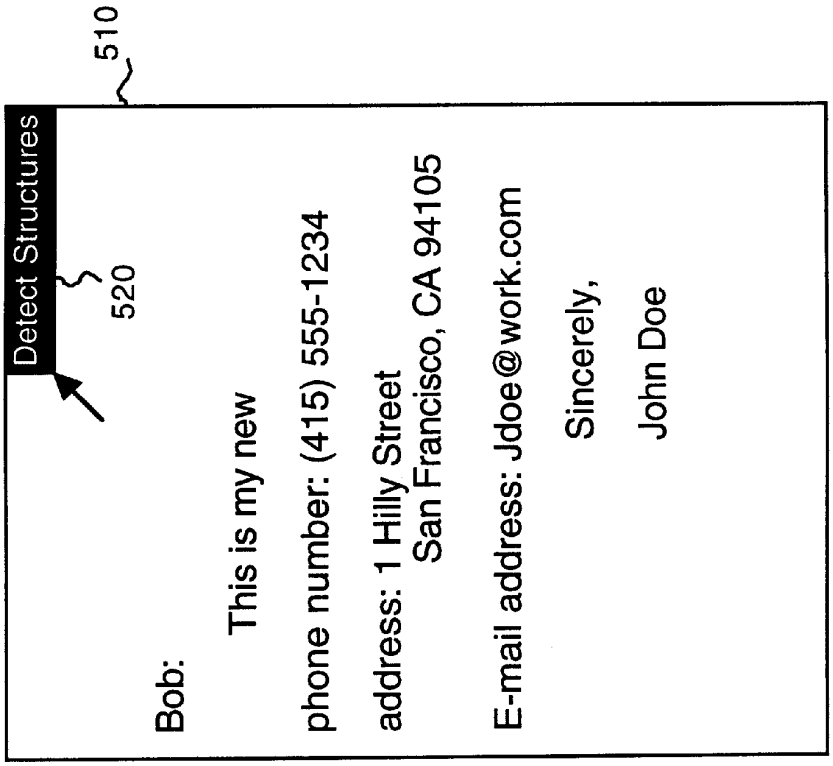

510
Detect Structures
520
Bob:
This is my new
phone number: (415) 555-1234
address: 1 Hilly Street
San Francisco, CA 94105
E-mail address: Jdoe@work.com
Sincerely,
John Doe

FIG. 5

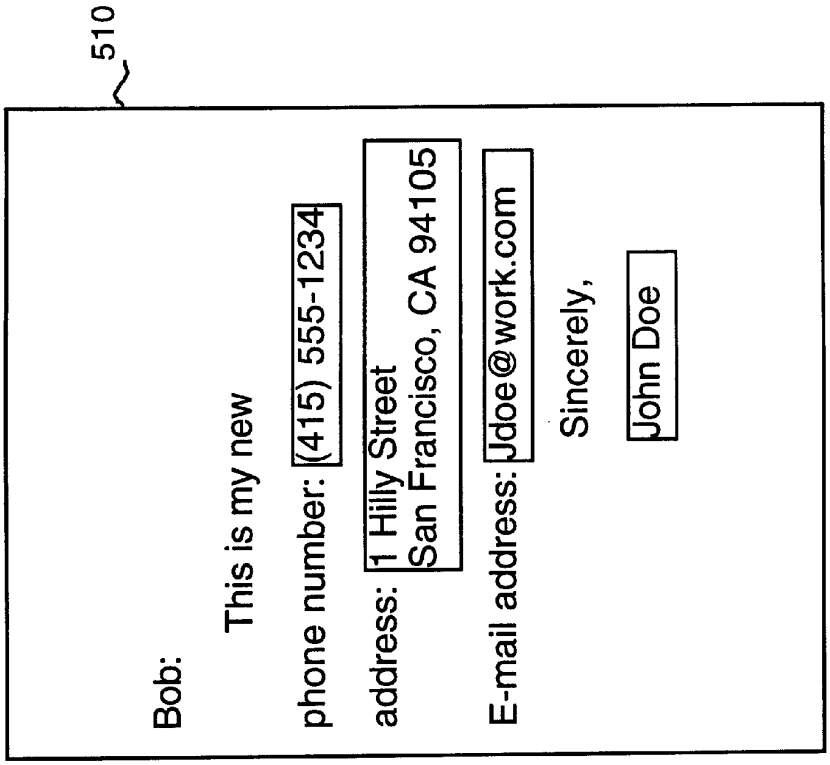

510
Bob:
This is my new
phone number: (415) 555-1234
address: 1 Hilly Street
San Francisco, CA 94105
E-mail address: Jdoe@work.com
Sincerely,
John Doe

FIG. 6

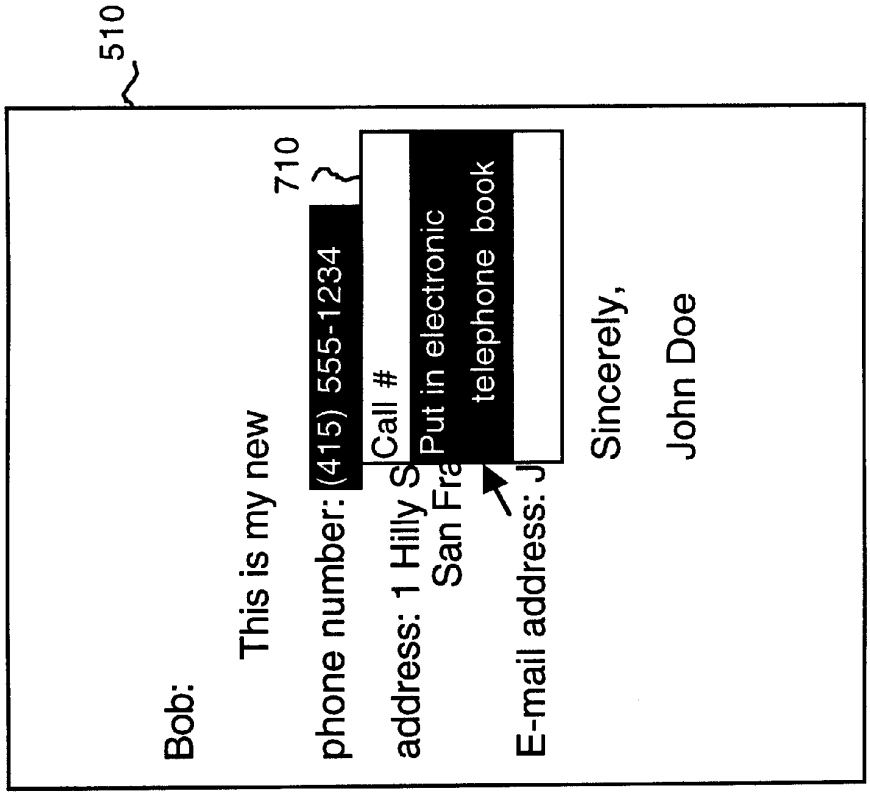

510
710
Bob:
This is my new
phone number: (415) 555-1234
address: 1 Hilly S
San Fra
E-mail address: J
Call #
Put in electronic telephone book
Sincerely,
John Doe

FIG. 7

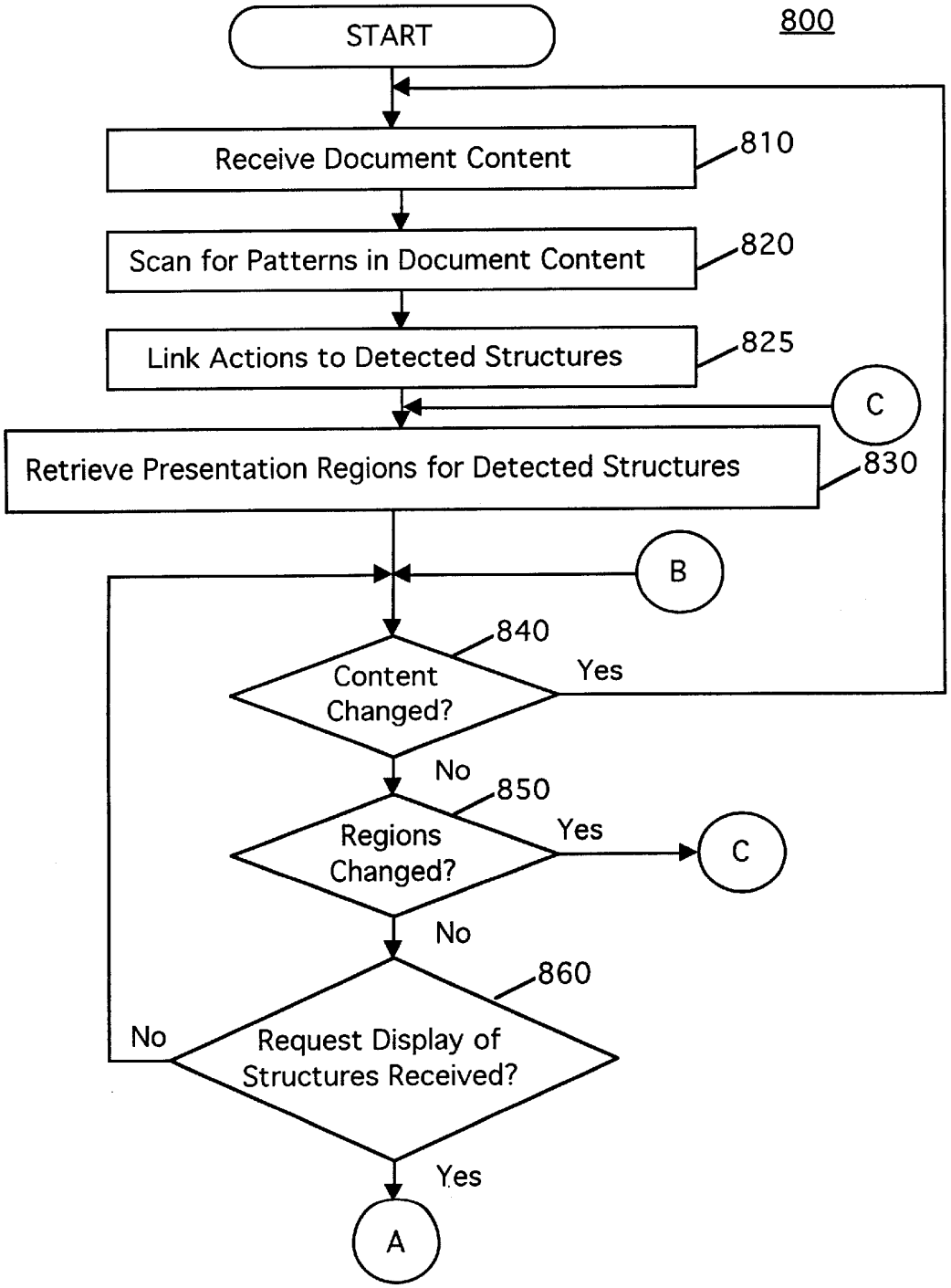

800
START
Receive Document Content
810
Scan for Patterns in Document Content
820
Link Actions to Detected Structures
825
C
Retrieve Presentation Regions for Detected Structures
830
B
840
Content Changed?
Yes
No
850
Regions Changed?
Yes
C
No
860
No
Request Display of Structures Received?
Yes
A

FIG. 8

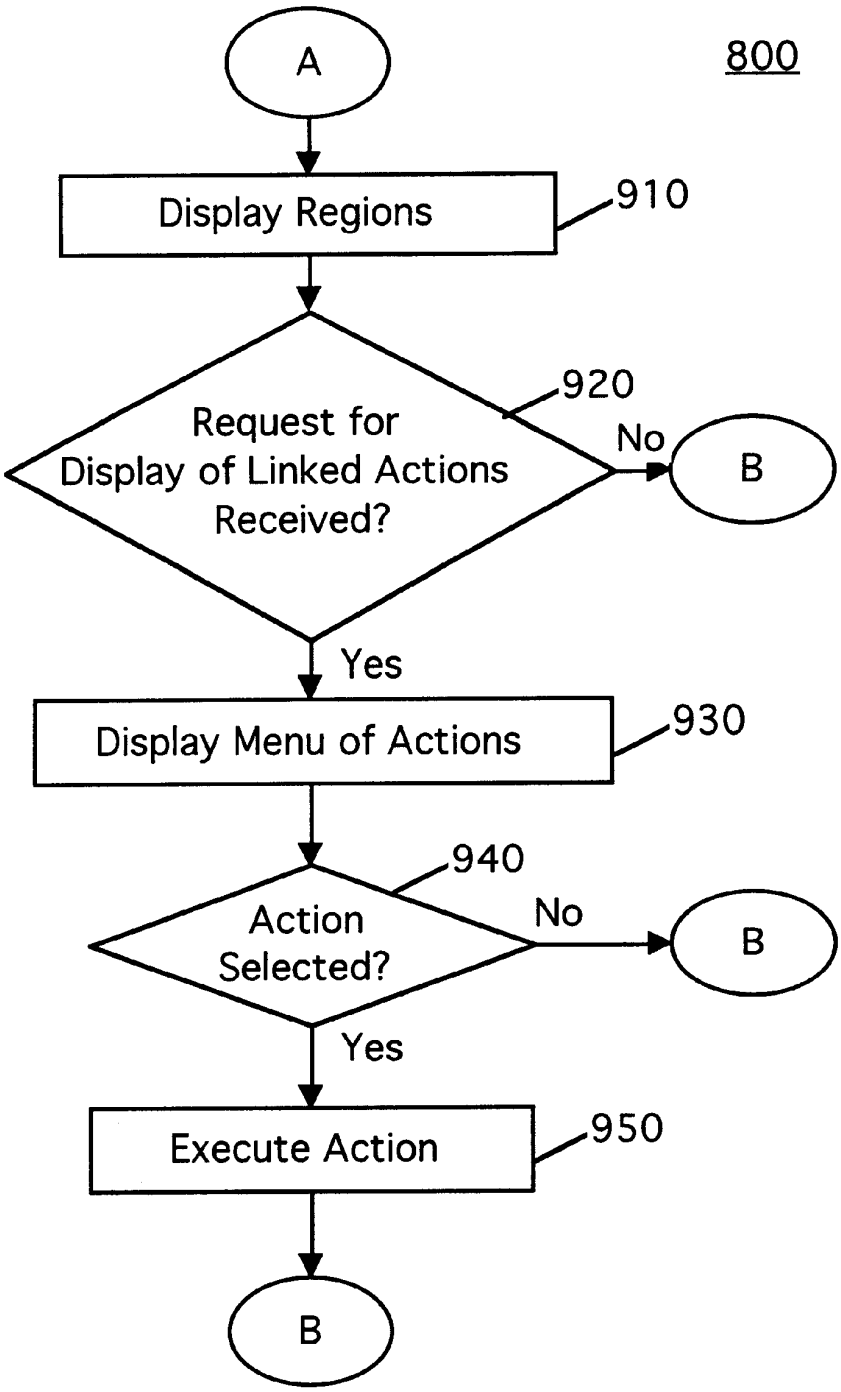

800
A
Display Regions
910
Request for Display of Linked Actions Received?
920
No
B
Yes
Display Menu of Actions
930
Action Selected?
940
No
B
Yes
Execute Action
950
B

FIG. 9

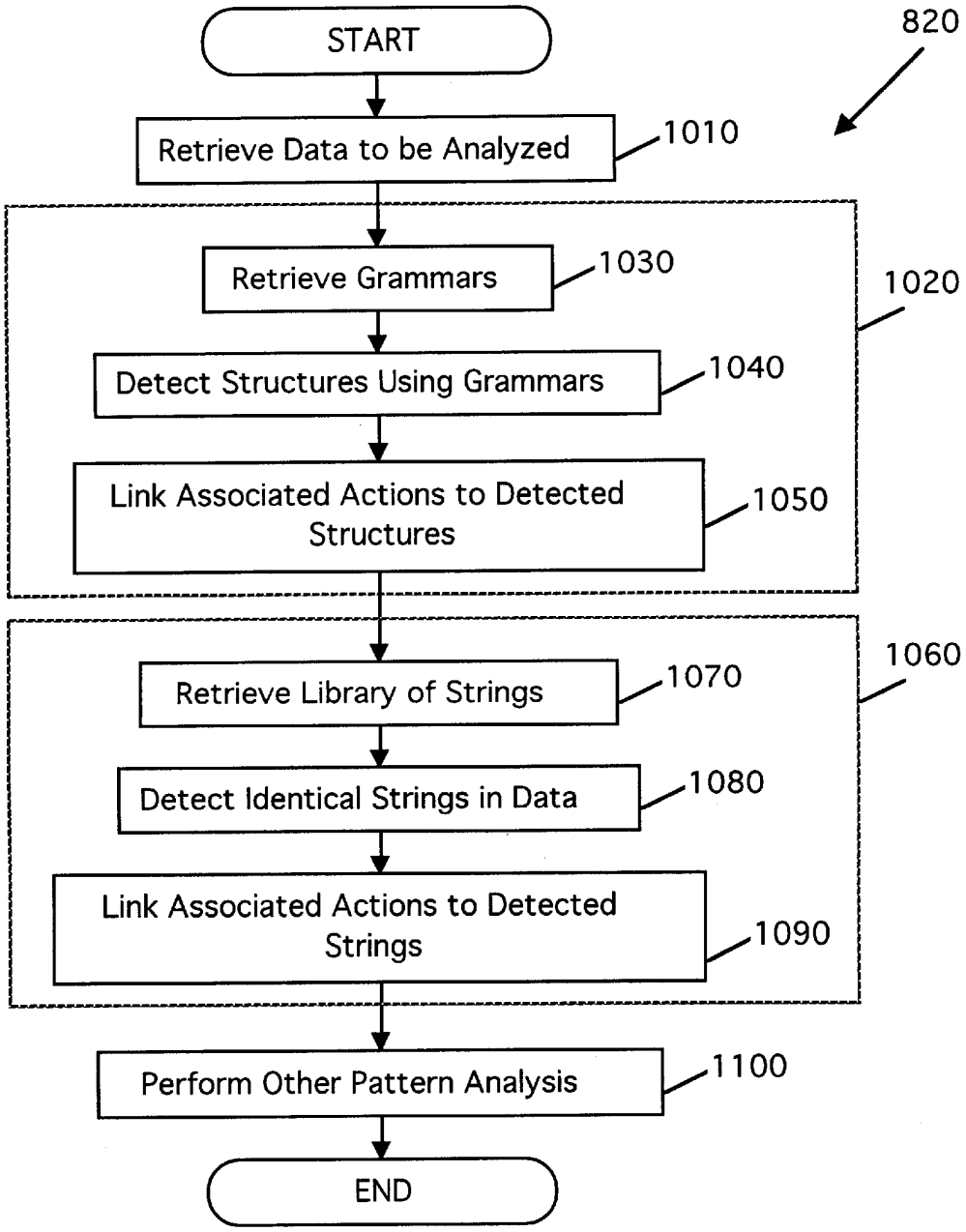

START
820
Retrieve Data to be Analyzed
1010
Retrieve Grammars
1030
1020
Detect Structures Using Grammars
1040
Link Associated Actions to Detected Structures
1050
1060
Retrieve Library of Strings
1070
Detect Identical Strings in Data
1080
Link Associated Actions to Detected Strings
1090
Perform Other Pattern Analysis
1100
END

FIG. 10

# SYSTEM AND METHOD FOR PERFORMING AN ACTION ON A STRUCTURE IN COMPUTER-GENERATED DATA

## BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates generally to manipulation of structures in computer data. More particularly, the invention relates to a system and method for performing computer-based actions on structures identified in computer data.

2. Description of the Background Art

Much data that appears in a computer user's day-to-day activities contains recognizable structures that have semantic significance such as phone numbers, e-mail addresses, post-office addresses, zip codes and dates. In a typical day, for example, a user may receive extensive files from word-processing programs and e-mail that contain several of these structures. However, visually searching data files or documents to find these structures is laborious and cognitively disruptive, especially if the document is lengthy and hard to follow. Furthermore, missing a structure such as a date may lead to missing an important meeting or missing a deadline.

To help facilitate searching a document for these structures, programmers can create or employ pattern analysis units, such as parsers, to automatically identify the structures. For the purposes of the present description, the term "pattern" refers to data, such as a grammar, regular expression, string, etc., used by a pattern analysis unit to recognize information in a document, such as dates, addresses, phone numbers, names, etc. The term "structure" refers to an instantiation of a pattern in the document. That is, a "date" pattern will recognize the structure "Oct. 31, 1995." The application of a pattern to a document is termed "parsing."

Conventional systems that identify structures in computer data do not enable automatic performance of an action on an identified structure. For example, if a long e-mail message is sent to a user, the user may implement a pattern analysis unit to search for particular structures, such as telephone numbers. Upon identification of a structure, the user may want to perform an action on the structure, such as moving the number to an electronic telephone book. This usually involves cutting the structure from the e-mail message, locating and opening the electronic telephone book application program, pasting the structure into the appropriate field, and closing the application program. However, despite the fact that computer systems are getting faster and more efficient, this procedure is still tedious and cognitively disruptive.

One type of system that has addressed this problem involves detecting telephone numbers. Such systems enable a user to select a telephone number and request that the application automatically dial the number. However, these systems do not recognize the selected data as a telephone number, and they generally produce an error message if the user selects invalid characters as a phone number. Also, they do not enable the performance of other candidate actions, such as moving the number to an electronic telephone book. That is, if a user wishes to perform a different action on an identified telephone number, such as storing the number in an address book, the user cannot automatically perform the action but must select and transfer the number to the appropriate data base as described above.

Therefore, a system is needed that identifies structures, associates candidate actions to the structures, enables selection of an action and automatically performs the selected action on the structure.



## SUMMARY OF THE INVENTION

The present invention overcomes the limitations and deficiencies of previous systems with a system that identifies structures in computer data, associates candidate actions with each detected structure, enables the selection of an action, and automatically performs the selected action on the identified structure. It will be appreciated that the system may operate on recognizable patterns for text, pictures, tables, graphs, voice, etc. So long as a pattern is recognizable, the system will operate on it. The present invention has significant advantages over previous systems, in that the present system may incorporate an open-ended number and type of recognizable patterns, an open-ended number and type of pattern analysis units, and further that the system may enable an open-ended number and type (i.e. scripts, macros, code fragments, etc.) of candidate actions to associate with, and thus perform, on each identified structure.

The present invention provides a computer system with a central processing unit (CPU), input/output (I/O) means, and a memory that includes a program to identify structures in a document and perform selected computer-based actions on the identified structures. The program includes program subroutines that include an analyzer server, an application program interface, a user interface and an action processor. The analyzer server receives data from a document having recognizable structures, and uses patterns to detect the structures. Upon detection of a structure, the analyzer server links actions to the detected structure. Each action is a computer subroutine that causes the CPU to perform a sequence of operations on the particular structure to which it is linked. An action may specify opening another application, loading the identified structure into an appropriate field, and closing the application. An action may further include internal actions, such as storing phone numbers in an electronic phone book, addresses in an electronic address book, appointments on an electronic calendar, and external actions such as returning phone calls, drafting letters, sending facsimile copies and e-mail, and the like.

Since the program may be executed during the run-time of another program, i.e. the application which presents the document, such as Microsoft Word, an application program interface provides mechanisms for interprogram communications. The application program interface retrieves and transmits relevant information from the other program to the user interface for identifying, presenting and enabling selection of detected structures. Upon selection of a detected structure, the user interface presents and enables selection of candidate actions. When a candidate action is selected, the action processor performs the selected action on the selected structure.

In addition to the computer system, the present invention also provides methods for performing actions on identified structures in a document. In this method, the document is analyzed using a pattern to identify corresponding structures. Identified structures are stored in memory and presented to the user for selection. Upon selection of an identified structure, a menu of candidate actions is presented, each of which may be selected and performed on the selected structure.

## BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of a computer system having a program stored in RAM, in accordance with the present invention.

FIG. **2** is a block diagram of the program of FIG. **1**.

FIG. **3** is a block diagram illustrating the analyzer server of FIG. **2**.

FIG. **4** is a block diagram illustrating a particular example of the analyzer server of FIG. **2**.

FIG. **5** illustrates a window presenting an example of a document having recognizable structures.

FIG. **6** illustrates a window with the identified structures in the example document of FIG. **5** highlighted based on the analyzer server of FIG. **4**.

FIG. **7** illustrates a window showing the display of a pop-up menu for selecting an action.

FIGS. **8** and **9** together are a flowchart depicting the preferred method for selecting and performing an action on an identified structure.

FIG. **10** is a flowchart depicting the preferred method for identifying a structure in a data sample.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Referring now to FIG. **1**, a block diagram is shown of a computer system **100** including a CPU **120**. Computer system **100** is preferably a microprocessor-based computer, such as a Power Macintosh manufactured by Apple Computer, Inc. of Cupertino, Calif. An input device **110**, such as a keyboard and mouse, and an output device **105**, such as a CRT or voice module, are coupled to CPU **120**. ROM **155**, RAM **170** and disk storage **175** are coupled to CPU **120** via signal bus **115**. Computer system **100** optionally further comprises a printer **180**, a communications interface **185**, and a floppy disk drive **190**, each coupled to CPU **120** via signal bus **115**.

Operating system **160** is a program that controls and facilitates the processing carried out by CPU **120**, and is typically stored in RAM **170**. Application **167** is a program, such as a word-processor or e-mail program, that presents data on output device **105** to a user. The program **165** of the present invention is stored in RAM **170** and causes CPU **120** to identify structures in the data presented by application **167**, to associate actions with the structures identified in the data, to enable the user to select a structure and an action, and to automatically perform the selected action on the identified structure. This program **165** may be stored in disk storage **175** and loaded into an allocated section of RAM **170** prior to execution by CPU **120**. Another section of RAM **170** is used for storing intermediate results and miscellaneous data **172**. Floppy disk drive **190** enables the storage of the present program **165** onto a removable storage medium which may be used to initially load program **165** into computer system **100**.

Referring now to FIG. **2**, a schematic block diagram of program **165** is shown together with its interaction with a document **210**. Program **165** contains program subroutines including an analyzer server **220**, an application program interface **230**, a user interface **240** and an action processor **250**. Analyzer server **220** receives data having recognizable patterns from a document **210**, which may be retrieved from a storage medium such as RAM **170**, ROM **155**, disk storage **175**, or the like, and presented on output device **105** by application **167**. Analyzer server **220** comprises one or more pattern analysis units, such as a parser and grammars or a fast string search function and dictionaries, which uses patterns to parse document **210** for recognizable structures. Upon detection of a structure, analyzer server **220** links actions associated with the responsible pattern to the detected structure, using conventional pointers.

After identifying structures and linking actions, application program interface **230** communicates with application **167** to obtain information on the identified structures so that user interface **240** can successfully present and enable selection of the actions. In a display-type environment, application program interface **230** retrieves the locations in document **210** of the presentation regions for the detected structures from application **167**. Application program interface **230** then transmits this location information to user interface **240**, which highlights the detected structures, although other presentation mechanisms can be used. User interface **240** enables selection of an identified structure by making the presentation regions mouse-sensitive, i.e. aware when a mouse event such as a mouse-down operation is performed while the cursor is over the region. Alternative selection mechanisms can be used such as touch sensitive screens and dialog boxes. It will be appreciated that detected structures can be hierarchical, i.e. that a sub-structure can itself be selected and have actions associated with it. For example, a user may be able to select the year portion of an identified date, and select actions specific to the year rather than to the entire date.

User interface **240** communicates with application **167** through application program interface **230** to determine if a user has performed a mouse-down operation in a particular mouse-sensitive presentation region, thereby selecting the structure presented at those coordinates. Upon selection of this structure, user interface **240** presents and enables selection of the linked candidate actions using any selection mechanism, such as a conventional pull-down or pop-up menu.

The above description of the user interface is cast in terms of a purely visual environment. However, the invention is not limited to visual interface means. For example, in an audio environment, user interface **240** may present the structures and associated actions to the user using voice synthesis and may enable selection of a pattern and action using voice or sound activation. In this type of embodiment, analyzer server **220** may be used in conjunction with a text-to-speech synthesis application **167** that reads documents to users over a telephone. Analyzer server **220** scans document **210** to recognize patterns and link actions to the recognized patterns in the same manner as described above. In the audio environment, user interface **240** may provide a special sound after application **167** reads a recognized pattern, and enable selection of the pattern through the use of an audio interface action, such as a voice command or the pressing of a button on the touch-tone telephone keypad as before. Thus, user interface **240** may present the linked actions via voice synthesis. One can create various environments having a combination of sensory mechanisms.

Upon selection of a candidate action, user interface **240** transmits the selected structure and the selected action to action processor **250**. Action processor **250** retrieves the sequence of operations that constitute the selected action, and performs the sequence using the selected structure as the object of the selected action.

Referring now to FIG. **3**, a block diagram illustrating an analyzer server **220** is shown. In this figure, analyzer server **220** is described as having a parser **310** and a grammar file **320**, although alternatively or additionally a fast string search function or other function can be used. Parser **310** retrieves a grammar from grammar file **320** and parses text using the retrieved grammar. Upon identification of a structure in the text, parser **310** links the actions associated with the grammar to the identified structure. More particularly, parser **310** retrieves from grammar file **320** pointers attached

to the grammar and attaches the same pointers to the identified structure. These pointers direct the system to the associated actions contained in associated actions file **330**. Thus, upon selection of the identified structure, user interface **240** can locate the linked actions.

FIG. **4** illustrates an example of an analyzer server **220**, which includes grammars **410** and a string library **420** such as a dictionary, each with associated actions. One of the grammars **410** is a telephone number grammar with associated actions for dialing a number identified by the telephone number grammar or placing the number in an electronic telephone book. Analyzer server **220** also includes grammars for post-office addresses, e-mail addresses and dates, and a string library **420** containing important names. When analyzer server **220** identifies an address using the "e-mail address" grammar, actions for sending e-mail to the identified address and putting the identified address in an e-mail address book are linked to the address.

FIG. **5** shows a window **510** presenting an exemplary document **210** having data containing recognizable structures, including a phone number, post-office address, e-mail address, and name. Window **510** includes a button **520** for initiating program **165**, although alternative mechanisms such as depressing the "option" key may be used. Upon initiation of program **165**, system **100** transmits the contents of document **210** to analyzer server **220**, which parses the contents based on grammars **410** and strings **420** (FIG. **4**). This parsing process produces the window shown in FIG. **6**. As illustrated in FIG. **6**, analyzer server **220** identifies the phone number, post-office address, e-mail address and name. Although not shown in FIG. **6**, analyzer server **220** links the actions associated with grammars **410** and strings **420** to these identified structures, and application program interface **230** retrieves information on the location of these structures from application **167**. User interface **240** then highlights the identified structures in document **210**, and makes the identified structures mouse-sensitive.

As shown in FIG. **7**, upon recognition of a mouse-down operation over a structure, user interface **240** presents a pop-up menu **710**. In this example, pop-up menu **710** displays the candidate actions linked to the selected telephone number grammar **410**, including dialing the number and putting the number into an electronic telephone book. Upon selection of the action for putting the number in an electronic telephone book, user interface **240** transmits the corresponding telephone number and selected action to action processor **250**. Action processor **250** locates and opens the electronic telephone book, places the telephone number in the appropriate field and allows the user to input any additional information into the file.

FIGS. **8** and **9** display a flowchart illustrating preferred method **800** for recognizing patterns in documents and performing actions. This method is carried out during the run-time of application **167**. Referring first to FIG. **8**, method **800** starts by receiving **810** the content, or a portion of the content, from document **210**. Assuming program **165** initiates with the receipt of any text, the received content or portion is scanned **820** for identifiable structures using the patterns in analyzer server **220**. Upon detection of a structure based on a particular pattern, actions associated with the particular pattern are linked **825** to the detected structure. Assuming a display-type environment, the presentation region location for a detected structure is retrieved **830** from application **167**. If the document content being displayed on output device **105** is changed **840**, for example by the user adding or modifying text, method **800** restarts. Otherwise, method **800** continues with block **850**. If the presentation regions change **850**, for example by the a user scrolling document **210**, then new presentation regions from application **167** are again retrieved **830**. Otherwise, method **800** continues to block **860**. As illustrated by block **860**, method **800** loops between blocks **840** and **860** until a request for display of identified structures is received **860**. It will be appreciated that the steps of the loop (blocks **840**, **850** and **860**) can be performed by application **167**.

Referring also to FIG. **9**, when a request for the display of detected structures is received **860**, the regions are displayed **910** using presentation mechanisms such as highlighting the presentation region around each detected structure, although alternative presentation mechanisms can be used. If a request for the display of candidate actions linked to a detected structure is not received **920**, method **800** returns to block **840**. However, if a request is received **920**, the actions linked in block **825** are displayed **930**. This request for display of candidate actions can be performed using a selection mechanism, such as a mouse-down operation over a detected structure, which causes the candidate actions linked to the structure to be displayed **930**. Display **930** of candidate actions may be implemented using a pop-up menu, although alternative presentation mechanisms can be used such as pull-down menus, dialog boxes and voice synthesizers.

As illustrated in block **940**, if an action from the displayed candidate actions is not selected **940**, method **800** returns to block **840**. However, if an action is selected **940**, the action is executed **950** on the structure selected in block **920**. After execution **950** of an action, method **800** returns to block **840**. Method **800** ends when the user exits application **167**, although other steps for ending method **800** can alternatively be used.

Referring now to FIG. **10**, a flowchart illustrating the preferred method **820** for scanning and detecting patterns in a document is shown. Method **820** starts by retrieving **1010** data to be analyzed. After the data is retrieved, several pattern analysis processes may be performed on the data. As illustrated in block **1020**, a parsing process retrieves **1030** grammars, detects **1040** structures in the data based on the retrieved grammars, and links **1050** actions associated with each grammar to each structure detected by that grammar. As illustrated in block **1060**, a fast string search function retrieves **1070** the contents of string library **420**, detects **1080** the strings in the data identical to those in the string library **420**, and links **1090** actions associated with the library string to the detected string. As illustrated in block **1100**, additional pattern analysis processes, such as a neural net scan, can be performed **1100** to detect in the data other patterns, such as pictures, graphs, sound, etc. Method **820** then ends. Alternatively, the pattern analysis processes can be performed in parallel using a multiprocessor multitasking system, or using a uniprocessor multithreaded multitasking system where a thread is allocated to execute each pattern detection scheme.

These and other variations of the preferred and alternate embodiments and methods are provided by the present invention. For example, program **165** in FIG. **1** can be stored in ROM, disk, or in dedicated hardware. In fact, it may be realized as a separate electronic circuit. Other components of this invention may be implemented using a programmed general purpose digital computer, using application specific integrated circuits, or using a network of interconnected conventional components and circuits. The analyzer server **220** of FIG. **2** may use a neural net for searching a graphical document **210** for faces, or a musical library for searching a stored musical piece **210** for sounds. The user interface **240**

may present structures and actions via voice synthesis over a telephone line connection to system **100**. The embodiments described have been presented for purposes of illustration and are not intended to be exhaustive or limiting, and many variations and modifications are possible in light of the foregoing teaching. The system is limited only by the following claims.

What is claimed is:

**1**. A computer-based system for detecting structures in data and performing actions on detected structures, comprising:

an input device for receiving data;

an output device for presenting the data;

a memory storing information including program routines including

an analyzer server for detecting structures in the data, and for linking actions to the detected structures;

a user interface enabling the selection of a detected structure and a linked action; and

an action processor for performing the selected action linked to the selected structure; and

a processing unit coupled to the input device, the output device, and the memory for controlling the execution of the program routines.

**2**. The system recited in claim **1**, wherein the analyzer server stores detected structures in the memory.

**3**. The system recited in claim **1**, wherein the input device receives the data from an application running concurrently, and wherein the program routines stored in memory further comprise an application program interface for communicating with the application.

**4**. The system recited in claim **1**, wherein the analyzer server includes grammars and a parser for detecting structures in the data.

**5**. The system recited in claim **4**, wherein the analyzer server includes actions associated with each of the grammars, and wherein the analyzer server links to a detected structure the actions associated with the grammar which detects that structure.

**6**. The system recited in claim **1**, wherein the analyzer server includes a string library and a fast string search function for detecting string structures in the data.

**7**. The system recited in claim **6**, wherein the analyzer server includes actions associated with each of the strings, and wherein the analyzer server links to a detected structure the actions associated with the grammar which detects that string structure.

**8**. The system recited in claim **1**, wherein the user interface highlights detected structures.

**9**. The system recited in claim **1**, wherein the user interface enables selection of an action by causing the output device to display a pop-up menu of the linked actions.

**10**. The system recited in claim **1**, wherein the programs stored in the memory further comprise an application running concurrently that causes the output device to present the data received by the input device, and an application program interface that provides interrupts and communicates with the application.

**11**. The system recited in claim **1**, wherein the user interface enables the selection of a detected structure and a linked action using sound activation.

**12**. The system recited in claim **1**, wherein a first one of the actions may invoke a second one of the actions.

**13**. A program storage medium storing a computer program for causing a computer to perform the steps of:

receiving computer data;

detecting a structure in the data;

linking at least one action to the detected structure;

enabling selection of the structure and a linked action; and

executing the selected action linked to the selected structure.

**14**. In a computer having a memory storing actions, a system for causing the computer to perform an action on a structure identified in computer data, comprising:

means for receiving computer data;

means for detecting a structure in the data;

means for linking at least one action to the detected structure;

means for selecting the structure and a linked action; and

means for executing the selected action linked to the selected structure.

**15**. In a computer having a memory storing actions, a method for causing the computer to perform an action on a structure identified in computer data, comprising the steps of:

receiving computer data;

detecting a structure in the data;

linking at least one action to the detected structure;

enabling selection of the structure and a linked action; and

executing the selected action linked to the selected structure.

**16**. The method recited in claim **15**, wherein the computer data is received from the application running concurrently.

**17**. The method recited in claim **15**, wherein the memory contains grammars, and wherein the step of detecting a structure further comprises the steps of retrieving a grammar and parsing the data based on the grammar.

**18**. The method recited in claim **17**, wherein the grammar is associated with a particular action, and wherein the step of linking at least one action to the detected structure includes the step of linking the particular action to the detected structure.

**19**. The method recited in claim **15**, wherein the memory contains strings, and wherein the step of detecting a structure further comprises the steps of retrieving a string from the memory and scanning the data to identify the string.

**20**. The method recited in claim **15**, further comprising after the step of detecting a structure, the step of highlighting the detected structure.

**21**. The method recited in claim **15**, further comprising, after the step of linking at least one action to the detected structure, the step of displaying and enabling selection of an action for performance on the detected structure.

**22**. A computer-based method for causing a computer to identify, select and perform an action on a structure in computer data received from a concurrently running application, said application presenting the computer data to the user, the method comprising the steps of:

receiving computer data from the application;

detecting a structure in the computer data;

linking at least one action to the detected structure;

communicating with the application to determine the location of the detected structure as presented by the application, to enable selection of the detected structure and a linked action, and to determine if the detected structure and a linked action have been selected; and

performing a selected action linked to the detected pattern.

**23**. The method recited in claim **15**, wherein the step of enabling uses sound activation.

**24**. The method recited in claim **15**, wherein a first one of the actions may invoke a second one of the actions.

* * * * *

# Exhibit 12

US005644735A

# United States Patent [19]

Luciw et al.

[11] Patent Number: 5,644,735

[45] Date of Patent: Jul. 1, 1997

[54] **METHOD AND APPARATUS FOR PROVIDING IMPLICIT COMPUTER-IMPLEMENTED ASSISTANCE**

[75] Inventors: **William W. Luciw**, Morgan Hill; **Stephen P. Capps**, San Carlos; **Lawrence G. Tesler**, Portola Valley, all of Calif.

[73] Assignee: **Apple Computer, Inc.**, Cupertino, Calif.

[21] Appl. No.: **424,959**

[22] Filed: **Apr. 19, 1995**

**Related U.S. Application Data**

[60] Division of Ser. No. 99,861, Jul. 30, 1993, Pat. No. 5,477,447, which is a continuation-in-part of Ser. No. 889,225, May 27, 1992, Pat. No. 5,390,281.

[51] **Int. Cl.**[6] ..... **G06F 17/30**

[52] **U.S. Cl.** ..... **395/338**; 395/336

[58] **Field of Search** ..... 395/155, 156, 395/157, 159, 161, 149, 336, 337, 338, 968, 759, 12, 51, 62; 364/419.08, 419.19, 419.1, 419.14, 419.15, 419.13

[56] **References Cited**

U.S. PATENT DOCUMENTS

| | | | |
|---|---|---|---|
| Re. 34,476 | 12/1993 | Norwood | 382/186 |
| 4,670,848 | 6/1987 | Schramm et al. | 395/60 |
| 4,713,775 | 12/1987 | Scott et al. | 395/50 |
| 4,736,296 | 4/1988 | Katayama et al. | 395/759 |
| 4,862,390 | 8/1989 | Weiner | 395/336 X |
| 4,875,187 | 10/1989 | Smith | 395/141 |
| 4,918,723 | 4/1990 | Iggulden et al. | 379/100 |
| 4,945,504 | 7/1990 | Nakama et al. | 364/709.11 |
| 4,953,106 | 8/1990 | Gansner et al. | 395/140 |
| 4,974,191 | 11/1990 | Amirghodsi et al. | 395/759 |
| 5,091,790 | 2/1992 | Silverberg | 358/434 |
| 5,103,498 | 4/1992 | Lanier | 395/68 |
| 5,109,509 | 4/1992 | Katayama et al. | 395/759 |
| 5,239,617 | 8/1993 | Gardner et al. | 395/12 |

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

| | | |
|---|---|---|
| 0441089A2 | 8/1991 | European Pat. Off. . |
| 1-130291 | 5/1989 | Japan . |

OTHER PUBLICATIONS

Wilensky, Robert; Arens, Yigal; and Chin, David, "Talking to UNIX in English: An Overview of UC," Communications of the ACM, Jun. 1984, vol. 27, No. 6, pp. 574 to 593.

Tello, Ernest R., "Natural Language Systems," Savvy PC, Clout 2, Q&A, Lotus HAL, Mastering A1 Tools and Techniques, Chapter 2, pp. 25 to 64.

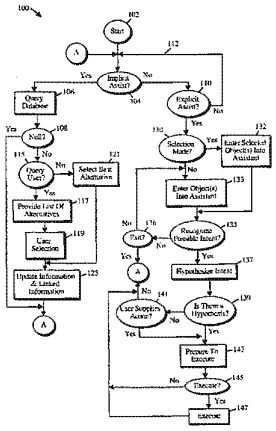
(List continued on next page.)

*Primary Examiner*—Kee M. Tung
*Assistant Examiner*—Crescelle N. dela Torre
*Attorney, Agent, or Firm*—Hickman Beyer & Weaver

[57] **ABSTRACT**

A method and apparatus for providing computer-assisted implicit and explicit assistance. If no implicit assist actions are desired or indicated, a logical process is initiated to determine whether explicit assistance should be undertaken. If implicit assistance is indicated, a list of action alternatives is displayed for the user. If explicit assistance is desired by the user, particular object(s) from which the assistance may be inferred are entered into an assistance operation. An attempt is made to recognize possible intents expressed by the objects entered into the assistance process. If no user intent is, in fact, recognized, the assist operation is usually terminated. If a possible intent is recognized, the actual intent is hypothesized. A check is further undertaken, to determine whether a hypothesis is in fact available. If no hypothesis is found, the process permits the user to supply a proposed action. If no hypothesis is found and no user action is proposed, assistance efforts terminate. However, if a hypothesis is available, preparations for execution are undertaken. A final inquiry is made as to whether to undertake the hypothesized assist. If the response to an inquiry whether to assist as hypothesized is affirmative, execution of the hypothesized action is undertaken. A pen-based computer preferably implements the indicated functions.

**20 Claims, 18 Drawing Sheets**



U.S. PATENT DOCUMENTS

| | | | |
|---|---|---|---|
| 5,255,386 | 10/1993 | Prager | 395/605 |
| 5,390,281 | 2/1995 | Luciw et al. | 395/12 |
| 5,432,902 | 7/1995 | Matsumoto | 395/338 |
| 5,477,447 | 12/1995 | Luciw et al. | 395/759 |
| 5,535,323 | 7/1996 | Miller et al. | 395/338 |

OTHER PUBLICATIONS

O'Connor, Rory J., "Apple Banking on Newton's Brain", *San Jose Mercury News,* Wednesday, Apr. 22, 1992.

Hendrix, Gary G. and Walter, Brett A., "The Intelligent Assistant: Technical Considerations Involved in Designing Q&A's Natural–language Interface", *Byte Magazine,* Dec. 1987, vol. 12, Issue 14, p. 251.

Edwards, John R., "Q&A: Integrated Software with Macros and an Intelligent Assistant", *Byte Magazine,* Jan. 1986, vol. 11, Issue 1, pp. 120–122.

Goldberg, Cheryl, "IBM Drawing Assistant: Graphics for the EGA", *PC Magazine,* Dec. 24, 1985, vol. 4, Issue 26, p. 255.

Garretson, R., "IBM Adds 'Drawing Assistant' Design Tool to Graphics Series", *PC Week,* Aug. 13, 1985, vol. 2, Issue 32, p. 8.

Glinert–Stevens, Susan, "Microsoft Publisher: Desktop Wizardry", *PC Sources,* Feb., 1992, vol. 3, Issue 2, p. 357.

Nilsson, B.A., "Microsoft Publisher is an Honorable Start for DTP Beginners", *Computer Shopper,* Feb. 1992, vol. 12, Issue 2, p. 426.

Poor, Alfred, "Microsoft Publisher", *PC Magazine,* Nov. 26, 1991, vol. 10, Issue 20, p. 40, evaluates Microsoft Publisher.

Rampe, Dan, et al. In a Jan. 9, 1989 news release, Claris Corporation announced two products, SmartForm Designer and SmartForm Assistant, which provide "Intelligent Assistance", such as custom help messages, choice lists, and data–entry validation and formatting.

Berry, Deanne, et al. In an Apr. 10, 1990 news release, Symantec announced a new version of MORE (TM).

Elofson, G. and Konsynski, B., "Delegation Technologies: Environmental Scanning with Intelligent Agents", *Journal of Management Information Systems,* Summer 1991, vol. 8, Issue 1, pp. 37–62.

Nadoli, Gajanana and Biegel, John, "Intelligent Agents in the Simulation of Manufacturing Systems", *Proceedings of the SCS Multiconference on AI and Simulation,* 1989.

Sharif Heger, A. and Koen, B. V., "KNOWBOT: an Adaptive Data Base Interface", *Nuclear Science and Engineering,* Feb. 1991, vol. 107, No. 2, pp. 142–157, describes an adaptive interface KNOWBOT.

Ohsawa, I. and Yonezawa, A., "A Computational Model of an Intelligent Agent Who Talks with a Person", *Research Reports on Information Sciences, Series C,* Apr. 1989, No. 92, pp. 1–18.

Ratcliffe, Mitch and Gore, Andrew, "Intelligent Agents take U.S. Bows.", *MacWeek,* Mar. 2, 1992, vol. 6, No. 9, p. 1.

Boy, Guy A., *Intelligent Assistant Systems,* Harcourt Brace Jovanovich, 1991, uses the term "Intelligent Assistant Systems".

*Microsoft Windows User's Guide for the Windows Graphical Environment*; Version 3.0; Microsoft Press copyright 1990–1995.

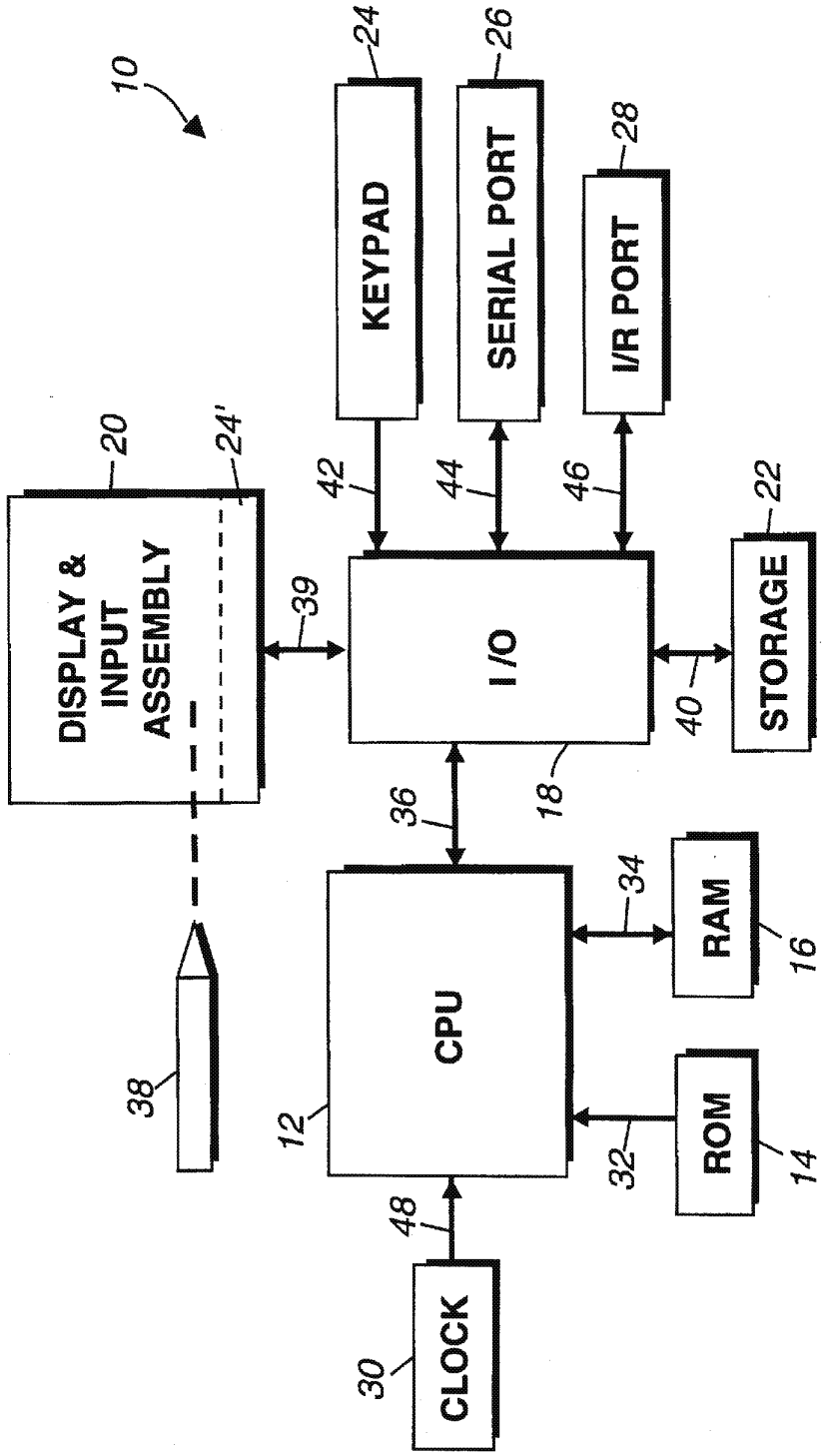

10
KEYPAD
24
SERIAL PORT
26
I/R PORT
28
20
24'
42
44
46
22
DISPLAY & INPUT ASSEMBLY
39
I/O
STORAGE
40
36
18
34
RAM
16
38
CPU
12
ROM
32
14
48
CLOCK
30

Figure 1

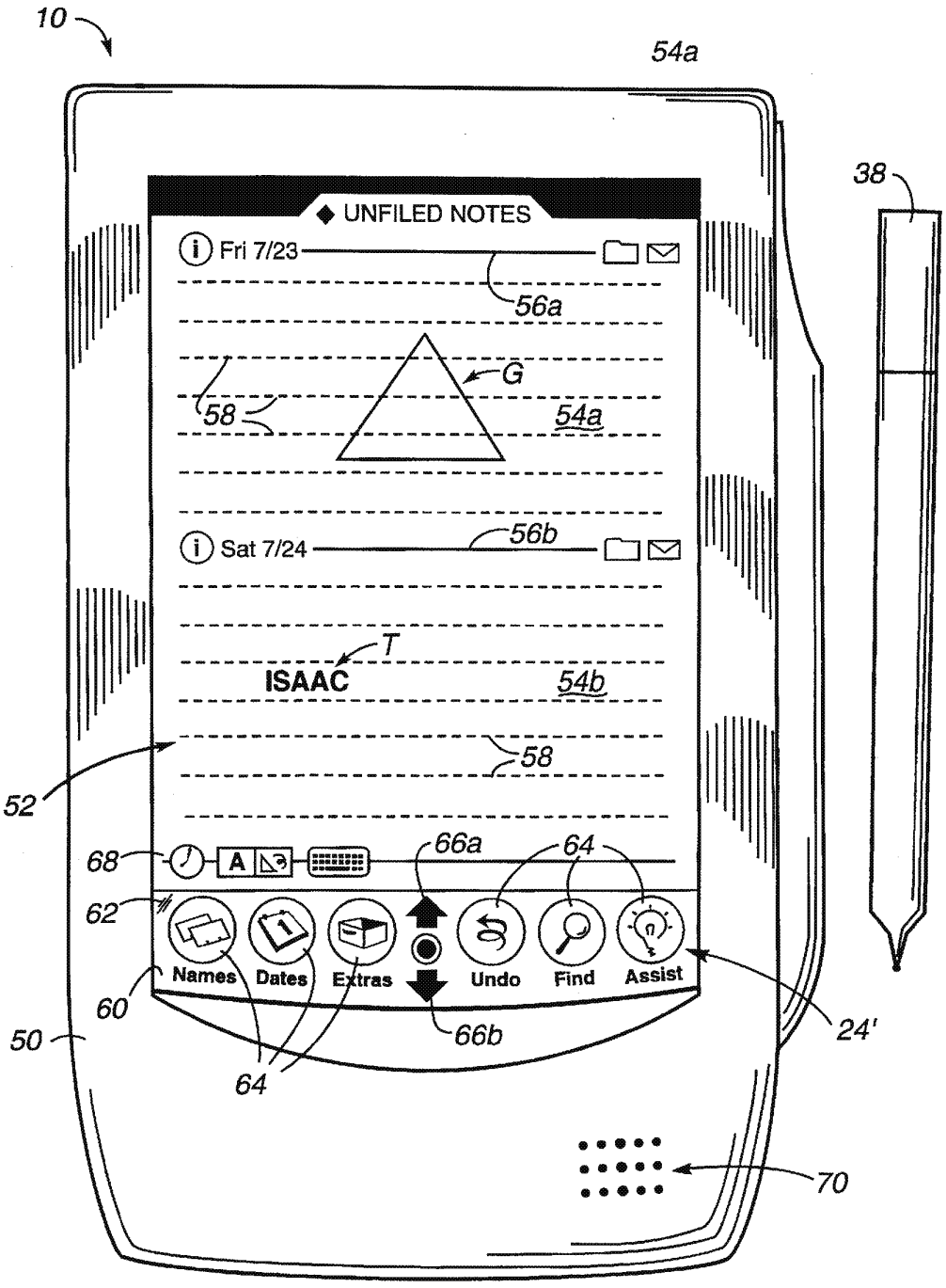

10
54a
38
UNFILED NOTES
Fri 7/23
56a
G
58
54a
Sat 7/24
56b
T
ISAAC
54b
58
52
68
66a
64
62
Names
Dates
Extras
Undo
Find
Assist
60
66b
24'
50
64
70

Figure 2

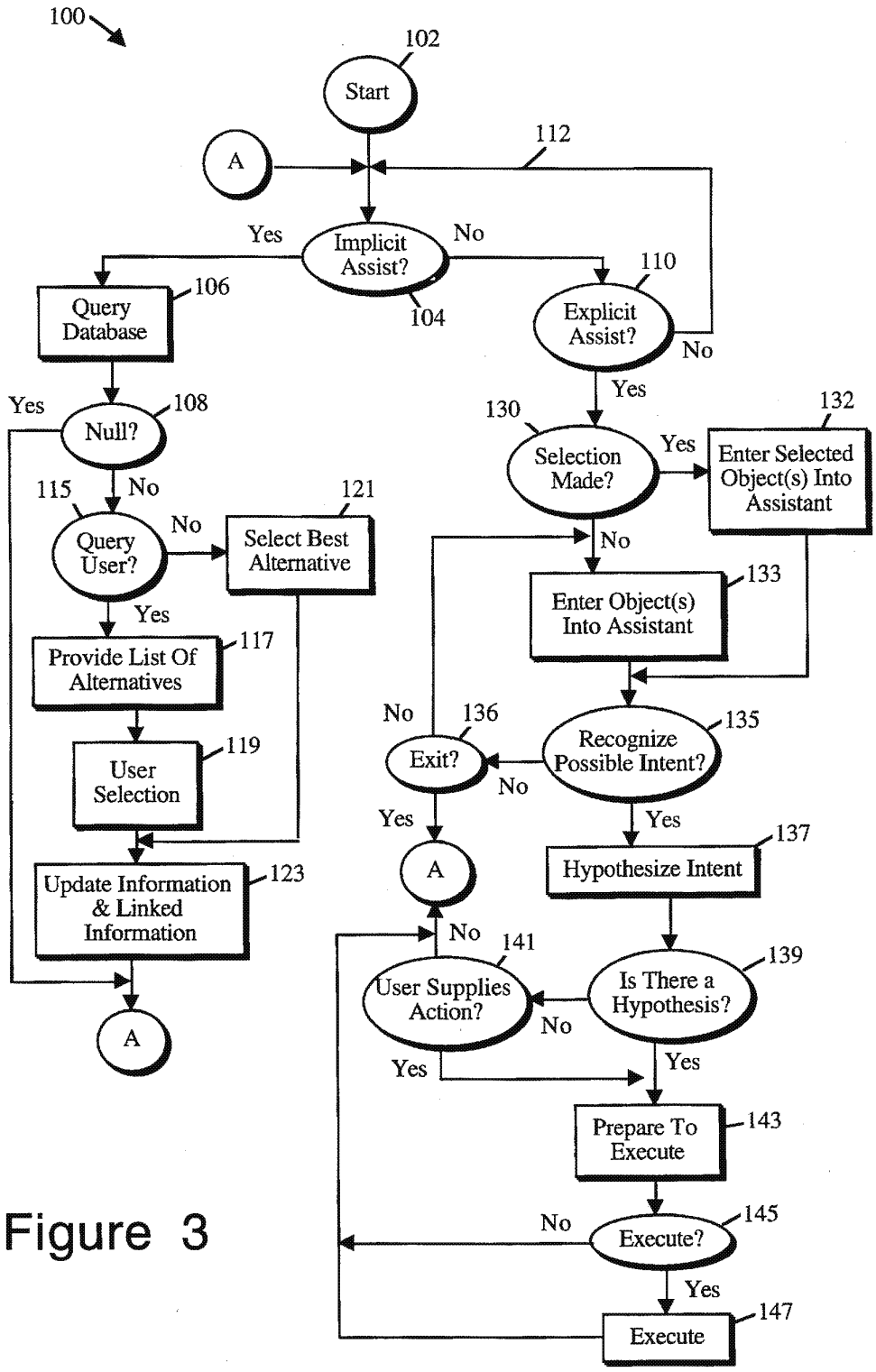

100
102
Start
A
112
Implicit Assist?
Yes
No
104
Query Database
106
Explicit Assist?
110
No
Yes
Null?
108
Yes
130
Selection Made?
Yes
132
Enter Selected Object(s) Into Assistant
No
115
Query User?
No
121
Select Best Alternative
Yes
133
Enter Object(s) Into Assistant
Provide List Of Alternatives
117
User Selection
119
No
136
Exit?
No
135
Recognize Possible Intent?
Yes
Yes
137
Hypothesize Intent
Update Information & Linked Information
123
A
No
141
User Supplies Action?
No
139
Is There a Hypothesis?
Yes
Yes
Prepare To Execute
143
No
145
Execute?
Yes
147
Execute
A

Figure 3

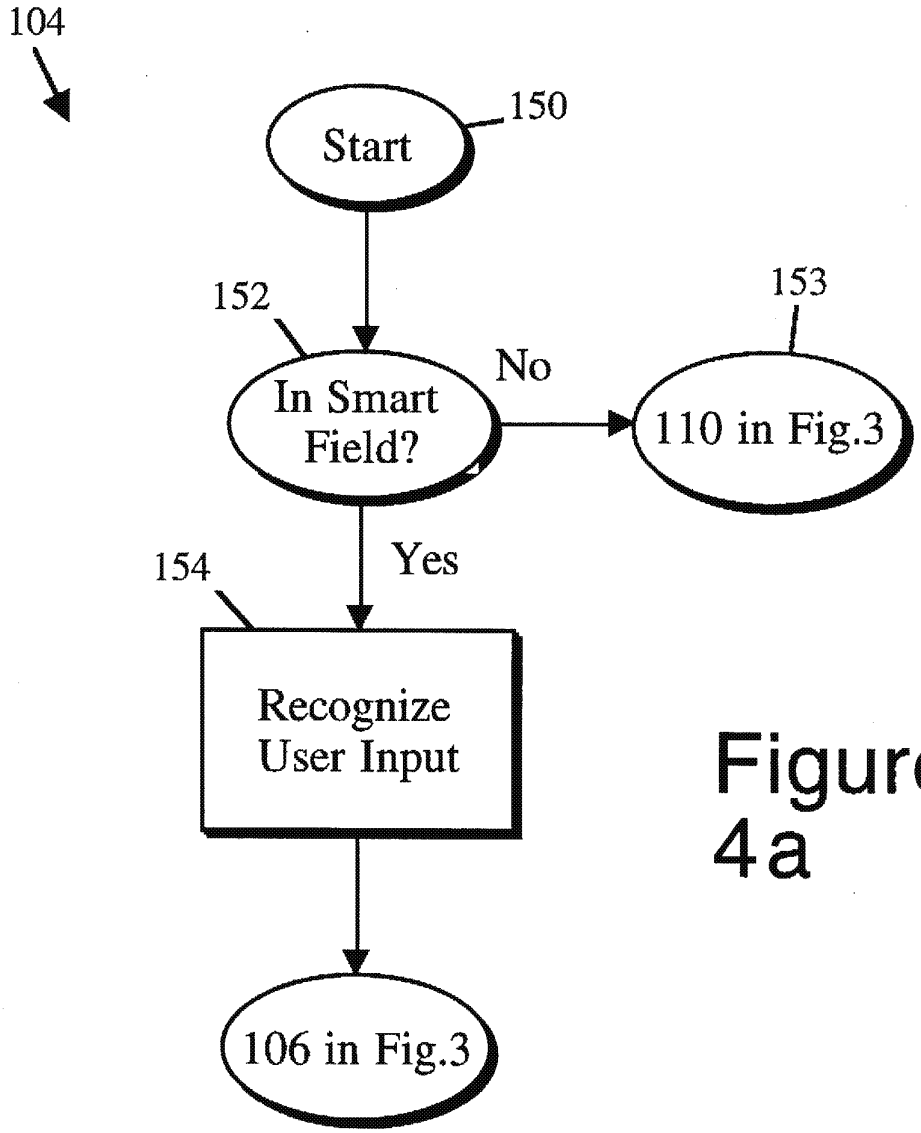

104
Start
150
152
In Smart Field?
No
153
110 in Fig.3
Yes
154
Recognize User Input
106 in Fig.3

Figure 4a

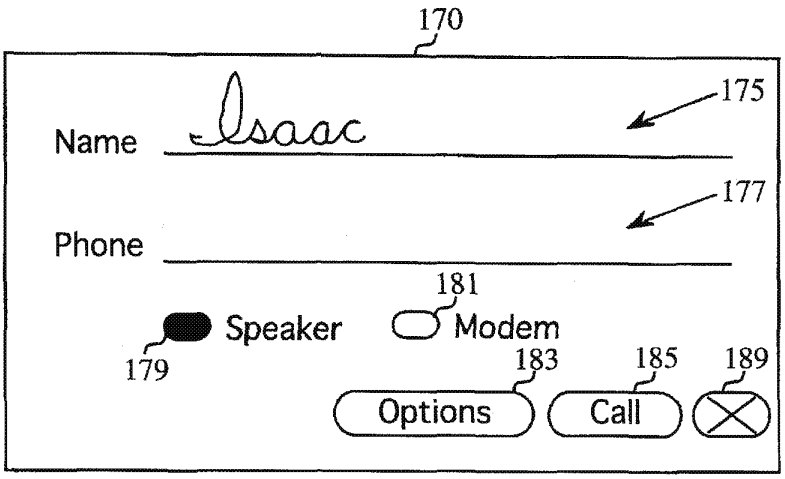



Figure 4b

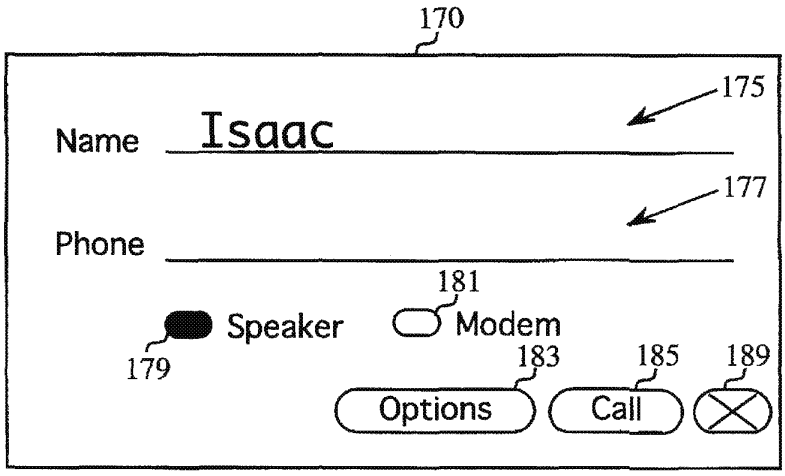



Figure 4c

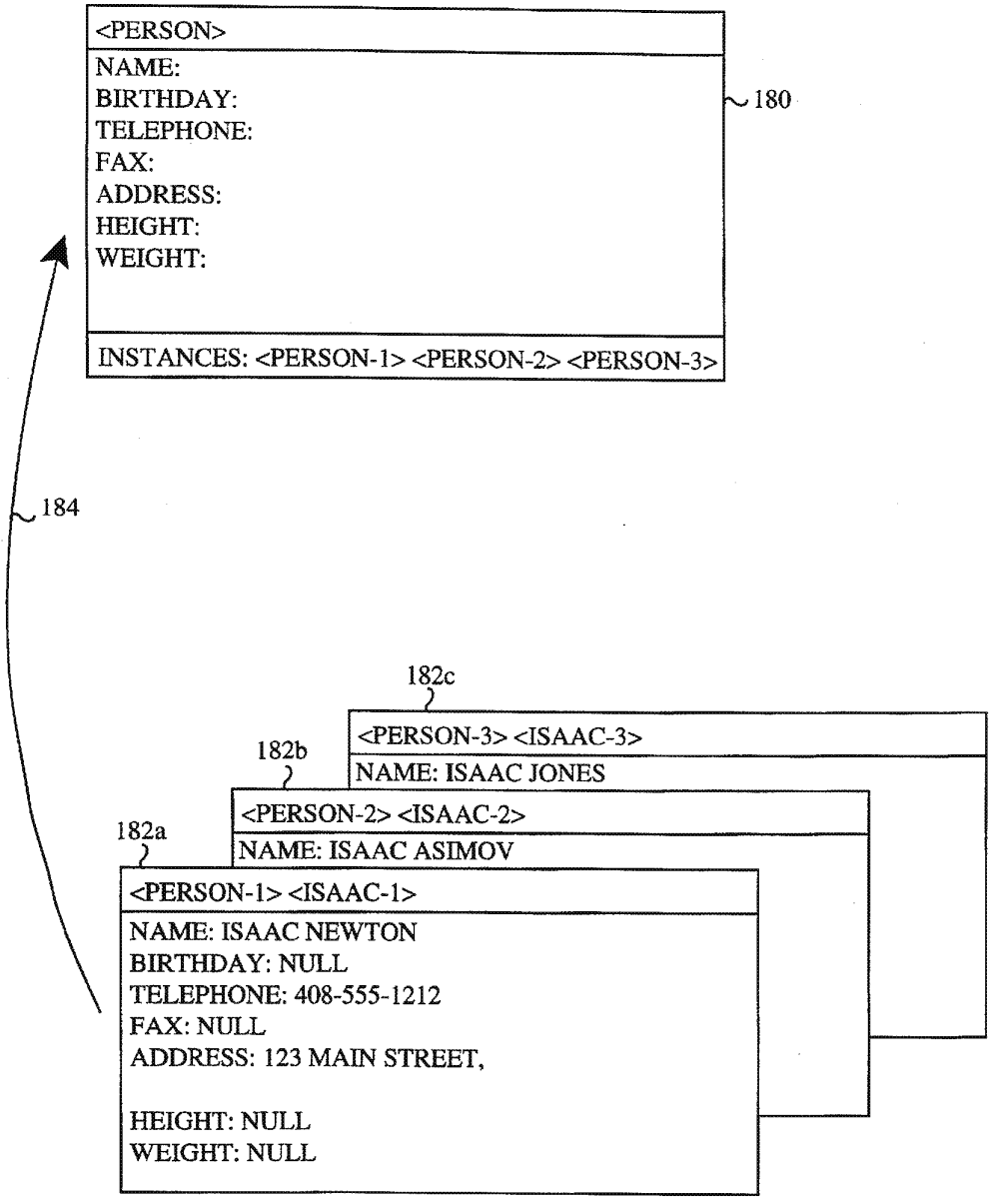

<PERSON>
NAME:
BIRTHDAY:
TELEPHONE:
FAX:
ADDRESS:
HEIGHT:
WEIGHT:
INSTANCES: <PERSON-1> <PERSON-2> <PERSON-3>
180
184
182c
<PERSON-3> <ISAAC-3>
NAME: ISAAC JONES
182b
<PERSON-2> <ISAAC-2>
NAME: ISAAC ASIMOV
182a
<PERSON-1> <ISAAC-1>
NAME: ISAAC NEWTON
BIRTHDAY: NULL
TELEPHONE: 408-555-1212
FAX: NULL
ADDRESS: 123 MAIN STREET,
HEIGHT: NULL
WEIGHT: NULL

*Figure 5*

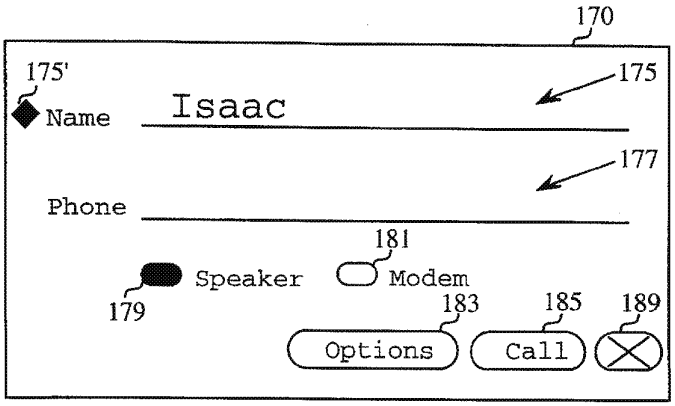



Figure 6a

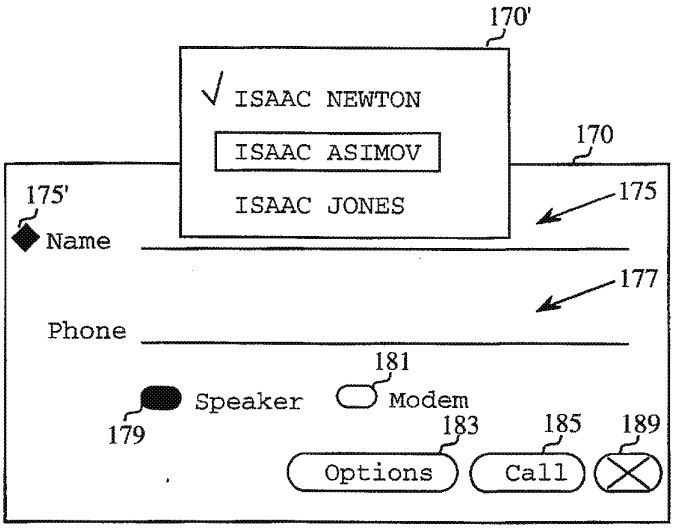



Figure 6b

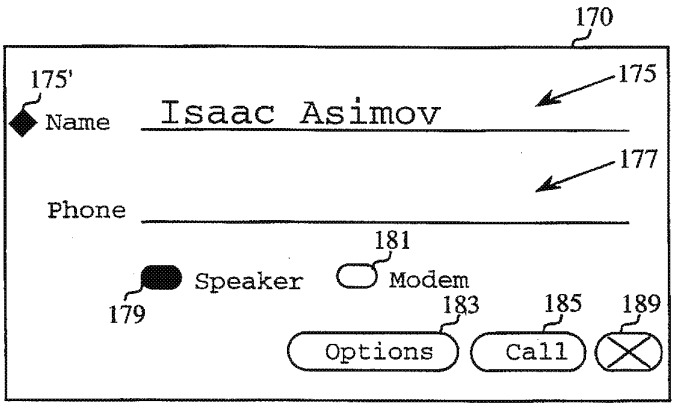



Figure 6c

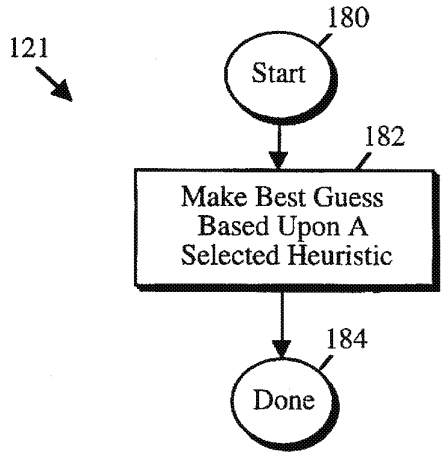



Figure 7

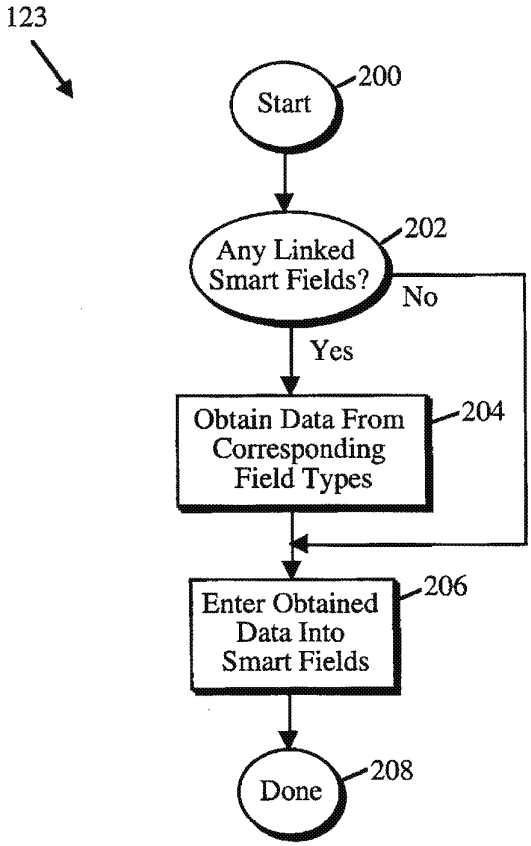



Figure 8a

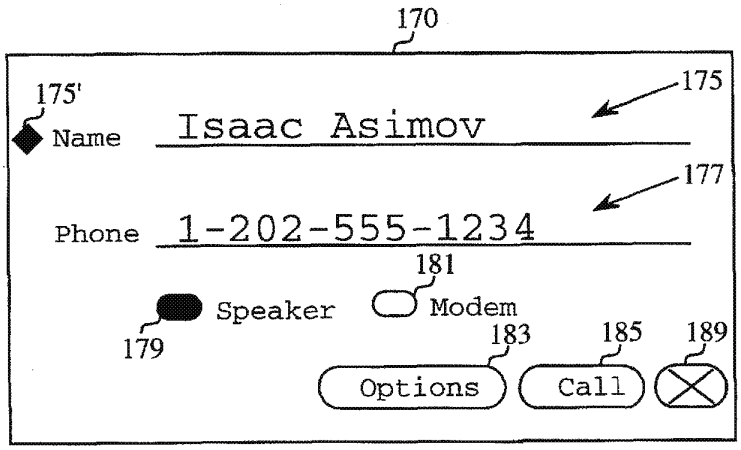

170
175'
175
Name
Isaac Asimov
177
Phone
1-202-555-1234
181
Speaker
Modem
179
183
185
189
Options
Call

*Figure 8b*

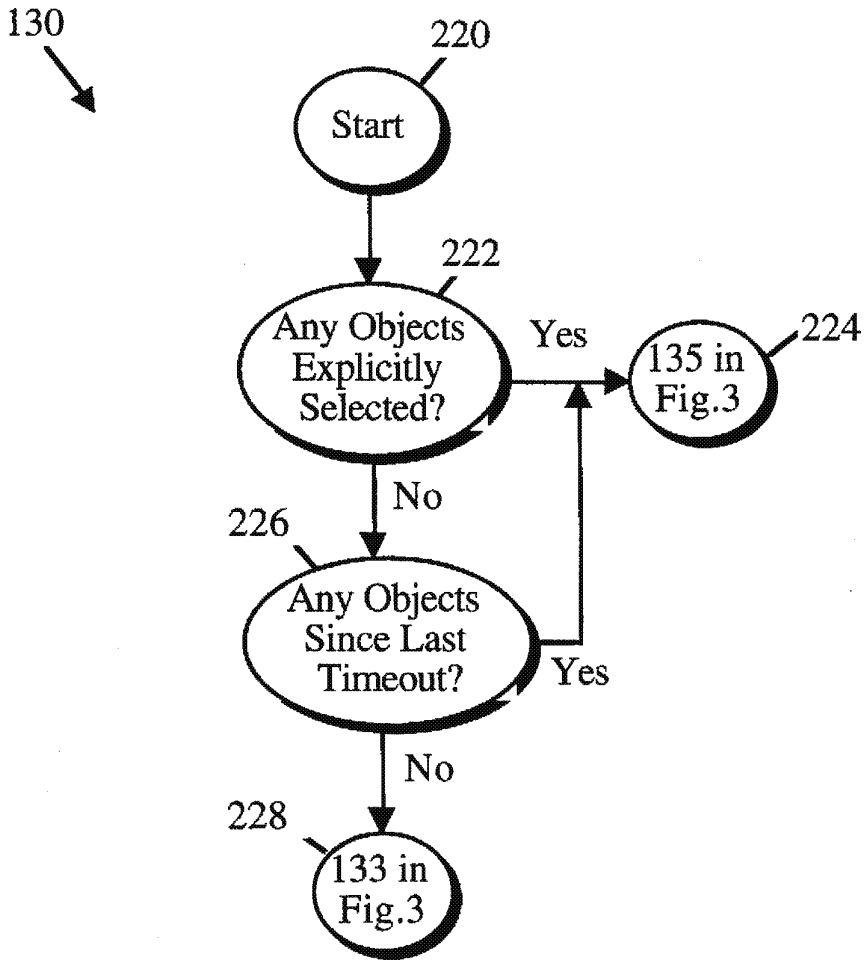

130
220
Start
222
Any Objects Explicitly Selected?
Yes
224
135 in Fig.3
No
226
Any Objects Since Last Timeout?
Yes
No
228
133 in Fig.3

Figure 9a

| INSTANCE | LAST USED |
|---|---|
| ISAAC-1 | T-1 |
| ISAAC-2 | T-2 |
| ISAAC-3 | T-3 |

*Figure 7a*

| INSTANCE | ORDER |
|---|---|
| ISAAC-1 | 1 |
| ISAAC-2 | 2 |
| ISAAC-3 | 3 |

*Figure 7b*

| INSTANCE | TIMES USED |
|---|---|
| ISAAC-1 | 2 |
| ISAAC-2 | 12 |
| ISAAC-3 | 100 |

*Figure 7c*

CALL ISAAC

*Figure 9b*

CALL ISAAC

*Figure 9c*

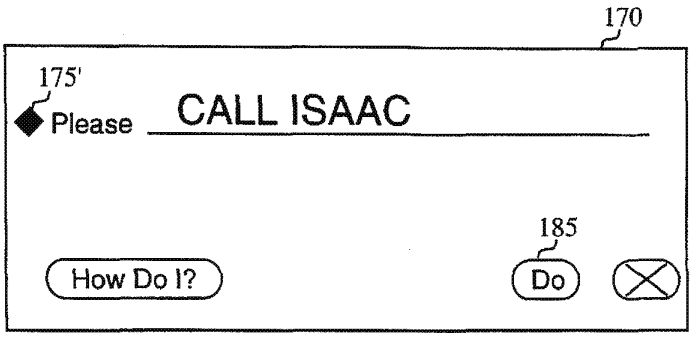



Figure 9d

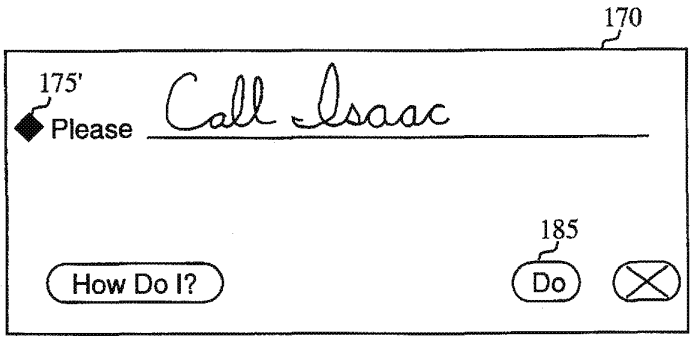



Figure 10a

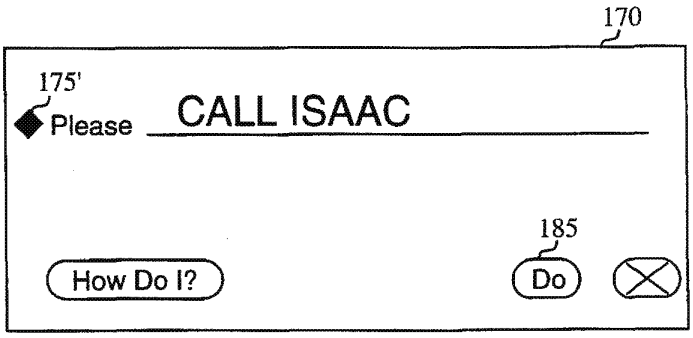



Figure 10b

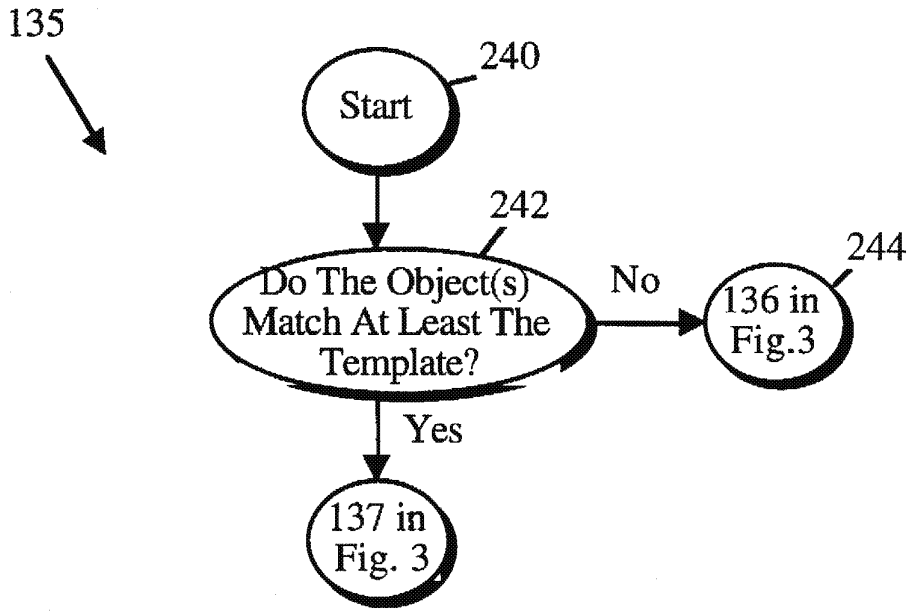

135
240
Start
242
Do The Object(s) Match At Least The Template?
No
244
136 in Fig.3
Yes
137 in Fig. 3

Figure 11a

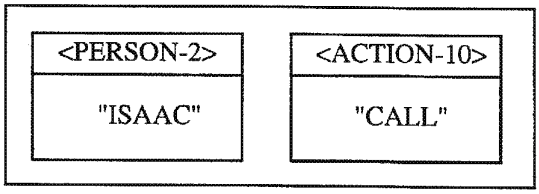



Figure 11b

| | | | |
|---|---|---|---|
| 1 | Scheduling | Meet | Person, Place, Day, Time Slot |
| 2 | Finding | Find | Quantifier, Object |
| 3 | Filing | File | Quantifier, Notes |
| • | | | |
| • | | | |
| • | | | |
| 6 | Formatting | Format | Notes, Form |
| 7 | Mailing | Mail | Person, Place, Letter |
| 8 | Faxing | Fax | Person, Place, Fax #, Notes |
| 9 | Print | Print | Object, Place |
| 10 | Calling | Call | Person, Place, Phone |

Figure 11c

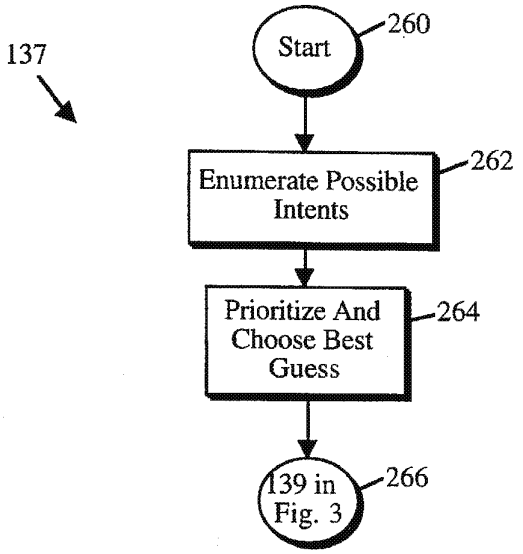



Figure 12a

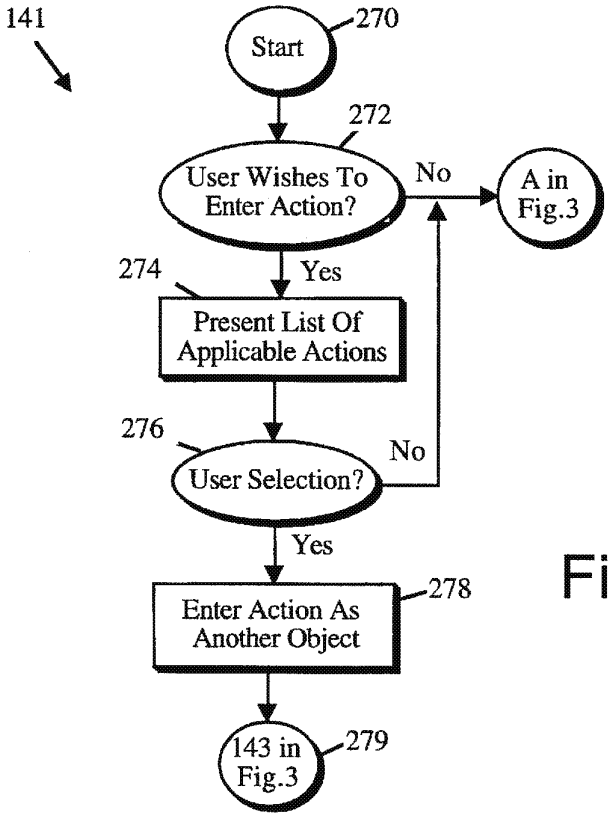



Figure 12b

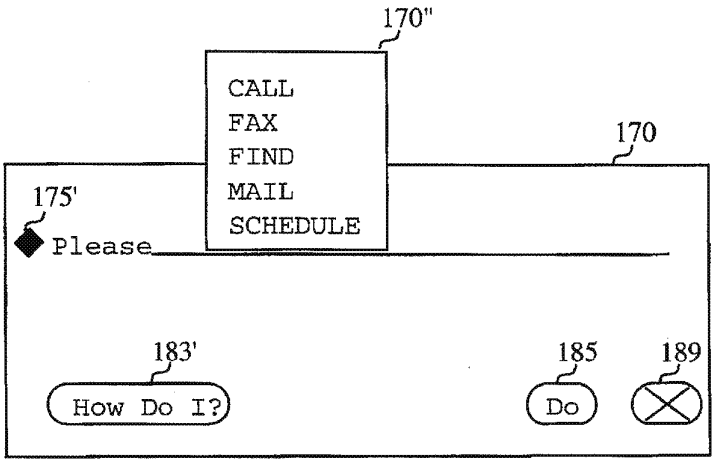



Figure 12c

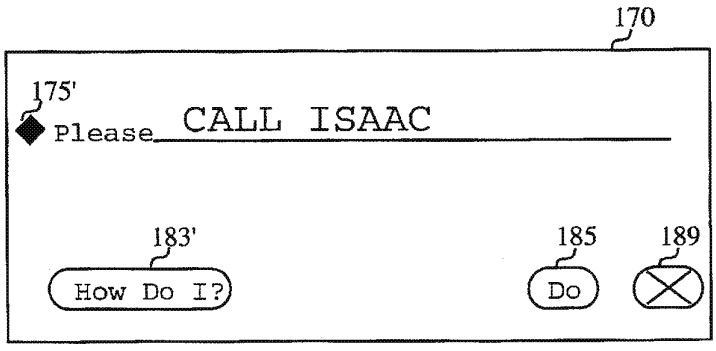



Figure 12d

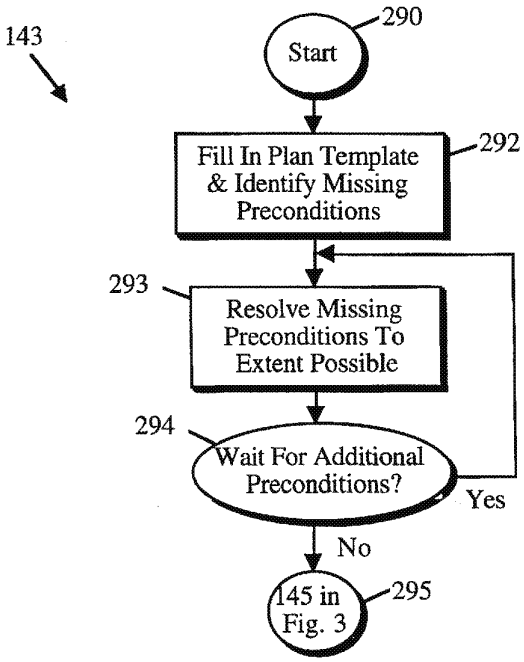



Figure 13

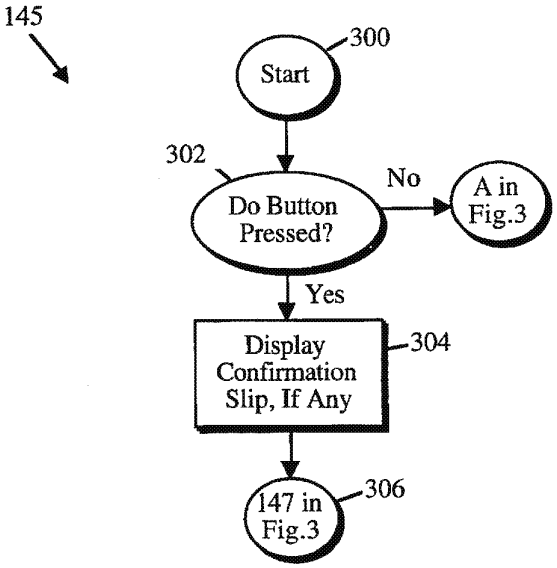



Figure 14a

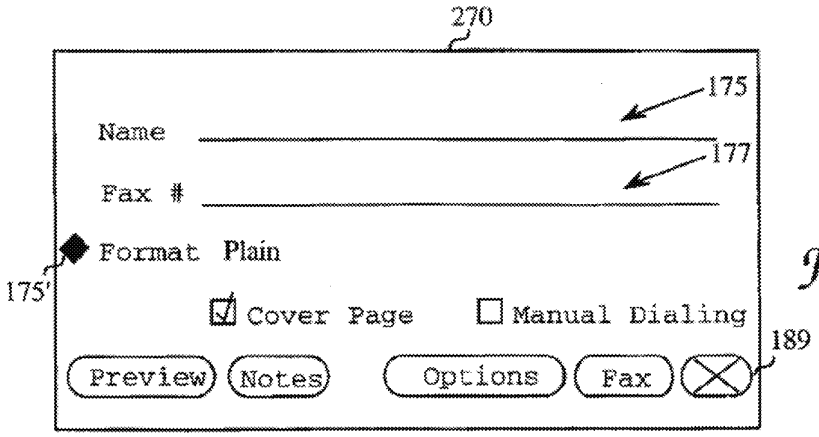

270
175
Name
177
Fax #
Format Plain
175'
Cover Page
Manual Dialing
189
Preview
Notes
Options
Fax

Figure 14b

# METHOD AND APPARATUS FOR PROVIDING IMPLICIT COMPUTER-IMPLEMENTED ASSISTANCE

Cross-Reference to a Related Application

This application is divisional of U.S. patent application Ser. No. 08/099,861, filed on Jul. 30, 1993, now U.S. Pat. No. 5,477,447, under the title "METHOD AND APPARATUS FOR PROVIDING COMPUTER-IMPLEMENTED ASSISTANCE" on behalf of Luciw et. al. and assigned to the same assignee as herein; which application is incorporated herein by reference in its entirety. Application Ser. No. 08/099,861 is a continuation-in-part of Ser. No. 889,225, issued U.S. Pat. No. 5,390,281 filed May 27, 1992 and issued Feb. 14, 1995 and which is incorporated herein by reference in its entirety. Priority rights based upon this earlier filed United States Patent are claimed.

## BACKGROUND OF THE INVENTION

The present invention relates generally to computer systems, and more particularly to computer-implemented assistance methods and apparatus.

Computerized personal organizers are becoming increasingly popular with a large segment of the population. Computerized personal organizers tend to be small, lightweight, and relatively inexpensive, and can perform such functions as keeping a calendar, an address book, a to-do list, etc. While many of these functions can also be provided in conventional computer systems, personal organizers are very well suited to the personal organization task due to their small size and portability. Personal organizers are available from such companies as Sharp and Casio of Japan.

A relatively new form of computer, the pen-based computer system, holds forth the promise of a marriage of the power of a general purpose computer with the functionality and small size of a personal organizer. A pen-based computer system is typically a small, hand-held computer where the primary method for inputting data includes a "pen" or stylus. A pen-based computer system is commonly housed in a generally rectangular enclosure, and has a dual-function display assembly providing a viewing screen along one of the planar sides of the enclosure. The dual-function display assembly serves as both an input device and an output device. When operating as an input device, the display assembly senses the position of the tip of a stylus on the viewing screen and provides this positional information to the computer's central processing unit (CPU). Some display assemblies can also sense the pressure of the stylus on the screen to provide further information to the CPU. When operating as an output device, the display assembly presents computer-generated images on the screen.

The dual-function display assemblies of pen-based computer systems permit users to operate the computer as a computerized notepad. For example, graphical images can be input into the pen-based computer by merely moving the stylus on the surface of the screen. As the CPU senses the position and movement of the stylus, it generates a corresponding image on the screen to create the illusion that the stylus is drawing the image directly upon the screen, i.e. that the stylus is "inking" an image on the screen. With suitable recognition software, text and numeric information can also be entered into the pen-based computer system in a similar fashion.

One approach to computerized assist operations is to provide assistance automatically when a situation in which assistance could be provided is recognized. However, such an approach may provide unsatisfactory results when the user is provided with assistance that is unwanted or disproportionate.



Simply stated, concerns have arisen about assist functions being undertaken by the computer without adequate user control and interaction. When the assist function has been undertaken without adequate user control, assistance would be provided awkwardly and at times when the assistance was not necessarily desired.

It is essential that the control, the timing, and the application of the assistance is considered appropriate and well-tuned. The confidence of the user is undermined when the assistance provided only obliquely addresses particularized user needs without precisely providing the particular results objectively and subjectively required.

## SUMMARY OF THE INVENTION

According to the invention, a method and apparatus has been developed for providing computer-assisted implicit and explicit assistance for a variety of user-supportive information functions. If no implicit assist actions are desired or indicated, then a logical process is initiated to determine whether explicit assistance should be undertaken. If implicit assistance is indicated, a list of action alternatives is displayed for the user. Alternatively, a process can be undertaken to automatically select a best action alternative of several identified alternatives.

If explicit assistance is desired by the user, particular object(s) from which the assistance may be inferred are entered into an assistance operation. An attempt is then made to recognize possible intents expressed by the objects entered into the assistance process. If no user intent is, in fact, recognized, the assist operation is usually terminated. If a possible intent is recognized, the actual intent is hypothesized. A check is further undertaken, to determine whether a hypothesis is in fact available. If no hypothesis is found, the process permits the user to supply a proposed action. If no hypothesis is found and no user action is proposed, assistance efforts terminate. However, if a hypothesis is available, preparations for execution are undertaken. A final inquiry is made as to whether to undertake the hypothesized assist. If the response to an inquiry whether to assist as hypothesized is affirmative, execution of the hypothesized action is undertaken.

Accordingly, it is an intent of the invention to provide user explicit assist functions which are provided under oversight and with the interaction of the user, and implicit assist functions in certain structured instances. The involvement of the user in the implementation of implicit assist operations is a key aspect of the invention. User intent is not merely deduced, but specific user queries are made to ensure controlled application of assist operations.

These and other advantages of the present invention will become apparent upon reading the following detailed descriptions and studying the various figures of the drawings.

## BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of a computer system in accordance with the present invention.

FIG. **2** is a top plan view of the screen, case, and keypad of the computer system of FIG. **1**.

FIG. **3** is a flow diagram of a process according to the invention for providing controlled computer-assisted user assistance.

FIG. **4***a* shows a process for determining whether or not implicit assistance is desired by the user, in connection with providing computer assisted support to the user.

FIG. **4***b* shows an example of an implicit assist operation with a phone slip window having a smart name field evoked, for example, by either highlighting the verb "call" or by writing it on the note field before evoking window.

FIG. **4***c* shows the phone slip window of FIG. **4***b* with the name formal font form. ISAAC having been recognized and established in the name field in

FIG. **5** shows an example of a generic <PERSON> type frame along with a particular set of specific frames of the <PERSON> type.

FIGS. **6***a*–**6***c* show respective assist windows in successive stages of an assist process, including first a window containing a first informational level directed at the name ISAAC alone, a second window with a pop-up menu offering a user choice among several known ISAACs, and a third window showing the selection of a particular ISAAC, that is ISAAC ASIMOV, having been accomplished.

FIG. **7** illustrates a brief flow diagram illustrating a heuristic process for the selection of a particular choice among alternatives when the user is not queried for selection of alternatives.

FIGS. **7***a*–**7***c* show selected examples of heuristic rules of thumb which are effective in making automatic choices between alternative ISAACs, respectively directed toward selection schemes such as selecting the last used ISAAC, selecting the last in order of ISAACs according to particular position within a selected table, and finally selecting a particular ISAAC based upon prior frequency of choice of that particular ISAAC.

FIG. **8***a* is a flow diagram illustrating the updating process for data base information in linked smart fields.

FIG. **8***b* is a call slip illustration of an updated smart field window in which the phone number field information has been updated.

FIG. **9***a* is a flow diagram of the process according to the invention in which a query is made as to whether a specific selection has been made as to a particular object.

FIG. **9***b*–**9***c* indicate graphically the performance of the selection query operation as expressed in FIG. **9**A.

FIG. **9***d* illustrates the transferal of the highlighted objects of FIG. **9***c* transferred to a selected window.

FIG. **10***a* illustrates the input of a handwritten object into a smart field in a window.

FIG. **10***b* illustrates the recognition of the handwritten object of FIG. **10***a* arid its conversion into formal font form.

FIG. **11***a* is a flow diagram illustrating the recognition of objects process.

FIG. **11***b* illustrates an object combination under operation.

FIG. **11***c* shows a template for organizing in preset form a variety of object combinations.

FIG. **12***a* illustrates a process for hypothesizing user intent as to particular activities.

FIG. **12***b* is a flow diagram setting forth a process for determining whether the user wishes to provide or supply a particular action.

FIG. **12***c* shows a window with a menu partially overlapping its topside in order to provide the user with an array of activity choices.

FIG. **12***d* shows the window of FIG. **12***c* with the activity of calling having been selected, establishing the combined objects CALL and ISAAC in the activity field of the window.

FIG. **13** illustrates the process for preparing for execution of a particular activity.

FIG. **14***a* is a short flow diagram of a process for determining whether to proceed with execution of a particular selected assist activity.

FIG. **14***b* illustrates an example of a confirmation of action slip that could be produced upon completion of a particular activity, in this case completion of the process of faxing information to another party.

## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention is well suited for pointer based computer systems such as the pen-based, pen-aware and mouse controlled systems that are currently popular. For the purposes of illustration, the invention will be described in connection with a pen-based system.

As shown in FIG. **1**, a pen-based computer system **10** in accordance with the present invention includes a central processing unit (CPU) **12**, read only memory (ROM) **14**, random access memory (RAM) **16**, input/output (I/O) circuitry **18**, and a display assembly **20**. The pen-based computer system **10** may also optionally include a mass storage unit **22**, a keypad (or keyboard) **24**, a serial port **26**, an infrared (I/R) port **28**, and a clock **30**.

The CPU **12** is preferably a commercially available, single chip microprocessor. While CPU **12** can be a complex instruction set computer (CISC) chip, it is preferable that CPU **12** be one of the commercially available, reduced instruction set computer (RISC) chips which are known to be of generally higher performance than CISC chips. CPU **12** is coupled to ROM **14** by a unidirectional data bus **32**. ROM **14** contains the basic operating system for the pen-based computer system **10**. CPU **12** is connected to RAM **16** by a hi-directional data bus **34** to permit the use of RAM **16** as scratch pad memory. ROM **14** and RAM **16** are also coupled to CPU **12** by appropriate control and address busses, as is well known to those skilled in the art. CPU **12** is also coupled to the I/O circuitry **18** by bi-directional data bus **36** to permit data transfers with peripheral devices.

I/O circuitry **18** typically includes a number of latches, registers and direct memory access (DMA) controllers. The purpose of I/O circuitry **18** is to provide an interface between CPU **12** and such peripheral devices as display assembly **20**, mass storage **22**, keypad **24**, serial port **26**, and I/R port **28**.

Display assembly **20** of pen-based computer system **10** is both an input and an output device. Accordingly, it is coupled to I/O from a variety of vendors. The input device of display assembly **20** is preferably a thin, clear membrane which covers the LCD display and which is sensitive to the position of a stylus **38** on its surface. With such a structure, the display assembly **20** can serve as an input "tablet." These position sensitive membranes are also readily available on the commercial market. Alternatively, other types of tablets can be used, such as inductively coupled tablets. Combination display assemblies such as display assembly **20** which include both the LCD and the input membrane are commercially available from such vendors as Scriptel Corporation of Columbus, Ohio.

Some type of mass storage **22** is generally considered desirable. Mass storage **22** can be coupled to I/O circuitry **18** by a bi-directional data bus **40**. However, the mass storage **22** can be eliminated by providing a sufficient amount of RAM **16** to store user application programs and data. In that case, the RAM **16** can be provided with a backup battery to prevent the loss of data even when the pen-based computer

system **10** is turned off. However, it is generally desirable to have some type of long term mass storage **22** such as a commercially available miniature hard disk drive, nonvolatile memory such as flash memory, battery backed RAM, a Personal Computers Memory Card International Association (PCMCIA) card, or the like.

The keypad **24** can comprise an array of mechanical buttons or switches coupled to I/O circuitry **18** by a data bus **42**. Alternatively, keypad **24** can comprise an entire, standard QWERTY keyboard. In the present embodiment, a separate keypad **24** is not used in favor of a "pseudo" keypad **24'**. This "pseudo" keypad **24'** comprises "button" areas which are associated with a bottom edge of the tablet membrane that extends beyond the lower edge of the LCD display. These button areas are defined by a printed or silk-screened icons which can be seen through the transparent membrane of the input tablet. When the "buttons" are selected by engaging the stylus **38** with the membrane over these printed icons, the membrane senses the pressure and communicates that fact to the CPU **12** via data bus **38** and I/O **18**. An example of pseudo keypad **24'** is shown in FIG. **2**.

Other types of pointing devices can also be used in conjunction with the present invention, for example, an interrupt port of the CPU **12** which can count the clock pulses to provide the time function. However, this alternative clock embodiment tends to be wasteful of CPU processing power. Clock **30** is coupled to CPU **12** by a data bus **48**.

In operation, information is input into the pen-based computer system **10** by "writing" on the screen of display assembly **20** with the stylus **38**. Information concerning the location of the stylus **38** on the screen of the display assembly **20** is input into the CPU **12** via data bus **38** and I/O circuitry **18**. Typically, this information comprises the Cartesian (i.e.×& y) coordinates of a pixel of the screen of display assembly **20** over which the tip of the stylus is positioned. Commercially available combination display assemblies such as the aforementioned assemblies available from Scriptel Corporation include appropriate circuitry to provide the stylus location information as digitally encoded data to the I/O circuitry of the present invention. The CPU **12** then processes the data under control of an operating system. While the method of the present invention is described in the context of a pen-based system, other pointing devices such as a computer mouse, a track ball, or a tablet can be used to manipulate a pointer on a screen of a general purpose computer. Therefore, as used herein, the terms "pointer", "pointing device", "pointing means", and the like will refer to any mechanism or device for pointing to a particular location on a screen of a computer display.

Serial port **26** is coupled to I/O circuitry by a bi-directional bus **44**. The serial port **26** can be used to couple the CPU to external devices and networks.

Infrared (I/R) port **28** is coupled to I/O circuitry by a bi-directional bus **46**. The I/R port can be used for outgoing information (e.g. to control a printer or some other external device, or to communicate with other computer systems) or for incoming information from other computers or devices.

Clock **30** preferably comprises a real-time clock to provide real-time information to the system **10**. Alternatively, clock **30** can simply provide regular clock pulses td possibly an application program stored in ROM **14**, RAM **16**, or mass storage **22**. The CPU **12** next produces data which is transferred to the display assembly **20** via I/O circuitry **18** and data bus **38** to produce appropriate images on the screen portion of the display assembly **20**.

In FIG. **2**, the pen-based computer system **10** of FIG. **1** is shown housed within a generally rectangular enclosure **50**. The CPU **12**, ROM **14**, RAM **16**, I/O circuitry **18**, and clock **26** are preferably fully enclosed within the enclosure **50**. The display assembly **20** (FIG. **1**) is mostly enclosed within the enclosure **50**, but a viewing screen **52** of the display assembly is exposed to the user. As used herein, the term "screen" will refer to the portion of the display assembly **20** which can display an image that can be viewed by a user. Also accessible to the user is the pseudo keypad **24'** that was described with reference to FIG. **1**.

Upon power-up, pen based computer system **10** displays on screen **52** an initial "note" area **54***a* including a header bar **56***a* and a number of guidelines **58**. The header bar **56***a* preferably includes the date of creation of the note area **54***a* and a number of icons and "soft" buttons, not particularly germane to the discussion of the present invention. For this reason, the header bar **56***a* will not be discussed in detail herein. The optional guidelines **58** aid a user in entering text, graphics, and data into the pen-based computer system **10**. A graphic object G in the form of a triangle is shown entered within note area **54***a*.

Additional note areas, such as a note area **54***b*, can be formed by the user by drawing a substantially horizontal line across the screen **52** with the stylus **38**. The substantially horizontal line is recognized by the system **10** and is converted into a second header bar **56***b*. Additional text, graphical, and other data can then be entered into this second note area **54***b*. For example, the text object T comprising "ISAAC" has been entered into second note area **54***b*.

In this preferred embodiment, the keypad **24'**, as explained previously, comprises a printed or silk-screened member **60** provided beneath a lower edge of a thin, clear, stylus-sensitive membrane **62** of the input "tablet." Alternatively, a keypad could comprise a mechanical keypad (or keyboard) **24**, or a keypad could comprise "soft buttons" i.e. images generated at convenient locations on the screen **52**, in which case a "button" would be activated by touching the stylus to the screen over the image of the button. The keypad **24'** preferably includes a number of dedicated function buttons **64** and a pair of scroll buttons **66***a* and **66***b*. The operation of the note areas **54***a*, **54***b*, etc., scroll buttons **66***a* and **66***b*, and other aspects of computer system **10** are discussed in greater detail in co-pending U.S. patent application 07/868,013, filed Apr. 13, 1992 on behalf of Tchao et al., assigned to the assignee of the present invention and incorporated herein by reference in its entirety.

The screen illustrated in FIG. **2** is referred to as the "notepad", and is preferably an application program running under the operating system of the pen based computer system **10**. In this preferred embodiment, the notepad is a special or "base" application which is always available beneath higher level applications. The notepad application, like other applications, runs within a window, which in this instance comprises the entire viewing screen **52**. Therefore, as used herein, a "window" is the entire screen or any portion of an entire screen which is dedicated to a particular application program.

A status bar **68** is provided at the bottom of the notepad application. The status bar **68** is provided with a number of active and display areas, which again are not particularly germane to the present invention and will therefore not be discussed in detail herein. U.S. patent application Ser. No. 07/976,970 filed Nov. 16, 1992 on behalf of Foster et. al, entitled "Status Bar for Application Windows" and assigned to the assignee of the present invention describes how to

make and use the status bar, and is incorporated herein by reference in its entirety.

The enclosure **50** is preferably provided with apertures **70** which permit the free transmission of sound from a speaker (not shown) which is housed within enclosure **50**. The speaker can be driven by the CPU **12**, by I/O circuitry **18**, or by specialized sound chips, as is well known to those skilled in the art. The speaker can be used to provide user feedback, or to transmit audible information to a user.

The term "object" will be used extensively in the following discussions. As is well known to software developers, an "object" is a logical software unit comprising data and processes which give it capabilities and attributes. For example, an object can be queried as to its type and can return such data as the number of words that it contains, what its bounding box (BBOX) is, etc. Objects can contain other objects of the same or of a different type. Objects can also be used to project images on a screen according to their object type. Example of object types used in the following description include paragraph, line, and word objects. There are many well known texts which describe object oriented programming. See, for example, *Object Oriented Programming for the Macintosh*, by Kurt J. Schmucher, Hayden Book Company, 1986.

In the present invention, objects are preferably implemented as part of a frame system that comprises frame objects related by a semantic network. A description of semantic networks can be found in "A Fundamental Tradeoff in Knowledge Representation and Reasoning", *Readings in Knowledge Representation*, by Brachman and Leveseque, Morgan Kaufman, San Mateo, 1985.

It will be noted there is a liberal use of graphic elements in the present invention. For example, the header bars **56***a* and **56***b* include lines and other graphical elements. Processes for drawing lines on a computer screen are well known to those skilled in the art. For example, graphics software such as QUICKDRAW from Apple Computer, Inc. of Cupertino, California can be used to draw lines, simple geometrical shapes, etc. A description of the QUICKDRAW graphics software is found in the book *Inside Macintosh, Volumes I, II, and III*, by C. Rose et al., Addison-Wesley Publishing Company, Inc., July 1988. With such graphics software, a line can be drawn by simply specifying the coordinates of the beginning and the end of the line, and by specifying the width of the line.

Another preferred tool for implementing the system of the present invention is a view system. Various types of view systems are well known to those skilled in the art. In the present system, the notepad application on the screen **52** forms a first or "root" layer, with the status bar **68**, for example, positioned in a second layer "over" the root layer. The various buttons of the status bar **68** are positioned in a third layer "over" the second and root layers. The view system automatically handles "taps" and other gestures of the stylus **38** on the screen **52** by returning information concerning the tap or gesture and any object to which it may be related. Again, the status bar **68** and the view system is described in greater detail in co-pending U.S. patent application 7/976,970, which has been incorporated herein by reference. It is therefore clear that the object oriented programming and view system software makes the implementation of the processes of the present invention less cumbersome than traditional programming techniques. However, the processes of the present invention can also be implemented in alternative fashions, as will be well appreciated by those skilled in the art.

A method or process **100** for providing implicit or explicit assistance in the provision of computer implemented services in accordance with the present invention is shown in FIG. **3**. The process begins at step **102** on power-up of the computer system **10** and runs concurrently with other system functions.

At step **104**, the process recognizes whether or not an implicit assistance function is to be provided by computer system **10**. As will be seen, implicit assistance may, for example, arise from an entry into a smart field by a users. If a user does enter information into a "smart field," the computer database will be queried at step **106** to determine whether assistance is possible given the user input.

A smart field is considered to be a predefined region on screen **52** of computer system **10** shown in FIG. **2**, or a predefined region within a window which appears on screen **52**, as suggested below with reference to FIG. **46***b* and which will be discussed in greater detail in the text description below associated with that Figure. For convenience, the smart fields are typically rectangular in shape. The particular geographic bounds of a smart field can conveniently be stored in computer memory by simply saving four numbers defining the corners of the rectangular shape of the field. A particular field is considered smart, because of the specialized capabilities of the smart field to respond with particularized effectiveness and intelligence to user needs, indications, or events registered, for example, by pen **38**, within the bounds of the particular smart field.

However, implicit assist may be indicated not just by entry of an indication in a smart field, but by the happening of any of a number of predefined allowable events which lead to a query of the database at process step **106**. A user entry made into a smart field is not the only way computer system **10** is caused to undertake an implicit assist operation. Certain kinds of events on screen **52**, for example, such as the writing of a particular indication or word on screen **52** outside of a particular smart field may trigger an implicit assist. In general, implicit assist can be triggered by the happening of any of a number of predefined allowable events.

If, however, a decision is made at decision process **104** not to perform the implicit assist function or approach suggested in FIG. **3**, a check is made at decision process **110** whether an explicit assist function should be undertaken. If neither implicit or explicit assist is indicated at decision processes **104** and **110**, operation continues past point A and along line **112** repeatedly checking for an indication whether implicit or explicit assistance is required, at the two decision processes **104** and **110**.

An example of an indication of user desire to have explicit assistance undertaken is the act of using pen **38** in FIG. **2** to tap or click on the assist icon or button **64** shown on the surface of stylus-sensitive membrane **62** or a keypad **24** including a range of dedicated function buttons **64**. If the query at process step **108** produces a negative, i.e., null, response to the question of whether any implicit assist actions are available in the database of computer **10**, indicating that no assistance actions are identified for performance, then process control returns to point A.

If a non-null result to the query for implicit assistance has been established by step **108**, a determination is made at process step **115** as to whether the user should be queried. This determination can be made by the system, or can be set by the user in a preference field. If it is decided that the user should be queried, step **117** displays a list of action alternatives for user selection. The user can make the selection of

a particular assistance action according to step **119**, by highlighting the particular course of action selected, for example. Alternatively, if it is desired that no user query is desired for selection of a particular mode of assistance, then a process is undertaken at step **121** to select the best alternative.

In either the case of an automatic selection of a best alternative according to process step **121** or in the case of steps **117** and **119** involving a presentation of alternatives to the user followed by user selection of a particular alternative, i.e., whether a user query has been undertaken, or whether an automatic selection of a best alternative has been made, upon accomplishment of the selected assistance action, the database information and any linked information are updated at step **123**. Once implicit assistance has been completed, process control returns to point A. If step **104** determines that there is no implicit assist, then step **110** determines if there is an explicit assist to be undertaken. For example, if the name ISAAC had been entered on screen **52** of FIG. **2** and no time-out had occurred, and additionally the assist button (the rightmost of buttons **64**) had been clicked, then an explicit assist will have been requested. If an explicit assist has been indicated at step **110**, then a step **130** determines, if a particular selection as to the explicit assistance has been made. Selection is typically made by clicking on a particular word with pen **38**, for example, and thereby highlighting the item selected. If a user selection has been made, particular selected objects are entered into the assistance operation in step **132**. If no user selection has been made, objects entered since a delimiter are entered into the assistant in a step **133**. Since no objects have specifically been selected, the objects to be entered into the assistant are selected automatically by a delimiter process.

An example of how the delimiter process can be accomplished, for example, involves the entry of only those objects on the screen **52** which are delimited in some fashion from the other objects which may have been entered on the screen. For example, if several paragraphs have been entered on the screen, only the last paragraph's objects will be considered for entry as objects into the assistant. Time may also be used as a delimiter. For example, if a considerable period of time separates a given object on the screen from another, only the most recent object will be entered into the assistant. The time threshold separating the particular objects may for example be a pre-set time-out.

Next, an attempt is made at step **135** to recognize the possible intent expressed by the objects entered into the assistance process. If no intent is recognized, the explicit assistance operation is considered for local termination by return of process control to point A, according to decision process **136**, or for continuation by return to the top of step **133** to wait for new objects which might permit intent to be recognized or to await user or process intervention.

If a possible intent is recognized at step **135**, the intent is hypothesized at step **137**. Next, step **139** checks whether there is a hypothesis available. If no hypothesis is found, process step **141** permits the user to supply a proposed action. If no hypothesis is found and no user action is proposed, process control returns to point A. However, with the availability of a user supplied action or if a hypothesis is available, preparations for execution of actions consistent with the action requested or hypothesized are undertaken at step **143**. A final inquiry is raised at step **145** as to whether to undertake the ordered or hypothesized assistance action. This final inquiry may entail simply permitting the user to supply a verb such as "call" for example, before the calling process, if that is the action hypothesized, is actually launched. If the response to the inquiry is affirmative, execution is undertaken at step **147**. Process control returns to point A in either case after execution by step **147** or by refusal to execute according to a decision made by process **145**.

FIG. **4***a* shows an example of a process for determining whether or not implicit assistance is desired at step **104**. Once the question has arisen whether or not implicit assistance is desired, the determination process under FIG. **4***a* is undertaken or started at step **150**. Implicit assistance is considered to be desired, for example, if a handwritten entry is made in a smart field as per step **152**. If the result is that no implicit assistance is desired, then a query at a step **153** about explicit assistance is undertaken as previously suggested at step **110**. If the entry in the smart field has been made by the user, the assistance process takes action to identify or recognize the kind of implicit assistance indicated at a step **154**. After recognition has been accomplished, operation continues as suggested in FIG. **3** at step **106** with a query of the database.

An example of an implicit assist operation is provided with reference to FIG. **4***b*. The Figure shows a phone slip window **170** with a smart name field **175** which has for example been evoked by either highlighting the verb "call" or by simply writing the word on the display surface either before or after establishment of window **170**. Once the particular window **170** is presented to the user, the name ISAAC can be handwritten into the particular smart field **175**. The assistance process recognizes the handwritten name "Isaac," and either continues operation as suggested at step **106** in FIG. **3** directly, or concurrently displays the recognized name in formal font form, as suggested in FIG. **4***c*, in the same position of the smart field, where formerly the handwritten name "Isaac" had been established. As will readily be recognized, window **170** in FIG. **4***b* may contain several smart fields, in this case for example definable for either the "name" field **175** or a "phone" field shown at step **177**. By way of an aside, it is further noted on the face of window **170**, that a speaker block **179** has been selected, indicating, for example, that a tone produced by a speaker element (not shown) is capable of being evoked. Alternatively, a modem option, indicated at step **181**, can be selected. Further options can be displayed in a pull-down menu button **183** entitled "options" at step **183** which can be presented as a help menu. Additionally, a "call" activity can be undertaken by selecting a "call" button **185** indicated on the face of window **170**. Finally, window **170** can be closed simply by selecting the "x" block shown in window **170**.

Details of one way to carry out the database query process indicated in FIG. **3** at step **106** can be understood in connection with FIG. **5**. In particular, FIG. **5** illustrates a frame **180** which is a special case of a frame, referred to commonly as a "type" frame, as the frame refers to a particular type, i.e., the type <PERSON>. Particular instances of the type <PERSON> are shown as frames. Frame **180** has a number of slots for various attributes of being a person, such as NAME, BIRTHDAY, TELEPHONE, FAX, etc. Frame **180** also includes a list of all frames which are an instance of the type frame <PERSON>, namely <PERSON-1>, <PERSON-2>, and <PERSON-3>.

Frames **182***a*, **182***b*, and **182***c* are "instance" frames of the type <PERSON>. For example, frame **182***a* is an instance <PERSON-1> of frame **180**, and has the name <ISAAC-1>. Frame **182***a* is provided with the same slots as frame **180**, except some of the slots are filled with data. For example, the NAME slot of frame **182***a* is filled with the name "ISAAC NEWTON." Untilled slots have a null value or can

be set to a default value. Likewise, frames **182***b* and **182***c* have slots filled with data about their ISAACs. As depicted by the arrow **184**, there is a "IS-A" link between the instances frames **182***a*–**182***c* and the type frame Semantic networks are well known to those skilled in the art of building knowledge bases. A description of semantic networks can be found in "A Fundamental Tradeoff in Knowledge Representation and Reasoning", *Readings in Knowledge Representation*, by Brachman and Leveseque, Morgan Kaufman, San Mateo, 1985. The frame system is an elaboration of a semantic network. See, Brachman and Leveseque, supra. Frame systems elaborate on the structure of types themselves and their attributes. Specifically, frame systems have three aspects:

1) Values of attributes are stored explicitly or stored as a default value that the individual slot can inherit. This effectively caches some of the graph traversal.
2) Value or role restriction are constraints that must be satisfied by attribute values. These restrictions can constrain a value to be of a certain type (known as value class), of a certain maximum or minimum cardinality (in the case of multivalued slots), or a combination of both.
3) Attached procedures (also known as daemons or angels) that are invoked when a values is accessed (either by getting or setting). This allows values to be computed on-the-fly. The procedures can be completely arbitrary or be expressed in a restricted language. In either case, the procedure returns the computed value and whatever side-effects that may have occurred.

Frames used in the present invention have a number of slots which may contain data, daemons, or other frames. Slots are accessed by making assertions to the knowledge base. For example, if it was desired to retrieve all of the frames that were colored red, a typical frame accessor language query would be in the form of:

(QUERY (MEMBER-VALUE COLOR ?×RED)

and would return a list of frames that have a COLOR slot whose value is red. Compound queries can also be made.

Shown in FIGS. **6***a*–**6***c* is the process of user selection of a particular assistance option indicated at steps **117** and **119** in FIG. **3**. The phone slip window **170** in FIG. **6***a* is shown with a smart name field **175**. The name ISAAC has been recognized in smart field **175** and displays the recognized name in formal font form. As noted above, window **170** in FIG. **4***b* contains an additional smart field, i.e., "phone" field **177**. Additionally, speaker block **179** has been selected, indicating that a tone is capable of being evoked. Alternatively, a modem option, indicated at step **181**, can be selected. Further options can be displayed in a pull-down menu button entitled "options" at step **183** which can be presented as a help menu. As already noted above, a "call" activity can be undertaken by selecting "call" button **185** indicated on the face of window **170**. Window **170** can be closed simply by selecting the "x" block shown in window **170**. Significantly, to the left of the name field is a diamond icon **175'** which can be invoked to produce a pull-down menu of selection items (not shown) which permit the user to initiate further assistance operations.

FIG. **6***b* illustrates a presentation of assistance options to the user in connection with step **117** in FIG. **3**. Responsive to the recognition of the name ISAAC, the assistance process has produced a list of alternatives by earlier query of the database per step **106** in FIG. **3**. In particular, three ISAAC are presented for selection of one of them, the presentation being made in an overlay window **170**, positioned partially over the underlying window **170**. The user-selected "ISAAC ASIMOV" is shown having been marked for selection by a rectangle indicating a highlighting operation. FIG. **6***c* illustrates the completion of the selection process, with the full name in formal font of ISAAC ASIMOV being presented in the name field **175** of window **170**.

FIG. **7** illustrates a basic process which can be used to implement selection of a best alternative absent user selection, as suggested at step **121** of FIG. **3**. Simply stated, the process calls for applying a particular selected procedure for choosing among one of several options, once the process is initiated at step **180** in FIG. **7**. The selection process for example entails making a best guess based upon a selected heuristic approach as would be well-known to one skilled in the art, as suggested at step **182**. This heuristic approach may for example follow the approach suggested in any one of FIGS. **7***a*–**7***c*.

FIG. **7***a* presents a "last used" selection scheme for determining which of several alternatives automatically to select. Three instances are presented for selection, in this case, three "ISAACs," namely ISAAC-1, ISAAC-2, and ISAAC-3. In a next row of the table of FIG. **7***a* are presented indications of the times at which the particular ISAAC instance on the particular row was last used. As shown, the instance ISAAC-1 was last used at time T-1. The instance ISAAC-2 was last used at time T-1. Finally, the instance ISAAC-3 was last used at time T-3. According to the particular heuristic approach presented at FIG. **7***a*, it is expected that ISAAC-2 will be chosen, assuming that T-1 is earlier in time.

FIG. **7***b* presents another such heuristic approach, permitting the automatic choice between alternative instances. In the approach shown, selection would be accomplished by a "top-of-the-list" selection process, not requiring any user interaction. The heuristic approach suggested in FIG. **7***c* is based upon the principle of most frequent use. The table presented suggests that ISAAC-3 again would be selected, based upon the highest number of uses over a particular period of time, in comparison with the usage level of the remaining instances, ISAAC-1 and ISAAC-2.

FIG. **8***a* illustrates details of the operation of step **123** of FIG. **3** dealing with the updating of information and linked information in smart fields. In the earlier example of FIG. **6***c* in which it was decided that Isaac Asimov was the desired ISAAC, the phone information in window **170** had not yet been entered. This information may be available and can be accessed according to the process of FIG. **8***a*. The process starts at **200** and immediately checks the data base for any linked smart fields as indicated at **202**. If there are applicable smart fields which contain the desired phone number information, this data is obtained from the corresponding linked field types as suggested at **203**. Then, as suggested at **206**, the data obtained is entered into the applicable smart field of the window **170** under operation. In the absence of there being any linked fields available, the data obtaining step of **204** is skipped and the data is entered manually, if available. Otherwise, the phone data field will remain vacant as to that particular data element. Operation of updating information and linked information in accordance with step **123** of FIG. **3** is completed with step **208** in FIG. **8***a*.

FIG. **8***b* illustrates the completion of a successful data updating operation performed according to the procedure of FIG. **8***a*. Speaker block **179** in FIG. **8***b* has been selected, to enable tone evocation of the phone number as for tone dialing in connection with a telephone call. Alternatively, a modem option, indicated at **181**, can be selected. A pull-down menu button entitled "options" at **183** can be produced

as a help menu. Further, a "call" activity can be undertaken by selecting "call" button **185** indicated on the face of window **170**. Window **170** can be closed simply by selecting the "x" block shown in window **170**. To the left of the name field is diamond icon **175'** which can be invoked to produce a pull-down menu of verbs (not shown) to permit the user to initiate further assistance operations.

FIG. **9***a* shows the process of checking whether a specific selection has been made of a particular object, as suggested at **130** in FIG. **3**. The process of checking for the selection of a particular object begins at **220** in FIG. **9***a* and is conducted at **222**. If no object has been explicitly selected, a query is made regarding the availability of any objects for selection which may have been available since the last time out. A time out can be defined as a paragraph, or as a particular set period of time. A paragraph can be denoted as simply the character expressed by pressing the ENTER key or the carriage return on a keyboard. If no object have been explicitly selected at **222** and no objects have been available since the last time out, then the assistance process continues in a step **228** as suggested at step **133** in FIG. **3** with the entry of objects into the assistant. If, however, a selection of an object has been made according to **222** or **226** in FIG. **9***a*, then the process can continue at **224** with transference of the assistance activity to entry of the specific object(s) into the assistant operation, as suggested at step **135** in FIG. **3**.

FIGS. **9***b*–**9***c* indicate graphically the performance of the selection query operation as expressed in FIG. **9***a*. In FIG. **9***b* the objects CALL ISAAC are indicated. In FIG. **9***c*, these objects are highlighted or blocked off to denote selection of the particular objects.

FIG. **9***d* shows the highlighted objects transferred to window **170**, to perform entry of the selected object(s) into the assistant. Such entry into the assistant function need not be accompanied by actual transferal into window **170** and may be transparently performed without direct user awareness. However, actual display of the entry into assistant operation of the face of the display is considered to be a useful and user friendly approach. The function CALL ISAAC is consequently performable by simply tapping the DO field at **185**.

In the event that no selection has been made as to objects to be entered into the assistant, object(s) can be directly entered into the assistant as suggested at step **133** in FIG. **3**. Such direct entry of object(s) into the assistant can be accomplished as suggested in FIGS. **10***a* and **10**B. Entry of the objects is directly into the call field of window **170** of FIG. **10***a*, by pen in handwriting for example. The input handwritten objects are duly recognized and converted to formal font form as re-expressed in the call field and as shown in FIG. **10**B.

FIG. **11***a* shows the recognition of object(s) process which is part of FIG. **3** at **135**, in order to enable recognition of possible user intent. The recognition process is started at step **240** in FIG. **11***a*. Next, a decision step **242** determines whether the object(s) match at least one template. If not, the process continues at a step **244** which corresponds to step **136** of FIG. **3**. If so, the process continues at step **137** of FIG. **3**. In substance, the process aims to determine whether the object(s) match at least one of the templates of object combinations set forth in FIG. **11***c*. FIG. **11***b* illustrates the object combination under operation, denoted by kind of object. The verb CALL is considered to be an action object and ISAAC is considered to be a person object. The two objects in combination are subject to template comparison. The template in FIG. **11***c* is effective for organizing in preset form the various object combinations which are capable of further operation as particular functions to be accomplished. FIG. **11***c* illustrates selected example functions such as scheduling, finding, filing, formatting, mailing, faxing, printing, and calling, just to cite a few of the possibilities.

FIG. **11***c* further provides example kinds of action objects, such as meet, find, file, format, mail, fax, print, and call. The Figure provides examples of allowable combinations of objects which correspond to the indicated functions and actions. For example, essential objects for scheduling a meeting include four objects, such as person, place, day, and time slot. Finding activities require the combination of two objects, which are for example a quantifier and an object. Filing requires a quantifier and notes. Formatting require notes and form, mailing requires a person, a place, and a letter. Faxing requires a person, a place, a fax number, and notes. Printing requires an object and a place. Calling requires a person, a place, and a phone number.

The recognition of possible user intent process called for at **135** in FIG. **3** and expressed in example form at FIG. **11***a*, calls for a matching operation between particular noted object(s) such as those illustrated in FIG. **11***b* and those expressed in the template of FIG. **11***c*. In this particular example, the intent is not obscure. The object <CALL> is clearly specified. There is a template match with the calling function expressed in the template. Both the action "call" and the person to be called are present in the template, permitting an effective, though not complete match. The place and the phone number are yet to be determined.

In many instances, intent will not so clearly be evident. For example, them may be multiple function matches based upon a particular combination of objects. Accordingly, it will be imperative to hypothesize the actual user intent based upon a selection of alternatives, as suggested at **137** in FIG. **3**. The process for hypothesizing user intent is made explicit in FIG. **12***a*. Once the hypothesizing process begins at step **260**, it continues with the enumeration of all possible intents, as indicated at step **262**. Then, prioritization is accomplished and a choice is made as to the best guess, as noted at step **264**. Finally, the process continues at step **266** with the FIG. **3** query at **139** whether in fact a hypothesis has been established as to whether a particular assistance activity is desired.

If no hypothesis has been produced, as determined by the answer to the query made at **139** of FIG. **3**, then the user may supply a proposed assistance course of action, as suggested at **141** of FIG. **3**. This is made more explicit in FIG. **12***b*. For example, a user proposed o course of action can be determined by the process beginning at step **270** of FIG. **12***b*. As a threshold step, it is asked whether the user wishes to enter a particular action, according to the step noted at step **272** of FIG. **12***b*. If there is no desire by the user to enter a particular course of action, operation returns to point A of FIG. **3**, and the cycle of inquiring whether an implicit assist is desired is made, according to **104** of FIG. **3**.

Alternatively, if the user does wish to provide or enter a particular action, the process can continue for example with the presentation of a particular list of applicable actions, as indicated at step **274** of FIG. **12***b*. This approach is graphically illustrated in FIG. **12***c*, which shows presentation of the list of actions being made as a pull-down menu **170"** partially superimposing over window **170**.

Next, the user may make a selection of the proposed list of actions, as set forth at step **276** of FIG. **12***b*. This may amount to having highlighted the call action verb in pull-down menu **170"**, to produce the image of FIG. **12***d*, in which the please field has the font formalism CALL ISAAC expressed explicitly in response to user selection. If the user

fails to select, control again shifts to point A of FIG. **3** with a check as to whether an implicit assist is desired. On the other hand, if the user has made a particular selection, the action is entered as another object at step **278** of FIG. **12***b*. The process then continues at **143** of FIG. **3**, as shown in FIG. **12***b*, which amounts to undertaking preparations for execution of the particular action called for.

Preparation for execution is expressed in the process of FIG. **13**, starting at step **290**. The process calls for example for the filling in of a plan template and the identification of any missing preconditions, as set forth at step **292** of FIG. **13**. Next, a step **293** resolves missing preconditions to the extent possible. As a next step, operation awaits additional preconditions to be fulfilled at step **294**. A loop may be taken to repeat the resolution of missing preconditions at step **294** for at least a predetermined number of times. If it is desired not to wait for completion of any additional preconditions, the process ends at step **295**, and action continues with the question as to whether to execute at **145** of FIG. **3**. This can involve a process as for example set forth in FIG. **14***a*. The process starts at **300** and then ascertains whether the "DO" button has been tapped or depressed at a step **302**. If yes, a confirmation slip or the like will be presented at a step **304**, and then operation continues at **306** with execution according to step **147** on FIG. **3**. A sample confirmation slip is indicated at FIG. **14***b* for convenience. If the decision is not to execute, then control passes to point A of FIG. **3**. If execution is desired, then step **147** of FIG. **3** is undertaken.

Accordingly, the invention herein provides implicit and explicit assistance for a variety of user-supportive information functions. If no implicit assistance actions are desired or indicated, an investigation is undertaken whether explicit assistance is to be undertaken. If no explicit assistance is desired, a check is again made whether an implicit assist is desired. An inquiry is made whether the user should be queried. If so, a list of action alternatives is displayed for user selection. If no user selection of particular assistance is desired, then a process is undertaken to select a best alternative of several identified. If explicit assistance is undertaken and assistance is desired by the user, particular object (s) expressing the assistance action desired are entered into an assistance operation. If user selection of an assistance alternative has been made, the particular selected object(s) are entered into assist operation. An attempt may then be made to recognize possible intents expressed by the objects entered into the assistance process. A check is further undertaken, to determine whether a hypothesis is in fact available. If no hypothesis is found, the process permits the user to supply a proposed action. If no hypothesis is found and no user action is proposed, assistance efforts terminate. However, if a hypothesis is available, preparations for execution are undertaken. A final inquiry is made as to whether to undertake the hypothesized assist. If the response to an inquiry whether to assist as hypothesized is affirmative, execution of the hypothesized action is undertaken.

While this invention has been described in terms of several preferred embodiments, there are alterations, permutations, and equivalents which fall within the scope of this invention. It should also be noted that there are may alternative ways of implementing the processes of the present invention. For example, much of the programming can be simplified by using the high-level utilities and data structures mentioned in the preceding specification. In particular, the described frame database system is preferred for simplifying the programming tasks required by the computer implemented processes of the present invention, but there are many other database and graphics systems which can be used to accomplish the same task. In general, the invention includes a method and apparatus for providing computer-assisted assistance, including the steps of noticing an event occurring within a computer system, determining whether the event from its context is suited for implicit assistance, and, if so, providing such implicit assistance, and determining whether explicit assistance is indicated, and, if indicated, providing such explicit assistance. The context includes such considerations, for example, as entry in a smart field window, and relationships between particular objects used in the assistance operation. The invention further includes a computer system having assistance capabilities and comprising computation means to perform a range of assistance functions, a memory in communication with the computation means to maintain a data base of assistance-pertinent events, a system for noticing events and occurrences which might require assistance, an arrangement for determining whether explicit and/or implicit assistance are indicated, and the necessary structures and functions effective for providing the needed or requested assistance.

It is therefore intended that the following appended claims be interpreted as including all such alterations, permutations, and equivalents as fall within the true spirit and scope of the present invention.

What is claimed is:

**1**. A computer system having assistance capabilities, comprising:

(a) computation means for performing assistance functions,

(b) memory means for maintaining a data base of assistance-pertinent events, said memory means being coupled to said computation means,

(c) means for providing a smart field responsive to information of a predefined type.

(d) means for noticing assistance-pertinent event occurrences subject to potential assist action performance, the means for noticing assistance-pertinent event occurrences including:

means for determining whether the information of a predefined type has been entered into the smart field; and

search means responsive to the entry of the information of a predefined type into the smart field, the search means operable for searching a portion of the data base of assistance-pertinent events associated with the type of information of the smart field to identify and compile a list of any alternatives matching or partially matching the specific information entered in the smart field,

(d) means for determining whether implicit assistance responsive to the event is indicated, and

(e) means for providing the implicit assistance indicated.

**2**. A computer system as recited in claim **1** wherein said means for providing the implicit assistance indicated includes means for automatically displaying an icon adjacent to the smart field, the icon indicating the existence of the compiled list.

**3**. A computer system as recited in claim **1** wherein said means for providing the implicit assistance indicated further includes:

means for determining when the icon has been selected; and

means for displaying the compiled list when the icon has been selected.

**4**. A computer system as recited in claim **1** wherein said means for providing the implicit assistance indicated includes means for automatically displaying the compiled list.

5. A computer system as recited in claim 1 wherein said means for providing the implicit assistance indicated includes:

means for displaying the compiled list; and

means for enabling a user of the computer system to select an alternative from the compiled list.

6. A computer system as recited in claim 5 further including means for updating the database to contain information regarding the selected alternative.

7. A computer system as recited in claim 1 wherein the nature of the smart field is indicated by text, graphics, or a combination of text and graphics.

8. A computer system as recited in claim 1 wherein the smart field is responsive to information selected from the group consisting of persons, telephone numbers, dates, document names, account numbers, addresses, and access codes.

9. A computer system as recited in claim 8 wherein the database is comprised of templates, each template comprised of linked fields, each linked field being selected from the group consisting of persons, telephone numbers, dates, document names, account numbers, addresses, and access codes.

10. A method of providing implicit assistance on a computer system, the method comprising the steps of:

maintaining a database of assistance-pertinent events on a memory of the computer system;

displaying on a display screen of the computer system a smart field responsive to information of a predefined type;

determining whether information of a predefined type has been entered into the smart field by a user of the computer system;

searching a portion of the data base of assistance-pertinent events associated with the type of information of the smart field to identify and compile a list of any alternatives matching or partially matching the specific information entered into the smart field;

determining whether implicit assistance responsive to the specific information entered in the smart field is indicated; and

providing the indicated implicit assistance.

11. A method as recited in claim 10 wherein the step of providing the indicated implicit assistance includes the substeps of:

automatically displaying an icon adjacent to the smart field, the icon indicating the existence of the compiled list;

determining whether the icon has been selected; and

displaying the compiled list when the icon has been selected.

12. A method as recited in claim 10 wherein the step of providing the indicated implicit assistance includes the substep of automatically displaying the compiled list.

13. A method as recited in claim 12 wherein the step of providing the indicated implicit assistance further includes the substeps of:

receiving a user selection from the compiled list; and

updating the database to contain information regarding the selected alternative.

14. A method as recited in claim 10 further including the step of displaying text, graphics or a combination of text and graphics in order to indicate the nature of the smart field.

15. A method as recited in claim 10 wherein the smart field is responsive to information selected from the group consisting of persons, telephone numbers, dates, document names, account numbers, addresses, and access codes.

16. A method as recited in claim 15 wherein the database is comprised of templates, each template comprised of linked fields, each linked field being selected from the group consisting of persons, telephone numbers, dates, document names, account numbers, addresses, and access codes.

17. A computer program stored on a computer readable medium, the computer program comprising computer executable instructions for:

maintaining a database of assistance-pertinent events on a memory of a computer system executing the computer program;

displaying on a display screen of the computer system a smart field responsive to information of a predefined type;

determining whether information of a predefined type has been entered into the smart field by a user of the computer system;

searching a portion of the data base of assistance-pertinent events associated with the type of information of the smart field to identify and compile a list of any alternatives matching or partially matching the specific information entered into the smart field;

determining whether implicit assistance responsive to the specific information entered in the smart field is indicated; and

providing the indicated implicit assistance.

18. A computer program as recited in claim 17 wherein the computer executable instruction for providing the indicated implicit assistance includes subinstructions for:

automatically displaying an icon adjacent to the smart field, the icon indicating the existence of the compiled list;

determining whether the icon has been selected; and

displaying the compiled list when the icon has been selected.

19. A computer program as recited in claim 18 wherein the computer executable instruction for providing the indicated implicit assistance further includes the subinstructions for:

receiving a user selection from the compiled list; and

updating the database to contain information regarding the selected alternative.

20. A computer program as recited in claim 17 wherein the smart field is responsive to information selected from the group consisting of persons, telephone numbers, dates, document names, account numbers, addresses, and access codes, and the database is comprised of templates, each template comprised of linked fields, each linked field being selected from the group consisting of persons, telephone numbers, dates, document names, account numbers, addresses, and access codes.

* * * * *

# Exhibit 13

# Intentionally Left Blank

# Exhibit 14

UNITED STATES DISTRICT COURT
DISTRICT OF DELAWARE

ARENDI S.A.R.L.
Plaintiff,
C.A. No. 12-1601-LPS
-against-
MOTOROLA MOBILITY LLC
f/k/a MOTOROLA MOBILITY, INC.
Defendant.

ARENDI S.A.R.L.
Plaintiff,
C.A. No. 12-01602-LPS
-against-
SONY MOBILE COMMUNICATIONS (USA)
INC., f/k/a SONY ERICSSON MOBILE
COMMUNICATIONS (USA) INC.,
SONY CORPORATION and SONY
CORPORATION OF AMERICA,
Defendant.

ARENDI S.A.R.L.
Plaintiff,
C.A. no. 13-919-LPS
-against-
GOOGLE LLC,
Defendant.

ARENDI S.A.R.L.
Plaintiff,
-against- C.A. No. 13-920-LPS
OATH HOLDINGS INC., and
OATH INC.,
Defendants.

October 29, 2019
9:22 a.m.

CONFIDENTIAL
OUTSIDE ATTORNEY EYES ONLY

(Caption continued on next page.)

Job No. CS3601311

----------x
ARENDI S.A.R.L.
Plaintiff,
-against-
C.A. No. 12-1595-LPS
LG ELECTRONICS, INC.
LG ELECTRONICS USA, INC. and
LG ELECTRONICS MOBILECOMM
U.S.A., INC.,
Defendants.
----------x
ARENDI S.A.R.L.
Plaintiff,
-against-
C.A. No. 12-1596-LPS
APPLE INC.
Defendant.
----------x
ARENDI S.A.R.L.
Plaintiff,
-against-
C.A. No. 12-1599-LPS
MICROSOFT MOBILE, INC.,
Defendant.
----------x

Videotaped Deposition of ATLE HEDLOY, taken by Defendants, pursuant to 30(b)(1) Notice, at the offices of Susman Godfrey, 1301 Avenue of the Americas, New York, New York, before William Visconti, a Shorthand Reporter and Notary Public within and for the State of New York.

A P P E A R A N C E S:

SUSMAN GODFREY LLP
Attorneys for Plaintiff
1000 Louisiana Street, Suite 5100
Houston, Texas 77002
BY: JOHN P. LAHAD, ESQ.
jlahad@susmangodfrey.com

PAUL HASTINGS LLP
Attorneys for Google & Motorola
71 S. Wacker Drive
Chicago, IL 60606

BY: ROBERT UNIKEL, ESQ.
robertunikel@paulhasings.com

VENABLE LLP
Attorneys for Sony and Oath
600 Massachusetts Avenue, NW
Washington, D.C. 20001
BY: JEFFRI A. KAMINSKY, ESQ.
jakiminsky@venable.com

FISH & RICHARDSON P.C.
Attorneys for LG Electronics
One Marina Park Drive
Boston, MA 02210-1878

BY: STEVEN R. KATZ, ESQ.
katz@fr.com
MATTHEW C. BERNTSEN, ESQ.
bernsten@fr.com (PM Session)

DLA PIPER LLP (US)
Attorneys for Apple
401 B Street, Suite 1700
San Diego, California 92101-4297

BY: ROBERT C. WILLIAMS, ESQ.
robert.williams@dlapiper.com

A P P E A R A N C E S (Continued)

COOLEY LLP
Attorneys for Microsoft
1299 Pennsylvania Avenue NW
Washington, DC 20004

BY: PHILLIP MORTON, ESQ.
pmorton@cooley.com

MC GUIRE WOODS LLP
Attorneys for Blackberry
2000 McKinney Avenue, Suite 1400
Dallas, TX 75201
BY: SHAUN W. HASSETT, ESQ.
shassett@mcguirewoods.com
(Via Phone)

ALSO PRESENT:

HOWARD BRODSKY, Videographer

THE VIDEOGRAPHER: Good morning. Here begins the video recorded testimony of Atle Hedloy taken by the Defendants in the matters of Arendi S.A.R.L., Plaintiff versus Google LLC, et al., Defendants, Civil Action Numbers 13-919-LPS and 12-1601-LPS, et al., in the United States District Court for the District of Delaware. This deposition is proceeding at Susman Godfrey, 1301 Avenue Of The Americas, New York, New York 10019 on Tuesday, October 29th, 2019 at approximately 9:22.

My name is Howard Brodsky and I'm the legal video specialist in association with Veritext Corporate Services with offices located in New York, New York. The court reporter is William Visconti in association with Veritext. The court reporter is entering all appearances for this proceeding into the court record and will the court reporter please swear in the witness.

A T L E H E D L O Y,

having been first duly sworn by the Notary Public, was examined and testified as follows:

EXAMINATION CONDUCTED BY MR. UNIKEL:

Q. Good morning, sir.

A. Good morning.

Q. My name is Rob Unikel and I'm an attorney representing Google and Motorola Mobility. Do you understand that?

A. I understand that.

Q. Today I will be asking questions on behalf all the defendants in this case that Arendi has filed, do you understand that?

A. Yes.

Q. You have been deposed many times before; is that correct?

A. At least some, yes.

Q. And you have given trial testimony in the past as well?

A. Yes.

Q. So am I correct that you have an understanding starting off today as to how a deposition works?

A. Yes.

Q. Just to be clear, I will be asking

Do you see that?

A. I see that.

Q. Why is it important to your invention that the first information be analyzed to determine if it is one of a plurality of types that can be searched for?

A. It is part of the invention, it needs to be described.

Q. Does it have any advantage to including that element?

MR. LAHAD: Objection to form.

A. It correctly describes what is we are claiming.

Q. Why not just specify that the information needs to be identified as a particular piece of contact information like a phone number or an address?

A. This claim doesn't require it to be a phone number or address. It just says it needs to be of a particular type.

Q. When it says it -- it doesn't just say it has to be of a particular type, right, it says it has to be of a particular type that can be searched for in order to find second information. Why include that requirement?

A. Because it will be searched later in the claim.

Q. Is it enough for someone to infringe this element that they simply identified a piece of information as a cell phone number without making a determination that it could be searched for?

MR. LAHAD: Objection to form.

A. You need to look at the claim in totality. You need to infringe -- what was the question about infringement?

Q. You have to infringe every element, correct?

A. Yes.

Q. So I'm looking at this element, this element you believe is an important part of your claimed invention?

MR. LAHAD: Objection to form.

A. It is one of the elements of the claim.

Q. And so my question for you, is it enough to practice that element if while the document is being displayed first information is identified as a phone number but no determination is made as to whether or not that

if the first information is at least one of a plurality of types of information that can be searched for in order to find second information related to the first information."

Q. We are both reading the same. That's my question is if I just make a determination that a piece of information is a phone number, but I do not make a determination that the phone number can be searched for, am I practicing this element or not, sir?

A. You need to practice the whole element so that's what you're doing, then if you practice the whole element, then it infringes -- then you practice the element.

Q. I understand. My question is, if I identify a piece of information as being a telephone number, but I do not determine whether or not that telephone number can be searched for, have I practiced this element?

MR. LAHAD: Objection to form.

A. You practice this element if you determine first information is at least one of a plurality of types of information that can be searched for.

Q. So if I don't make that

determination, then I'm not practicing this element?

A. I think you need to do what it says here to practice the element.

MR. LAHAD: Objection to form.

Q. That includes making the determination of whether it can be searched for?

A. It includes "to determine the first information is at least one of a plurality of types of information that can be searched for in order to find second information related to the first information."

Q. Is there any advantage to requiring that the information must be determined to be a type that can be searched for?

A. You're going to use it for a search later so if you can't use it to be searched for then you can't practice the rest of the claim.

Q. So you feel that is an important part of the claim?

MR. LAHAD: Objection to form.

A. It's an important -- the whole

invention?

A. Several advantages. One is that you can work in one program, search information from one program while you're simultaneously working in another.

Q. Are there any other advantages?

A. Yes.

Q. List them all for me and try not to leave anyone out.

A. Some of them are mentioned at the beginning of the patent here. The user can access -- can perform these tasks with a minimal number of actions or key strokes. Does not have to learn or even have access to directly the database or the second application of database where the data is stored. There are a number of advantages.

Q. We have you can work in one program while simultaneously continuing to work on the document. Is that correct, that was one advantage?

A. Yes, that is one advantage.

Q. Another advantage is the user can access these functions with a minimal number actions; is that correct?

least part of the second information includes simply displaying the second information on the computer screen?

A. Okay.

Q. Is that correct, am I understanding you correctly?

A. You're understanding me correctly.

Q. The next element you say, "In consequence by receipt of the first computer program from the user command from the input device causing a search for the search term in the information source using a second computer program in order to find second information related to the search term." Do you see that?

A. I see that.

Q. Why is it important that the search be performed in consequence of receipt by the first computer program of the user command?

MR. LAHAD: Objection to form.

A. That is the action that we wanted to claim here.

Q. Why does the first computer program need to receive the command as opposed to the second computer program receiving the

command?

A. The invention that we want to claim here is this one where the first computer program receives it.

Q. Why did you want to capture that idea in this claim that the first computer program received the command from the input device?

A. Because that is the invention that we want to claim.

Q. Is there any advantage to having the first computer program be the program that receives the user command from the input device?

A. There could be.

Q. What advantages did you intend by including that particular element?

A. We included that particular element because that is the invention that we wanted to claim.

Q. It's a bit circular. I assume that you had a reason for wanting that to be part of the invention that you wanted to claim, is that accurate?

A. Presumably yes.

Q. Once again, I assume that you

where you can type text into it as we discussed about.

Q. If you can type text into it then you believe it becomes a word processing document?

A. It may become a word processing document as a result.

Q. That is my question. How do I know when it does and when it doesn't?

A. First of all word processing is defined in this specification to mean all kinds of programs. It is just a computer program and we give some examples of that and these are examples of those.

Q. So now I'm asking you about webpages in particular and how I will determine when they are word processing documents and when they are not.

A. Like I said, if it is a word processing -- if it consists of a field that you can fill in. For example, a search field would not be a word processing document.

Q. Why not?

A. Because you can't -- it's not a document. It is just a search field.

# Exhibit 15

UNITED STATES DISTRICT COURT
DISTRICT OF DELAWARE

ARENDI S.A.R.L.
Plaintiff,
C.A. No. 12-1601-LPS
-against-
MOTOROLA MOBILITY LLC
f/k/a MOTOROLA MOBILITY, INC.
Defendant.

ARENDI S.A.R.L.
Plaintiff,
C.A. No. 12-01602-LPS
-against-
SONY MOBILE COMMUNICATIONS (USA) INC., f/k/a SONY ERICSSON MOBILE COMMUNICATIONS (USA) INC., SONY CORPORATION and SONY CORPORATION OF AMERICA,
Defendant.

ARENDI S.A.R.L.
Plaintiff,
C.A. no. 13-919-LPS
-against-
GOOGLE LLC,
Defendant.

ARENDI S.A.R.L.
Plaintiff,
-against-
C.A. No. 13-920-LPS
OATH HOLDINGS INC., and OATH INC.,
Defendants.

November 5, 2019
9:01 a.m.

(Caption continued on next page.)

Job No. CS3601408

----------x
ARENDI S.A.R.L.
Plaintiff,
-against-
C.A. No. 12-1595-LPS
LG ELECTRONICS, INC.
LG ELECTRONICS USA, INC. and
LG ELECTRONICS MOBILECOMM
U.S.A., INC.,
Defendants.
----------x
ARENDI S.A.R.L.
Plaintiff,
-against-
C.A. No. 12-1596-LPS
APPLE INC.
Defendant.
----------x
ARENDI S.A.R.L.
Plaintiff,
-against-
C.A. No. 12-1599-LPS
MICROSOFT MOBILE, INC.,
Defendant.
----------x

Videotaped Deposition of ATLE HEDLOY, taken by Defendants, pursuant to 30(b)(6) Notice, at the offices of Susman Godfrey, 1301 Avenue of the Americas, New York, New York, before William Visconti, a Shorthand Reporter and Notary Public within and for the State of New York.

A P P E A R A N C E S:

SUSMAN GODFREY LLP
Attorneys for Plaintiff
1000 Louisiana Street, Suite 5100
Houston, Texas 77002
BY: JOHN P. LAHAD, ESQ.
jlahad@susmangodfrey.com

PAUL HASTINGS LLP
Attorneys for Google & Motorola
71 S. Wacker Drive
Chicago, IL 60606

BY: ROBERT UNIKEL, ESQ.
robertunikel@paulhasings.com

VENABLE LLP
Attorneys for Sony and Oath
600 Massachusetts Avenue, NW
Washington, D.C. 20001
BY: JEFFRI A. KAMINSKY, ESQ.
jakiminsky@venable.com

FISH & RICHARDSON P.C.
Attorneys for LG Electronics
One Marina Park Drive
Boston, MA 02210-1878

BY: MATTHEW C. BERNTSEN, ESQ.
bernsten@fr.com (PM Session)
DLA PIPER LLP (US)
Attorneys for Apple
401 B Street, Suite 1700
San Diego, California 92101-4297

BY: CHRISTINE K. CORBETT, ESQ.
christine.corbett@dlapiper.com
ROBERT C. WILLIAMS, ESQ.
robert.williams@dlapiper.com

A P P E A R A N C E S (Continued)

COOLEY LLP
Attorneys for Microsoft
1299 Pennsylvania Avenue NW
Washington, DC 20004

BY: PHILLIP MORTON, ESQ.
pmorton@cooley.com

MC GUIRE WOODS LLP
Attorneys for Blackberry
2000 McKinney Avenue, Suite 1400
Dallas, TX 75201
BY: SHAUN W. HASSETT, ESQ.
shassett@mcguirewoods.com

ALSO PRESENT:

HOWARD BRODSKY, Videographer

THE VIDEOGRAPHER: Good morning. Here begins the video recorded 30(b)(6) testimony of Atle Hedloy appearing on behalf of Arendi S.A.R.L. taken by the Defendants in the matter of Arendi S.A.R.L., Plaintiff versus Google LLC, et al, Defendants. Civil action numbers 13919 LPS, et al. in the United States District Court for the District Court of Delaware.

This deposition is proceeding at Susman Godfrey LLP, 1301 Avenue of the Americas, suite 3200, New York, New York 10019 on Tuesday, November 5th, 2019 at approximately 9:01. My name is Howard Brodsky and I'm the legal video specialist in association with Veritext Corporate Services with offices located in New York, New York. The court reporter is William Visconti in association with Veritext. The court reporter has recorded all appearances for this proceeding and will the court reporter please swear in the witness.

A T L E H E D L O Y,

having been first duly sworn by the Notary Public, was examined and testified as follows:

EXAMINATION CONDUCTED BY MS. CORBETT:

Q. Good morning, Mr. Hedloy.

A. Good morning.

Q. My name Christine Corbett and I'm here on behalf of Apple?

A. Okay.

Q. You have been deposed many times where your most recent deposition taking place last week; is that correct?

A. That's correct.

Q. So I assume that you understand how the deposition process works, but let me cover some of the basic ground rules.

A. Okay.

Q. You understand that you are under oath with an obligation to tell the truth; correct?

A. Yes.

Q. And if you don't understand my question, please let me know otherwise I will assume that you understand my question, okay?

A. Okay.

Q. Do you see there is a screen shot there of the Everybody.net search contact information screen?

A. I see that.

Q. Again, do you consider what is shown in that screen shot to be a document?

A. I don't think I do.

Q. Why not?

A. Again, it is just showing you a form and fields, not a document.

Q. But I could enter text into those fields, couldn't I?

MR. LAHAD: Objection to form.

A. I don't know exactly, but possibly, yes.

Q. It is search contact information screen shot, correct?

A. That's what it looks like, it seems like you could.

Q. Does the fact that I could enter text into those various fields alter your opinion at all as to whether you view this as a document?

A. I don't think it does.

Q. Why not?

A. These are just fields, if you type something in you cannot type in free text. You can only type in predefined text.

(Hedloy Exhibit 73 for identification, Document Bates number AHL 00067169.)

Q. Let me show you a document that I marked as Exhibit 73. On this one I'm going to need your translation services.

A. Okay. I don't charge too much for that. Just a little bit.

Q. Sir, do you see that Exhibit 73 has on it an e-mail from you to somebody named Torjeir.Jacobsen?

A. I think that's the beginning of his e-mail address. His name doesn't include the dot.

Q. How do you pronounce?

A. Torjeir Jacobsen.

Q. Who is that individual?

A. A Norweigian person I know.

Q. Is he a friend, is he a customer, is he --

A. He is a friend.

Q. This particular e-mail was sent on

# Exhibit 16

UNITED STATES DISTRICT COURT
DISTRICT OF DELAWARE
----------x
ARENDI S.A.R.L.
Plaintiff,
C.A. No. 12-1601-LPS
-against-
MOTOROLA MOBILITY LLC
f/k/a MOTOROLA MOBILITY, INC.
Defendant.
----------x
ARENDI S.A.R.L.
Plaintiff,
C.A. No. 12-01602-LPS
-against-
SONY MOBILE COMMUNICATIONS (USA)
INC., f/k/a SONY ERICSSON MOBILE
COMMUNICATIONS (USA) INC.,
SONY CORPORATION and SONY
CORPORATION OF AMERICA,
Defendant.
----------x
ARENDI S.A.R.L.
Plaintiff,
C.A. no. 13-919-LPS
-against-
GOOGLE LLC,
Defendant.
----------x
ARENDI S.A.R.L.
Plaintiff,
-against- C.A. No. 13-920-LPS
OATH HOLDINGS INC., and
OATH INC.,
Defendants.
----------x
November 6, 2019
9:04 a.m.

(Caption continued on next page.)

Job No. CS3601420

----------x
ARENDI S.A.R.L.
Plaintiff,
-against-
C.A. No. 12-1595-LPS
LG ELECTRONICS, INC.
LG ELECTRONICS USA, INC. and
LG ELECTRONICS MOBILECOMM
U.S.A., INC.,
Defendants.
----------x
ARENDI S.A.R.L.
Plaintiff,
-against-
C.A. No. 12-1596-LPS
APPLE INC.
Defendant.
----------x
ARENDI S.A.R.L.
Plaintiff,
-against-
C.A. No. 12-1599-LPS
MICROSOFT MOBILE, INC.,
Defendant.
----------x

Continued Videotaped Deposition of ATLE HEDLOY, taken by Defendants, pursuant to 30(b)(6) Notice, at the offices of Susman Godfrey, 1301 Avenue of the Americas, New York, New York, before William Visconti, a Shorthand Reporter and Notary Public within and for the State of New York.

A P P E A R A N C E S:

SUSMAN GODFREY LLP
Attorneys for Plaintiff
1000 Louisiana Street, Suite 5100
Houston, Texas 77002
BY: BEATRICE KATHERINE FRANKLIN, ESQ.
bfranklin@susmangodfrey.com

PAUL HASTINGS LLP
Attorneys for Google & Motorola
71 S. Wacker Drive
Chicago, IL 60606

BY: ROBERT UNIKEL, ESQ.
robertunikel@paulhasings.com

VENABLE LLP
Attorneys for Sony and Oath
600 Massachusetts Avenue, NW
Washington, D.C. 20001
BY: JEFFRI A. KAMINSKI, ESQ.
jakaminski@venable.com

FISH & RICHARDSON P.C.
Attorneys for LG Electronics
One Marina Park Drive
Boston, MA 02210-1878

BY: STEVEN R. KATZ, ESQ.
bernsten@fr.com (PM Session)
JACOB B. PECHT, ESQ.
pecht@fr.com

DLA PIPER LLP (US)
Attorneys for Apple
401 B Street, Suite 1700
San Diego, California 92101-4297

BY: CHRISTINE K. CORBETT, ESQ.
christine.corbett@dlapiper.com
ROBERT C. WILLIAMS, ESQ.
robert.williams@dlapiper.com

A P P E A R A N C E S (Continued)

COOLEY LLP
Attorneys for Microsoft
1299 Pennsylvania Avenue NW
Washington, DC 20004

BY: PHILLIP MORTON, ESQ.
pmorton@cooley.com

MC GUIRE WOODS LLP
Attorneys for Blackberry
2000 McKinney Avenue, Suite 1400
Dallas, TX 75201
BY: SHAUN W. HASSETT, ESQ.
shassett@mcguirewoods.com

ALSO PRESENT:

HOWARD BRODSKY, Videographer

THE VIDEOGRAPHER: Good morning. Here begins volume 2 of the video recorded 30(b)(6) testimony of Atle Hedloy appearing on behalf of Arendi S.A.R.L. taken by the Defendants in the matter of Arendi S.A.R.L. Plaintiff versus Google LLC et al. Defendants, civil action numbers 13919 LPS, et al. in the United States District Court for the District of Delaware.

Today is Wednesday, November 6, 2019 and the time is approximately 9:04. The location remains the same as in volume 1 of this proceeding. And all appearances have been recorded by the court reporter, William Visconti. The witness has been sworn to be truthful on the record.

A T L E H E D L O Y,

having been previously duly sworn by the Notary Public, was examined and testified as follows:

EXAMINATION CONDUCTED BY MR. UNIKEL:

Q. Good morning Mr. Hedloy.

A. Good morning.

Q. You understand that you're still

you would not have included it in the claim element?

MS. FRANKLIN: Objection to form.

A. It is important because it is included.

Q. Am I correct then that if the analyzing identifies first information that cannot be searched for in order to find second information, that would not be information that satisfies this element of the claim?

MS. FRANKLIN: Objection to form.

A. The way that I understood you, I think that is correct.

Q. So for example, let's assume that the first information is a phone number?

A. Okay.

Q. The phone number can be first information under this claim element, can't it?

A. Yes.

Q. If while the document was displayed there was analyzing of the phone number to determine that it was in fact a phone number. If it was also determined that the phone number could not be searched for in a source external to the document, this claim

element would not be met, correct?

MS. FRANKLIN: Objection to form.

A. I don't understand how it is possible to determine that the phone number cannot be searched for.

Q. Why not?

A. I don't understand it because I don't understand it. We can search for phone numbers if have a database of phone numbers, of course?

Q. How does that determination get made in accordance with this element? Let me ask a different question.

How can someone who is practicing a method determine whether or not information is information that be can searched for or not in order to find second information?

MS. FRANKLIN: Objection to form.

A. I think it is quite easy. The person who wants to implement something using this will know what he is looking for and that's what the type of stuff he will find. So if he wants to look up an address, based on a name he knows that he needs to look for a name to identify a name in the top part.

Q. As an example you just gave me, if a user did not have a contact database on their computer then they might not be able to search for a phone number, right?

A. That's correct.

Q. So, it is possible that you could have information that can be searched for in a source external to the document or that could not be searched for in a source external to the document, it is possible; correct?

A. In theory, I suppose.

Q. In this claim it is required not that you identify first information, correct? It doesn't simply say identify first information that is one of a plurality of types.

A. The words are -- it doesn't say identify at all, it says, determine if the first information is at least a plurality of types of information that can be searched for in order to find second information related to the first information.

Q. Correct, and I'm just -- we are just establishing that this element doesn't stop after plurality of types of information,

correct? It doesn't simply say, analyzing in a computer process first information from the document to determine if the first information is at least one of a plurality of types of information.

A. There are more words after the word information there?

Q. Correct.

A. That is correct.

Q. So if a user identifies a piece of information as a phone number, but does not make any determination whether this phone number can be searched for in a source external to the document, that last element would not be met, correct?

MS. FRANKLIN: Objection to form.

A. I don't think a user is doing this at all. It is the computer process is doing this.

Q. So if the computer process that is doing this identifies information as a phone number but does not in fact make a determination whether that phone number "Can be searched for in order to find second information related to the first information" this element would not

be practiced, correct?

MS. FRANKLIN: Objection to form.

A. I think I disagree.

Q. Why?

A. I just read the words here. The words say, "this computer process must determine if the first information is at least one of a plurality of types of information that can be searched for in order to find second information related to first information." So if it does that, then that is sufficient.

Q. That doesn't really answer the question because that just says if you do it you do it. What I'm asking you is, it's possible to identify a phone number without making a determination whether or not you can search for that phone number or not, correct?

MS. FRANKLIN: Objection to form.

A. That's an abstract hypothetical, so in the vacuum that is true.

Q. So again, if a computer process was to identify a phone number as a phone number but make no determination whatsoever whether that phone number could be searched for in order to find second information related to

the first information, then that language of this claim element would not be present, would it?

MS. FRANKLIN: Objection to form.

A. I disagree.

Q. Again, why, I would like to understand?

A. Okay. I think the words say what they say and I don't think they mean what you say.

Q. I'm asking you why as the inventor. As you told me these words were included because they are important and that's why they were included, correct?

A. Yes. And as the inventor I said we use those words because we thought that was the best way to say it.

Q. So, because those words are important, if a process, a computer process analyzed first information from a document and determines that first information is a phone number, we have now met all of the claim words here up to "that can be searched for in order to find second information." Would you agree with that?

A. I think that the determination that it's a phone number might -- is probably a determination if it is one, at least one of a plurality of types of information that can be searched for in order to find second information related to first information. I think you have done it when you find the phone number.

Q. You already said to me that you agree that it might be possible that the phone number can't be searched for if, for example, I didn't have any contact database on my device; correct?

A. It just makes no sense to me what you're saying. What you're saying makes no sense.

Q. Okay, let's break it down because I want to understand. Do you agree with me that a phone number can be searched for, for example, if you have a contact database on your device?

A. If you have a contact database on your device and you're able to search on that contact database for a phone number then you can search for a phone number, yes.

Q. But it is possible that you could have a contact database that doesn't let you search by phone number, right?

A. It is possible in a vacuum, yes.

Q. And it is possible that you might not have a contact database on your device, right

A. That is also in a vacuum possible.

Q. It is possible that you can identify information in a document as a phone number but not be able to search for it to find second information in a source external to the document; correct?

MS. FRANKLIN: Objection to form.

A. In theory you're looking at one element, in theory it is possible to find that situation. But it is not the claim.

Q. You're right. The claim does in fact include the words that there must also be a determination that the type of information can be searched for in order to find second information related to the first information?

A. The words don't say that. There are no words that say there must be a determination that it can be searched for. The

words say, to determine if first information is at least one of a plurality of types of information that can be searched for in order to find second information related to the first information. That the words not the words that you said.

Q. Again, are you saying that as long as the analyzing of the first information identifies the information as one of a plurality of types, such as a phone number or an address, that automatically the element that can be searched for in order to find second information related to the first information is necessarily satisfied?

A. I think you need to look at the claim in totality. You're going to do a search afterwards, there is so no reason to identify anything on top unless you're going to use it for a search. It makes no sense what you're saying to find the type of things that you can't search for.

Q. Sir, you have asked me to look at language of the claims which I'm doing.

A. And I'm answering you also looking at the language of the claim.

Q. I'm going to ask one more time and then we will start going in circles.

A. Okay.

Q. You agree with me it is possible to identify first information as a phone number, correct?

A. Yes.

Q. It is also possible that a phone number might not be information that can be searched on a particular device in a source external to the document?

A. In vacuum that is true. If that device doesn't have a database or information about phone numbers, that's true.

Q. So just analyzing first information in a document and identifying it as a phone number does not automatically tell that computer process whether in fact the phone number can be searched for in a source external to the document to find second information, correct?

A. In a vacuum that it is true that if you just find a phone number, that doesn't tell you whether you can search for it or not.

Q. This element of the claim that can

be searched for in order to find second information related to the first information, you are not suggesting that that language is unimportant in the claim, are you?

MS. FRANKLIN: Objection to form.

A. I think by definition all the words of the claim are supposed to be important.

Q. Find the '993 patent which was Exhibit 7.

A. Okay.

Q. Find column 13, claim one, please.

A. Okay.

Q. I told you last week when we were together I was going to ask you again this question so hopefully we are on the same page this time.

A. Okay.

Q. Do you see that the last element -- there is in the middle of the claim a list of three potential actions, correct?

A. I see that.

Q. Potential action one is initiating an electronic search in the contact database?

A. Etc., yes.

file?

MS. FRANKLIN: Objection to form.

A. Unfortunately I'm not sure it matters what I think, it matter what the law says and I'm not sure what the law says.

Q. What do you think?

MS. FRANKLIN: Objection to form.

A. I think certainly that a webpage can be a document.

Q. What would you have to -- what about a webpage makes it be a document in your view?

A. Certainly a webpage can contain -- you can even do -- these days you can do a Microsoft Word for Web I think it is called on webpage in a browser, so that would certainly count. If webpage only has a field then it would not be a document. So there is some range in that scope.

Q. Just so I understand when you say a webpage if it only contains a field, what does that mean?

A. I don't think the Google search bar for example is a document.

Q. So, if a webpage did not -- had

only a field in it that you could enter text into that field, you don't believe that that would be a document?

MS. FRANKLIN: Objection to form.

A. There is a scale here, I don't know where to put the line.

Q. If I was somebody reading the patent, how would I know where to put the line between something that is a document and isn't a document?

MS. FRANKLIN: Objection to form.

A. I think you have to ask the judge who made the decision.

Q. I notice the next paragraph in '993 column 12 around line 14 you say "Although the present invention is defined in terms of information management or database programs such as Outlook, etc., the present invention is applicable to all types of information management or database programs such as Access, Oracle, D Base, R Base, Card File, including Flat Files, etc., as will be readily apparent to those skilled in the art." Do you see that?

A. I see those words.

Q. Again, you have included Outlook

# Exhibit 17

# Exhibit 18

update a data source) to the known method of Miller to yield a predictable result.

(*Id.*)

| | |
|---|---|
| 10. A method according to claim 1, wherein receipt by the first computer program of the user command precedes analyzing the document. | The user's selection of the "detect structures" button 520 precedes analyzing the document. *See e.g.*, 5:22-31 ("Window 510 includes a button 520 for initiating program 165… Upon initiation of program 165, system 100 transmits the contents of document 210 to analyzer server 220, which parses the contents based on grammars 410 and strings 420 (FIG. 4)."). *See also* claim 1e. |

| | |
|---|---|
| 11. A method according to claim 1, wherein analyzing the document is completed after the receipt of the user command is completed and before searching is initiated. | *See* claim 10. |

| | |
|---|---|
| 12. A method according to claim 1, wherein the input device is a graphical input device. | Button 520 is a graphical input device. *See* claim 1e. |

| | |
|---|---|
| 13. A method according to claim 1, wherein the user command is the only command from a user necessary to initiate performing the operation. | There is no antecedent for "performing the operation." Selection of button 520 is the only command necessary to initiate, *i.e.*, begin, the operation. *See* claims 1e and 1h. |

| | |
|---|---|
| 14. A method according to claim 1, wherein the input device is a menu and the entry of the user command includes a user's selection of the menu and click on a menu choice from the menu. | Miller discloses selection of the "detect structures" button. (5:22-31.) It also discloses a pop-up user menu. (*See* Fig. 7; 4:23-31; 5:38-40.) *See* narrative below. |

It was common knowledge that commands can be selected via a button or via a pop-up menu (which are both disclosed in Miller). (Menascé Decl. ¶ 77.) Thus, it would have been obvious to a person of ordinary skill in the art that selection of the "detect structures" command could be made from a conventional menu, as a matter of user interface design. Providing such functionality would have been a predictable modification of Miller well within ordinary skill. (*Id.*)

| | |
|---|---|
| 15. A method according to claim 1, further comprising, if searching results in a plurality of distinct instances of second information, displaying such instances to enable user selection of one of them for use in performing the action. | Fig. 4 shows an action for calling a person with the identified name. *See* narrative below. |

It would have been common knowledge that an address book would contain information such as plural email addresses (*e.g.*, work and personal), physical addresses (*e.g.*, home and/or work), telephone numbers (*e.g.*, home, work, or mobile), etc. (Menascé Decl. ¶ 78.) If, for example, several telephone numbers were associated with a name (*i.e.*, "plurality of distinct instances of second information"), it would have been obvious to display the plural phone numbers to enable to the user to select one to call (*i.e.*, "displaying such instances to enable user selection of one of them for use in performing the action."). This would have been simply a matter of common sense and common knowledge and there would

have been design and market incentives to provide such functionality. (*Id.*) One of ordinary skill would have been able to apply a known technique (displaying for selection plural results of a search) to the known method of Miller to yield a predictable result. (*Id.*)

| | |
|---|---|
| 16. A method according to claim 1, wherein the information source is associated with the second computer program and is available on the computer. | The information source, such as an address book, is associated with program 165 and is available on and through the computer. *See, e.g.*, 5:6-17, 44-50 ("Upon selection of the action for putting the number in an electronic telephone book, user interface 240 transmits the corresponding telephone number and selected action to action processor 250. Action processor 250 locates and opens the electronic telephone book, places the telephone number in the appropriate field and allows the user to input any additional information into the file."). |

| | |
|---|---|
| 17. A method according to claim 1, wherein the information source is associated with the second computer program and is available through the computer. | *See* claim 16. |

| | |
|---|---|
| 18. A method according to claim 1, wherein performing the action includes causing insertion of at least part of the second information into the document. | *See* claim 5. |

| | |
|---|---|
| 19. A method according to claim 1, wherein performing the action includes causing insertion of at least part of the second information into the document by the first computer program. | *See* claims 5 and 7. |

| | |
|---|---|
| [20a] 20. A computer-implemented method for finding data related to the contents of a document using a first computer program running on a computer, the method comprising: | *See* claim 1a. |
| [20b] displaying the document electronically using the first computer program; | *See* claim 1b. |
| [20c] while the document is being displayed, analyzing, in a | *See* claim 1c. |

| | |
|---|---|
| computer process on the computer, first information from the document to determine if the first information is at least one of a plurality of types of information that can be searched for in order to find second information related to the first information, and | |
| [20d] wherein the first information comprises at least one of name-, person-, company-, and address-related information; | *See* claim 2. |
| [20e] providing an input device, configured by the first computer program, that allows a user to enter a user command to initiate an operation, | *See* claim 1e. |
| [20f] the operation comprising (i) performing a search using at least part of the first information as a search term in order to find the second information, of a specific type or types, associated with the search term in a user editable information source outside the document, wherein the specific type or types of second information is dependent at least in part on the type or types of the first information, and | *See* claim 1f. The user can add information to the phone book. Thus it is user editable. *See* claim 8. |
| [20g] (ii) performing an action using at least part of the second information, | *See* claim 1g. |
| [20h] wherein the input device includes a menu; | *See* claim 14. |
| [20i] retrieving the first information; | *See* claim 1d. |
| [20j] in consequence of receipt by the first computer program of the user command, such user command including a user's selection of the menu and click on a menu choice from the menu, | *See* claim 14. |
| [20k] causing a search for the search term in the user editable information source, using a second computer program, in order to find second information related to the search term in the user editable information source; and | *See* claims 1f and 1h. |
| [20l] if searching finds any second information related to the search term, performing the action using at least part of the second information, | *See* claim 1i. |
| [20m] wherein the action is of a type depending at least in part on the type or types of the first information and | *See* claim 1k. |
| [20n] performing the action includes at least causing display of at least part of the second information. | *See* claim 6. |

| | |
|---|---|
| 21. A method according to claim 20, further comprising, if | *See* claim 15. |

| | |
|---|---|
| searching results in a plurality of occurrences of second information, causing display of such instances to enable user selection of one of them for use in performing the action. | |

| | |
|---|---|
| 22. A method according to claim 20, wherein performing the action includes causing addition of at least part of the second information to the first information in the document. | *See* claim 5. |

### C. Computer Readable Medium Claims

Computer readable medium claims 23-44 would have been obvious in view of Miller. These claims correspond to method claims 1-22. Miller discloses or renders obvious the steps in the body of the computer readable medium claims (as set forth above with respect to the corresponding method claims) and further discloses a computer readable medium including program instructions (*see, e.g.*, Fig. 1 at 170).

## VIII. GROUND 3: OBVIOUSNESS OF CLAIMS 1-7, 10-29, AND 32-44 IN VIEW OF LUCIW

### A. Background Of Luciw

Luciw was filed on April 19, 1995 and thus qualifies as prior art under § 102(e) based on the earliest alleged U.S. filing date of the '843 patent. Luciw relates to Apple's pen-based, handheld Newton device developed in the 1990s. It discloses providing user assistance based on information entered into a document, such as a note area 54a, 54b displayed by a notepad application, as shown in Fig. 2 below. (2:19-22; 6:24-59.)

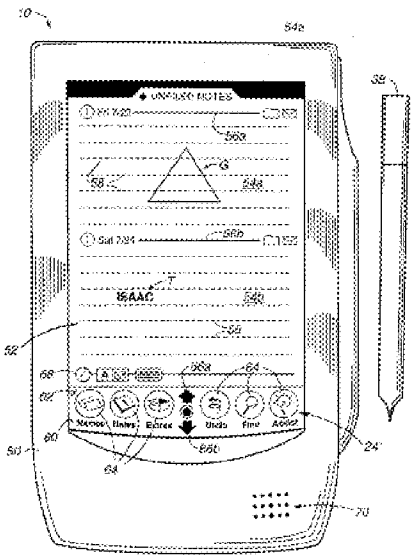


Figure 2

When the user selects the "explicit assist" button 64 in Fig. 2, the document is analyzed to determine what type of assistance, if any, is possible given the user's entry. (9:22-10:5; 13:52-14:4.) For example, if the user enters a first name, such as "Isaac," Luciw then searches a database and presents for user selection a list of persons with the full name identified as shown in Fig. 6b below. (11:60-12:6.)

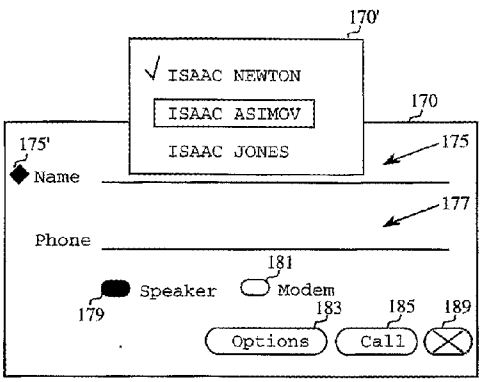



Figure 6b

When the user makes a selection, information associated with the person, such as the person's full name, is inserted into the document. (*Id.*; 12:41-54.)

It must be emphasized that Luciw's disclosure is not limited to Fig. 6b. The figure provides just one example of the identifications and corresponding actions available as understood by one of ordinary skill in the art as discussed below and in the accompanying declaration of Dr. Menascé (Ex. 1002).

### B. Method Claims

Set forth below is a claim chart that specifies where each element of method claims 1-7 and 10-22 is met by Luciw. Any narrative discussion with respect to obviousness for a given claim or claim element is provided directly under that claim or claim element with double line spacing.

| Claim | Luciw |
|---|---|
| [1a] 1. A computer-implemented method for finding data related to the contents of a document using a first computer program running on a computer, the method comprising: | Luciw discloses a computer-implemented assistance method that finds data related to the contents of a document (*e.g.*, finding contact information for a person with the name entered in the document) using a computer program running on a computer. *See, e.g.*, 1:20-22; 11:60-12:6; 12:41-54. |
| [1b] displaying the document electronically using the first computer program; | Note areas 54a and 54b are documents displayed by the notepad application (first computer program). *See, e.g.*, 6:24-31 ("Additional note areas, such as a note area 54b, can be formed by the user by drawing a substantially horizontal line across the screen 52 with the stylus 38."); 6:49-59 ("The screen illustrated in FIG. 2 is referred to as the 'notepad', and is preferably an application program running under the operating system of the pen based computer system 10."); Fig. 2. |
| [1c] while the document is being displayed, analyzing, in a computer process, first | Luciw discusses entering information into a smart field whether in window 170 as in Fig. 4b or in the notepad application. *See, e.g.*, 8:15-18. |

| | |
|---|---|
| information from the document to determine if the first information is at least one of a plurality of types of information that can be searched for in order to find second information related to the first information; | Further, an implicit assist can also be indicated by writing in the notepad outside of a smart field. *See, e.g.*, 8:30-41 ("However, implicit assist may be indicated not just by entry of an indication in a smart field … [T]he writing of a particular indication or word on screen 52 outside of a particular smart field may trigger an implicit assist."). <br><br> While the document is being displayed, the device in Luciw analyzes a user's entry (first information from the document) to determine if implicit assistance is possible and the kind of implicit assist indicated (determine whether first information can be used to find second information). *See, e.g.*, Figs. 3 and 4a; 10:15-20 ("If the entry in the smart field has been made by the user, the assistance process takes action to identify or recognize the kind of implicit assistance indicated at a step 154."); 8:7-13 ("At step 104, the process recognizes whether or not an implicit assistance function is to be provided by computer system 10. … If a user does enter information into a 'smart field,' the computer database will be queried at step 106 to determine whether assistance is possible given the user input."). |
| [1d] retrieving the first information; | The first information is retrieved to determine if implicit assist is possible. *See* claim 1c. |
| [1e] providing an input device, configured by the first computer program, that allows a user to enter a user command to initiate an operation, | Luciw provides an input device (the pen pointer and touch screen) configured by the first computer program that allows a user to write a command or hit a button to initiate an operation. Specifically, the user can initiate the assist operation by writing a particular word or hitting the "explicit assist" button. *See, e.g.*, 8:51-53 ("An example of an indication of user desire to have explicit assistance undertaken is the act of using pen 38 in FIG. 2 to tap or click on the assist |

| | |
|---|---|
| | icon or button 64 shown on the surface of stylus-sensitive membrane 62 …"); 8:30-41 ("However, implicit assist may be indicated not just by entry of an indication in a smart field … [T]he writing of a particular indication or word on screen 52 outside of a particular smart field may trigger an implicit assist."); Fig. 12c. *See* narrative below. |

It would have been obvious for the notepad application to provide an interface, such as the "explicit assist" button, to receive a user command. (Menascé Decl. ¶ 89.) As shown in Fig. 2 of Luciw, the "explicit assist" button is provided at the bottom of screen 52, and the notepad application includes buttons on status bar 68. (*See also* 7:53-54 ("The various buttons of the status bar 68 are positioned in a third layer 'over' the second and root layers.").) Therefore, to any extent the "explicit assist" button is not provided in, and thus configured by, the notepad application, it would have been obvious to a person of ordinary skill in the art. (*Id.*) Having the notepad application configure a button would have been a predictable modification of Luciw that was well within ordinary skill, because configuring GUI elements, such as buttons, was a well-known function of word processing programs. (*Id.*)

| | |
|---|---|
| [1f] the operation comprising (i) performing a search using at least part of the first information as a search term in order to find the second information, of a specific type or types, associated with the | The system in Luciw searches a database (external information source) using first information entered into a document, *e.g.*, a first name, for second information associated with the entry, *e.g.*, a last name. *See, e.g.* 10:49-11:39; 11:60-12:6 ("Responsive to the recognition of the name ISAAC, the assistance process has |

| | |
|---|---|
| search term in an information source external to the document, wherein the specific type or types of second information is dependent at least in part on the type or types of the first information, and | produced a list of alternatives by earlier query of the database per step 106 in FIG. 3."); 12:41-54 ("In the earlier example of FIG. 6c in which it was decided that Isaac Asimov was the desired ISAAC, the phone information in window 170 had not yet been entered. This information may be available and can be accessed according to the process of FIG. 8a. *The process starts at 200 and immediately checks the data base for any linked smart fields as indicated at 202.*") (emphasis added); Figs. 3 and 5.<br><br>The system determines the action the user intends to take based on the categories of information entered. *See, e.g.*, 13:52-14:4; 9:46-48 ("Next, an attempt is made at step 135 to recognize the possible intent expressed by the objects entered into the assistance process."). As shown in Fig. 11c, performing actions requires different categories of information. *See, e.g.* 14:5-17. The system determines if any information required to perform the action is missing and retrieves it from the database. *See e.g.*, 15:8-13 ("The process calls for example for the filling in of a plan template and the identification of any missing preconditions, as set forth at step 292 of FIG. 13. Next, a step 293 resolves missing preconditions to the extent possible."). Thus, the action taken and the type of information retrieved (second information) depend on the type of information entered by the user (first information). |
| [1g] (ii) performing an action using at least part of the second information; | The system in Luciw presents for user selection a list of people with the first name identified. The system then inserts the full name (second information) of the person selected (*i.e.*, performing an action using the second information). *See, e.g.*, Figs. 6a-6c; 11:60-12:6 ("Responsive to the recognition of the name |

| | |
|---|---|
| | ISAAC, the assistance process has produced a list of alternatives by earlier query of the database per step 106 in FIG. 3. … The user-selected 'ISAAC ASIMOV' is shown having been marked for selection by a rectangle indicating a highlighting operation. FIG. 6c illustrates the completion of the selection process, with the full name in formal font of ISAAC ASIMOV being presented in the name field 175 of window 170."); 12:41-54. |
| [1h] in consequence of receipt by the first computer program of the user command from the input device, causing a search for the search term in the information source, using a second computer program, in order to find second information related to the search term; and | When the user enters information into a smart field or hits the explicit assist command, the system uses a second program, *i.e.*, a contact database, to search an information source to find related information. For example, the user enters "Isaac" in Fig. 6a and the system searches the contact database for persons with the first name Isaac. *See* claim 1f. |
| [1i] if searching finds any second information related to the search term, performing the action using at least part of the second information, | *See* claim 1g. |
| [1j] wherein the action is of a type depending at least in part on the type or types of the first information. | *See* claim 1f. |

| | |
|---|---|
| 2. A method according to claim 1, wherein the first information comprises at least one of name-, person-, company- and address-related information. | First information can be a name, such as "Isaac" in Fig. 6a. *See* claim 1f. |

| | |
|---|---|
| 3. A method according to claim 2, wherein performing the action includes performing the action in the first computer program. | The information retrieved is inserted into the document and displayed by the first computer program. *See* claims 1b and 1g. |

| | |
|---|---|
| 4. A method according to claim 1, wherein performing the action includes performing the action in the first computer program. | *See* claim 3. |

| | |
|---|---|
| 5. A method according to claim 4, wherein performing the action includes causing addition of at least part of the second information to the first information in the document. | The last name and phone number are added to the first name in the document. *See, e.g.*, Figs. 6a-6c and 8b; 11:60-12:6 ("FIG. 6c illustrates the completion of the selection process, with the full name in formal font of ISAAC ASIMOV being presented in the name field 175 of window 170."); 12:41-63 ("If there are applicable smart fields which contain the desired phone number information, this data is obtained from the corresponding linked field types as suggested at 203. Then, as suggested at 206, the data obtained is entered into the applicable smart field of the window 170 under operation."). |

| | |
|---|---|
| 6. A method according to claim 4, wherein performing the action includes causing display of at least part of the second information. | In Figs. 6b-6c a list of available "Isaacs" is displayed, and the user's selection is inserted into the document. *See* claim 1g. In Fig. 8b the associated phone number is inserted into the document (*i.e.*, displayed). |

| | |
|---|---|
| 7. A method according to claim 4, wherein performing the action includes causing display of at least part of the second information by the first computer program. | Second information inserted into the document (*e.g.*, the last name or phone number) is displayed by the first computer program. *See* claims 1g and 5-6. |

| | |
|---|---|
| 10. A method according to claim 1, wherein receipt by the first computer program of the user command precedes analyzing the document. | The user's selection of the "explicit assist" command precedes analyzing the document. *See, e.g.*, 9:16-10:5 ("If an explicit assist has been indicated at step 110, then a step 130 determines, if a particular selection as to the explicit assistance has been made. … If no user selection has been made, objects entered since a delimiter are entered into the assistant in a step 133. Since no objects have specifically been selected, the objects to be entered into the assistant are selected automatically by a delimiter process."). |

| | |
|---|---|
| 11. A method according to claim 1, wherein analyzing the document is completed after the receipt of the user command is completed and before searching is initiated. | *See* claim 10. |

| | |
|---|---|
| 12. A method according to claim 1, wherein the input device is a graphical input device. | The menu offering selection of several "Isaacs" and the explicit assist button are both graphical input devices. *See, e.g.*, Figs. 6b and 2; 3:14-20; 8:51-55. |

| | |
|---|---|
| 13. A method according to claim 1, wherein the user command is the only command from a user necessary to initiate performing the operation. | Upon selection of the "explicit assist" command, the system automatically identifies the user assist information, such a first name. *See, e.g.*, 9:16-10:5 ("If an explicit assist has been indicated at step 110, then a step 130 determines, if a particular selection as to the explicit assistance has been made. … Since no objects have specifically been selected, the objects to be entered into the assistant are selected automatically by a delimiter process."). The system then automatically selects the person to search for contact information. *See, e.g.*, Figs. 7a-7c, 12:7-40. The database is then searched for related contact information to insert into the document. *See* claims 1f and 1g.<br><br>Thus, selection of the "explicit assist" command is the only command necessary to initiate performing the operation. |

| | |
|---|---|
| 14. A method according to claim 1, wherein the input device is a menu and the entry of the user command includes a user's selection of the menu and click on a menu choice from the menu. | The "explicit assist" command in Luciw is executed by selection of an on-screen button. *See, e.g.*, Fig. 2 at 24; 8:51-53. Other commands in Luciw are selected from a menu. *See, e.g.*, 3:14-20; 11:60-12:6 ("Responsive to the recognition of the name ISAAC, the assistance process has produced a list of alternatives by earlier query of the database per step 106 in FIG. 3. In particular, three ISAAC are presented for selection of one of them …"). |

It was common knowledge that commands can be selected via a button or via a pop-up menu (which are both disclosed in Luciw). (Menascé Decl. ¶ 90.) Thus, it would have been obvious to a person of ordinary skill in the art that selection of the "explicit assist" command could be made from a conventional

menu, as a matter of user interface design. Providing such functionality would have been a predictable modification of Luciw well within ordinary skill. (*Id.*)

| | |
|---|---|
| 15. A method according to claim 1, further comprising, if searching results in a plurality of distinct instances of second information, displaying such instances to enable user selection of one of them for use in performing the action. | If the search returns a plurality of distinct results, the results are displayed to enable the user to select one, as shown by the three "Isaacs" in Fig. 6b. *See, e.g.*, 11:60-12:6 ("Responsive to the recognition of the name ISAAC, the assistance process has produced a list of alternatives by earlier query of the database per step 106 in FIG. 3. In particular, three ISAAC are presented for selection of one of them …"). |

| | |
|---|---|
| 16. A method according to claim 1, wherein the information source is associated with the second computer program and is available on the computer. | Luciw searches a database (information source) available on the computer using a database search program, such as a contact database. *See* claims 1f and 1h. |

| | |
|---|---|
| 17. A method according to claim 1, wherein the information source is associated with the second computer program and is available through the computer. | *See* claim 16. |

| | |
|---|---|
| 18. A method according to claim 1, wherein performing the action includes causing insertion of at least part of the second information into the document. | *See* claim 5. |

| | |
|---|---|
| 19. A method according to claim 1, wherein performing the action includes causing insertion of at least part of the second information into the document by the first computer program. | *See* claim 5. |

| | |
|---|---|
| [20a] 20. A computer-implemented method for finding data related to the contents of a document using a first computer program running on a computer, the method comprising: | *See* claim 1a. |
| [20b] displaying the document electronically using the first computer | *See* claim 1b. |

| | |
|---|---|
| program; | |
| [20c] while the document is being displayed, analyzing, in a computer process on the computer, first information from the document to determine if the first information is at least one of a plurality of types of information that can be searched for in order to find second information related to the first information, and | *See* claim 1c. |
| [20d] wherein the first information comprises at least one of name-, person-, company-, and address-related information; | *See* claim 2. |
| [20e] providing an input device, configured by the first computer program, that allows a user to enter a user command to initiate an operation, | *See* claim 1e. |
| [20f] the operation comprising (i) performing a search using at least part of the first information as a search term in order to find the second information, of a specific type or types, associated with the search term in a user editable information source outside the document, wherein the specific type or types of second information is dependent at least in part on the type or types of the first information, and | *See* claim 1f.<br><br>In claim 15 the user selects a person from suggestions provided. However, the system in Luciw can automatically make a selection for the user by selecting the last used selection, most frequently used, etc. To do so, it maintains a database of persons and usage information, then updates the database upon user selection. *See, e.g.*, Figs. 7a-7c, 12:7-40, 17:7-9 ("A computer system as recited in claim 5 further including means for updating the database to contain information regarding the selected alternative."). When a user enters a person for the first time, the person must necessarily be added to the database. Thus, the database is user editable. |
| [20g] (ii) performing an action using at least part of the second information, | *See* claim 1g. |

| | |
|---|---|
| [20h] wherein the input device includes a menu; | *See* claim 14. |
| [20i] retrieving the first information; | *See* claim 1d. |
| [20j] in consequence of receipt by the first computer program of the user command, such user command including a user's selection of the menu and click on a menu choice from the menu, | *See* claim 14. |
| [20k] causing a search for the search term in the user editable information source, using a second computer program, in order to find second information related to the search term in the user editable information source; and | *See* claim 1h. |
| [20l] if searching finds any second information related to the search term, performing the action using at least part of the second information, | *See* claim 1i. |
| [20m] wherein the action is of a type depending at least in part on the type or types of the first information and | *See* claim 1j. |
| [20n] performing the action includes at least causing display of at least part of the second information. | *See* claim 6. |

| | |
|---|---|
| 21. A method according to claim 20, further comprising, if searching results in a plurality of occurrences of second information, causing display of such instances to enable user selection of one of them for use in performing the action. | *See* claim 15. |

| | |
|---|---|
| 22. A method according to claim 20, wherein performing the action includes causing addition of at least part of the second information to the first information in the document. | *See* claim 5. |

### C. Computer Readable Medium Claims

Computer readable medium claims 23-29 and 32-44 would have been obvious in view of Luciw. These claims correspond to method claims 1-7 and 10-

22. Luciw discloses or renders obvious the steps in the body of the computer readable medium claims (as set forth above with respect to the corresponding method claims) and further discloses a computer readable medium including program instructions (*see, e.g.*, Fig. 1 at 22).

## IX. GROUND 4: OBVIOUSNESS OF CLAIMS 1, 2, 8, 14-17, 20, 21, 23, 24, 30, 36-39, 42, AND 43 IN VIEW OF PANDIT

Pandit was filed on December 27, 1995 and thus qualifies as prior art under § 102(e) based on the earliest alleged U.S. filing date of the '843 patent. As set forth in the title, Pandit is directed to recognition of and operation on text data. For example, a document is illustrated in Figs. 1a-1f. Various text items in the document can be selected by the user and analyzed to determine the nature of the text, *e.g.*, whether it is a date, an e-mail address or a phone number. Based upon this determination, various actions relating to the determined type of text can be made available for selection by the user. For example, as shown in Fig. 1f below, determination that a selected text item is a phone number can result in provision of available actions including calling the number, adding the number to an address book or sending a fax to the number.

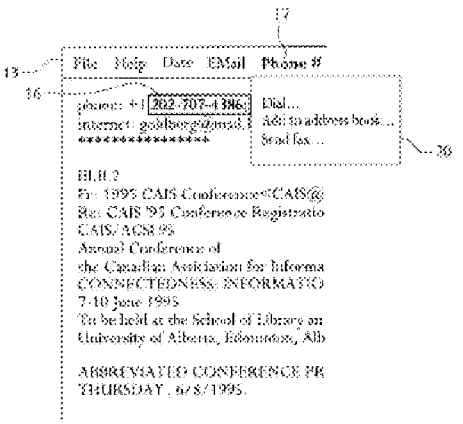



FIG. 1f

### A. Method Claims

Set forth below is a claim chart that specifies where each element of method claims 1, 2, 8, 14-17, 20, and 21 are met by Pandit. Any narrative discussion with respect to obviousness for a given claim or claim element is provided directly under that claim or claim element with double line spacing.

| Claim | Pandit |
|---|---|
| [1a] 1. A computer-implemented method for finding data related to the contents of a document using a first computer program running on a computer, the method comprising: | Pandit discloses a computer-implemented method for finding data related to identified text. *See, e.g.*, 5:25-43; Abstract ("Text of a predetermined class is recognized in a body of text. After recognition, operations relevant to the recognized text may be performed."); 3:1-15. |
| [1b] displaying the document electronically using the first computer program; | The document is displayed using a first computer program. *See, e.g.*, 5:18-21 ("Any text appearing on a video monitor can be operated on by the invention, whether the text is within an EMail message, World-Wide Web site, created by a word processing or database program, etc."). |
| [1c] while the document is being displayed, analyzing, in | While the document is being displayed, the text is analyzed in a computer process to determine if |

| | |
|---|---|
| a computer process, first information from the document to determine if the first information is at least one of a plurality of types of information that can be searched for in order to find second information related to the first information; | the text is of a type that can be searched to find related information. *See, e.g.*, 2:8-15, 25-32 ("…the invention is not limited to the recognition of dates in text and preferred embodiments of the invention can recognize e-mail addresses and telephone numbers…") |
| [1d] retrieving the first information; | The text is retrieved and identified. *See* claim 1c. |
| [1e] providing an input device, configured by the first computer program, that allows a user to enter a user command to initiate an operation, | Pandit discloses providing a menu (input device) that allows a user to select an operation to be performed. *See, e.g.*, FIGS. 1b, 1d, and 1f; 2:8-23 ("A view of an embodiment of a pulled-down Date menu 18 is shown in FIG. 1b. A user may directly call a calendar or appointment database program from pulled-down menu 18. Other programs may be included in pulled-down date menu 18 as discussed below."). *See* narrative below. |

To the extent that Pandit does not explicitly disclose that the menu is "configured by" the first application program, it would have been obvious to a person of ordinary skill in the art that the menu would be configured by the first application program in order to be displayed with the first application program. (Menascé Decl. ¶ 98.) This would have been a predictable modification of Pandit that was well within ordinary skill, because configuring a menu was a well-known function of word processing programs. (*Id.*)

| | |
|---|---|
| [1f] the operation comprising | If the identified text is of a certain type, the user |

| | |
|---|---|
| (i) performing a search using at least part of the first information as a search term in order to find the second information, of a specific type or types, associated with the search term in an information source external to the document, wherein the specific type or types of second information is dependent at least in part on the type or types of the first information, and | can use the text to search an external information source to find information associated with the text. For example, searching a dictionary for the meaning of an identified word. *See, e.g.*, 3:11-15 ("Where the invention is capable of recognizing nouns or verbs, pull-down menus can, for example, identify executable programs which provide the meaning of the highlighted word, appropriate synonyms and the singular or plural version of the noun or conjugation of the verb.").<br><br>Further, Pandit discloses adding an identified number to an address book. *See, e.g.*, Figs. 1d and 1f; 2:56-53; 3:1-10 ("As shown in FIG. 1f on pulled-down menu 20, possible programs include a writable computer database of telephone and telefax numbers …"). *See* narrative below.<br><br>The type of second information depends on the type of first information. For example, if the first information is a phone number, the second information is contact information associated with the phone number. |

It would have been obvious to a person of ordinary skill in the art that the first step in adding to an address book is searching the address book to determine if an entry already exists with this information and displaying any associated information which is located. (Menascé Decl. ¶ 99.) This would have been a matter of common sense to one of ordinary skill, in order to avoid multiple entries of the same address. (*Id.*)

| | |
|---|---|
| [1g] (ii) performing an action using at least part of the second information; | The meaning of the identified word and the contact information associated with the identified number are displayed for the user (an action using |

| | |
|---|---|
| | second information). *See* claim 1f. |
| [1h] in consequence of receipt by the first computer program of the user command from the input device, causing a search for the search term in the information source, using a second computer program, in order to find second information related to the search term; and | When the user selects an action from the menu of available operations, the system uses a second computer program (*e.g.*, dictionary program or address book program) to search the information source using the identified text. *See* claim 1f. |
| [1i] if searching finds any second information related to the search term, performing the action using at least part of the second information, | *See* claim 1g. |
| [1j] wherein the action is of a type depending at least in part on the type or types of the first information. | The action performed depends on the type of information identified in the document. For example, defining an identified word versus looking up associated contact information for an identified number. *See* claims 1f-1g. |

| | |
|---|---|
| 2. A method according to claim 1, wherein the first information comprises at least one of name-, person-, company- and address-related information. | The system in Pandit can identify name and address related information. *See, e.g.*, 2:28-32 (”[T]here is no limit on the type of text which can be recognized by the invention and additional embodiments can recognize such classes of text as … names, street addresses, etc.”); 4:29-31. |

| | |
|---|---|
| 8. A method according to claim 1, further comprising, providing a prompt for updating the information source to include the first information. | The system prompts the user to select an action to perform by displaying a pop-up menu of actions, as discussed in claim 1e. Several of these actions update an information source to include first information. *See, e.g.*, Figs. 1d and 1f (“Add to address book”); 3:1-3 (“As shown in FIG. 1f on pulled-down menu 20, possible programs include a writable computer database of telephone and telefax numbers …”); 2:46-49. |

| | |
|---|---|
| 14. A method according to claim 1, wherein the input device is a menu and the entry of the user command includes a user's selection of the menu and click on a menu choice from the menu. | The user selects a command from a menu by clicking a menu choice. *See, e.g.*, 2:41-45 ("A user is able to run one or more of the programs relevant to dates which are identified in the pulled-down menu in a known manner, such as by clicking on the name of the program as it appears in the pulled-down menu …"). |

| | |
|---|---|
| 15. A method according to claim 1, further comprising, if searching results in a plurality of distinct instances of second information, displaying such instances to enable user selection of one of them for use in performing the action. | *See* narrative below. |

Pandit discloses recognizing names in a document (2:25-32) and performing actions using an address book (Figs. 1d and 1f; 2:56-61; 3:1-3). In view of this, it would have been obvious to a person of ordinary skill in the art to enable a user to call a person with the identified name by searching the address book, and, if the person had multiple numbers in the address book, to display them for selection. (Menascé Decl. ¶ 100.) This would have been simply a matter of common sense and common knowledge and there would have been design and market incentives to provide such functionality. (*Id.*) One of ordinary skill would have been able to apply a known technique (displaying for selection plural results of a search) to the known method of Pandit to yield a predictable result. (*Id.*)

| | |
|---|---|
| 16. A method according to claim 1, wherein the information source is | The dictionary and address book (information source) are associated with |

| | |
|---|---|
| associated with the second computer program and is available on the computer. | the dictionary and address book applications and are available on and through the computer. |

| | |
|---|---|
| 17. A method according to claim 1, wherein the information source is associated with the second computer program and is available through the computer. | *See* claim 16. |

| | |
|---|---|
| [20a] 20. A computer-implemented method for finding data related to the contents of a document using a first computer program running on a computer, the method comprising: | *See* claim 1a. |
| [20b] displaying the document electronically using the first computer program; | *See* claim 1b. |
| [20c] while the document is being displayed, analyzing, in a computer process on the computer, first information from the document to determine if the first information is at least one of a plurality of types of information that can be searched for in order to find second information related to the first information, and | *See* claim 1c. |
| [20d] wherein the first information comprises at least one of name-, person-, company-, and address-related information; | *See* claim 2. |
| [20e] providing an input device, configured by the first computer program, that allows a user to enter a user command to initiate an operation, | *See* claim 1e. |
| [20f] the operation comprising (i) performing a search using at least part of the first information as a search term in order to find the second information, of a specific type or types, associated with the search term in a user editable information source outside the document, wherein the specific type or types of second information is dependent at least in part on the type or types of the first information, and | *See* claim 1f. The user can add information to the address book, thus it is user editable. *See* claim 8. |
| [20g] (ii) performing an action using at least part of the second information, | *See* claim 1g. |
| [20h] wherein the input device includes a menu; | *See* claim 14. |
| [20i] retrieving the first information; | *See* claim 1d. |
| [20j] in consequence of receipt by the first computer program of the user command, such user command including a user's selection of the menu and click on a menu choice from the | *See* claim 14. |

| | |
|---|---|
| menu, | |
| [20k] causing a search for the search term in the user editable information source, using a second computer program, in order to find second information related to the search term in the user editable information source; and | *See* claim 1h. |
| [20l] if searching finds any second information related to the search term, performing the action using at least part of the second information, | *See* claim 1i. |
| [20m] wherein the action is of a type depending at least in part on the type or types of the first information and | *See* claim 1j. |
| [20n] performing the action includes at least causing display of at least part of the second information. | *See* claim 15. |

| | |
|---|---|
| 21. A method according to claim 20, further comprising, if searching results in a plurality of occurrences of second information, causing display of such instances to enable user selection of one of them for use in performing the action. | *See* claim 15. |

### B. Computer Readable Medium Claims

Computer readable medium claims 23, 24, 30, 36-39, 42, and 43 would have been obvious in view of Pandit. These claims correspond to method claims 1, 2, 8, 14-17, 20, and 21. Pandit discloses or renders obvious the steps in the body of the computer readable medium claims (as set forth above with respect to the corresponding method claims) and further discloses a computer readable medium including program instructions (*see, e.g.*, 5:25-46).

## X. CONCLUSION

For the reasons detailed above, there is a reasonable likelihood that Petitioner will prevail as to each of claims 1-44 of the '843 patent. Accordingly, *inter partes* review of claims 1-44 of the '843 patent is respectfully requested. The

USPTO is authorized to charge any required fees, including the fee as set forth in 37 C.F.R. § 42.15(a) and any excess claim fees, to Deposit Account No. **03-1952** referencing Docket No. **106842805100**.

Dated: December 2, 2013

Respectfully submitted,

By /David L. Fehrman/
David L. Fehrman
Registration No.: 28,600
MORRISON & FOERSTER LLP
707 Wilshire Blvd., Suite 6000
Los Angeles, California 90017-3543
(213) 892-5601

By /Mehran Arjomand/
Mehran Arjomand
Registration No.: 48,231
MORRISON & FOERSTER LLP
707 Wilshire Blvd., Suite 6000
Los Angeles, California 90017-3543
(213) 892-5630

**Certificate of Service (37 C.F.R. § 42.6(e)(4))**

I hereby certify that the attached Petition for *Inter Partes* Review and supporting materials were served as of the below date by FedEx, which is a means at least as fast and reliable as U.S. Express Mail, on the Patent Owner at the correspondence address indicated for U.S. Patent No. 7,917,843 (*i.e.*, Sunstein Kann Murphy & Timbers LLP, 125 Summer Street, Boston, MA 02110-1618).

Dated: December 2, 2013

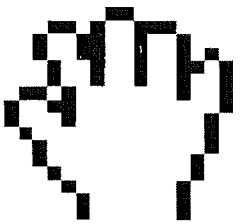
/Mehran Arjomand/
Mehran Arjomand
MORRISON & FOERSTER LLP
707 Wilshire Blvd, Suite 6000
Los Angeles, CA 90017-3543

# Exhibit 19

SIGCHI



**April 1998**
Volume 30, Number 2
ISSN 0736-6906



# A Farewell to the Apple Advance Technology Group



acm PRESS

A Quarterly Publication of the ACM Special Interest Group on
**Computer-Human Interaction**
www.acm.org/sigchi/bulletin

SIGCHI Bulletin (ISSN 0736-6906) is published quarterly by ACM, 1515 Broadway, New York, NY 10036, USA. Periodicals postage paid at New York, NY 10001, and at additional mailing offices. POSTMASTER: Send address changes to SIGCHI Bulletin, ACM, 1515 Broadway, New York, NY 10036. Printed in USA.

Annual subscription cost of $17.19 is included in the member dues of $30 (for students, cost is included in $10); the non-member annual subscription is $57.


**Scope.** The scope of Special Interest Group on Computer-Human Interaction (SIGCHI) is the study of human factors in the human-computer interaction process, including research, design, development, and evaluation of interactive computing systems. The focus is on human communication and interaction with computer systems. SIGCHI provides a forum for the exchange of ideas among computer scientists, behavioral and cognitive scientists, system designers, and end users, and it serves as a clearinghouse of information for the field of human factors and user psychology research and development.

**Membership.** You are invited to join and participate in SIGCHI functions. Membership in SIGCHI, which includes a subscription to the *SIGCHI Bulletin*, is open to ACM members and non-members. For subscriptions and to become a member of SIGCHI (or for change of address), complete the form in the back of this issue. Correspondence regarding subscriptions, back issues, or membership should be sent to ACM Member Services, 1515 Broadway, New York, NY 10036, USA; email: ACMhelp@acm.org

**Submissions.** Materials for editorial consideration should be submitted to the appropriate editor. *Electronic submissions are strongly encouraged.* Please include an email address if possible, as well as a postal address and short biography. Materials suitable for publication include technical submissions, correspondence, announcements of research reports and publications, conference and workshop reports, news items, annotated bibliographies, book reviews and other items of general interest.

**Deadlines.** Copy deadlines are the 1st of the third month preceding date of issue e.g., October 1 for the January issue. For further submission instructions please refer to http://www.acm.org/sigchi/bulletin/

Opinions expressed in signed articles and letters are those of the writer and do not necessarily express the position of the ACM or SIGCHI. Reports and technical papers in the Bulletin are unrefereed working papers, unless otherwise stated. Materials may be reproduced for noncommercial purposes, if credit is given to the *SIGCHI Bulletin* and ACM/SIGCHI.

**Advertising.** ACM accepts recruitment advertising under the basic premise that the advertising employer does not discriminate on the basis of age, color, race, religion, gender, sexual preference, or national origin. ACM recognizes, however, that laws on such matters vary from country to country and contain exceptions, inconsistencies or contradictions. This is as true of laws of the United States of America as it is of other countries. Thus ACM policy requires each advertising employer to state explicitly in the advertisement any employment restrictions that may apply with respect to age, color, race, religion, gender, sexual preference, or national origin. Observance of the legal retirement age in the employer's country is not considered discrimination under this policy.

For advertising information, please contact Walter Andrzejewski (Advertising Manager) at ACM, 1515 Broadway, New York, NY 10036 USA; +1-212-869-7440; Fax: +1-212-869-0481. Email: acm-advertising@acm.org.

# An Overview of the ATG Intelligent Systems Program

**James R. Miller**

The potential of intelligent user interfaces have been obvious for many years one only has to look at the *Knowledge Navigator* video or any of a large number of science fiction novels or films to appreciate what it could be like to interact with computers in ways analogous to how people interact with other people. However, the computing industry, throughout this time, has remained incapable of building systems with these capabilities. The technology demands are great, as are the human interaction design problems that must ultimately be solved to yield a system that is truly useful to people.

Consequentially, the Intelligent Systems Program chose a different approach: that of *partial understanding*. Since the dream of truly intelligent interaction lies beyond our capabilities, we worked to approximate this dream with technologies that were available to us. This led to a different approach to the intelligent interface question and to our work: we thought less about problems that have resisted solution for decades, and more about the real needs that users have and how they can best be satisfied. The tools of the intelligent interfaces community – parsers, inference engines, knowledge representation, language understanding – remained central our work, lurking beneath the surface. However, they were applied in ways that work today, and that yield successful solutions to user needs. One of these solutions – *Apple Data Detectors*[1] [5] – is shipping as an Apple product today; we hope and expect that other solutions based on this approach will join it soon.

Over the lifetime of the program, we applied this general strategy of partial understanding to several problem areas. These addressed different questions about user tasks and interaction styles, and presented different opportunities for technologies to help solve those problems. Several of the papers in this volume describe our experiences with this approach; in particular:

- *Apple Data Detectors and beyond*: Our work on Apple Data Detectors was based, in part, on a desire to move towards a broader notion of what constitutes a document, and how documents can be transformed from today's simple streams of characters into highly interactive computational objects. Our work on *LiveDoc* [4] and *Drop Zones* [3] outline increasingly powerful systems that take us in this direction by finding meaningful components of documents and making them the locus of task-based interaction.
- *Language-based knowledge mining*. Perhaps, at some time in the future, computers will be able to derive the same kinds of understandings of written documents that humans can. But what do we do until then? One approach, which we have pursued in our work on natural language processing, tries to find a middle ground between these statistical techniques and AI-based understanding techniques. The idea here is to use a low-level linguistic analysis of the document to identify terms from the document that are probably central to that document's meaning. It is then possible to use these terms to reconstitute how the terms are interrelated and, from there, some of the document's high-level structure. The paper on SCOOP [2] describes this set of language technologies and some sample applications of them; the paper on dynamic document presentation [1] shows how this approach can be used as a central part of an innovative interactive system.
- *Networked communities and distance education*: Finding the content of documents does us little good if we can't communicate that content to the people who need it. Hence, we also

1. Downloadable at http://applescript.apple.com/data_detectors

studied how communities grow up around bodies of information, and what technologies can insure the successful dissemination of that information to the members of that community. One experience with such a community is described in [6], both in terms of the technologies we found to be useful for providing community support, and in how the participants in the community felt about these kinds of interaction and collaboration.

It's important to be clear about the perspective we chose for this work. This was not a rejection of artificial intelligence as a source of useful technologies. It was, however, a fairly explicit rejection of the belief that we could count on the imminent solution of "the AI problem" – that, suddenly, systems with the near-human capabilities envisioned for so long would suddenly become available. But this was not so much pessimism as pragmatism and practicality. We were not working to find great new applications of AI technologies, but to identify problems in peoples' everyday lives and to propose solutions for them. If we chose to let go of the full-bore AI dream, we were perhaps choosing instead the freedom that comes from making use of the technologies and strategies that we have today, and that can be applied to real problems with immediate as well as long-term payoff. And, if the AI problem gets solved next week, there are plenty of ways to fit them into what you'll read about here....

## References


1. Boguraev, B. & Bellamy, R. (1988). Dynamic document presentation. *SIGCHI Bulletin*, this volume.
2. Boguraev, B., Kennedy, C., & Brawer, S. (1988). An architecture for content analysis of documents and its use in information and knowledge management tasks. *SIGCHI Bulletin*, this volume.
3. Bonura, T., & Miller, J. R. (1998). Drop Zones: An extension to LiveDoc. *SIGCHI Bulletin*, this volume.
4. Miller, J. R., & Bonura, T (1998). From documents to objects: An overview of LiveDoc. *SIGCHI Bulletin*, this volume.
5. Nardi, B. A., Miller, J. R., & Wright, D. J. (1998). Collaborative, programmable intelligent agents. *Communications of the ACM*, Vol. 41, No. 3 March,1998.
6. Nardi, B. A., Reilly, B., & Steinbeck, R. (1988). An online digital photography course for high school teachers. *SIGCHI Bulletin*, this volume.


## Acknowledgments

As ATG came to an end, I was joined in the Intelligent Systems Program by Bran Boguraev, Tom Bonura, Eric Hulteen, Bonnie Nardi, and Dave Wright. Thanks are also due to our colleagues from previous years, including Yu-Ying Chow, Laile Di Silvestro, Dan Russell, and Bob Strong, as well as a number of contractors and interns, including (and with apologies to those I may have forgotten) Ken Anderson, Elizabeth Bratt, Sascha Brawer, Jon Doyel, Bruce Horn, Chris Kennedy, Henry Lieberman, Pattie Maes, Dave McDonald, Max Metral, Justina Ohaeri, Jeremy Sandmel, Shilpa Shulka, Oliver Steele, Heinrich Schwarz, Jason Swartz, Marc Verhagen, Yin Yin Wong, and Dave Yost.

## About the Author

Jim Miller, until recently, was the program manager for Intelligent Systems in Apple's Advanced Technology Group. He is currently exploring consumer applications of Internet technology as part of *Miramontes Computing*.

## Author's Address

Jim Miller
Miramontes Computing
828 Sladky Avenue
Mountain View CA 94040, USA
email: jmiller@millerclan.com
Tel: +1-650-967-2102

# From Documents to Objects

## An Overview of LiveDoc

**James R. Miller and Thomas Bonura**

One of the changes that the World Wide Web has brought to the computing industry is a new way of thinking about documents. Traditionally, documents have been seen as simple streams of characters, like those in a document editor. Applications that manage these documents may do more or less interesting things to the characters, but they rarely attempt to interpret any of the meaning of the document. There's obviously meaning there, but it only becomes apparent when read or otherwise manipulated by a human. In contrast, the Web has brought with it the concept of a document that has been authored in such a way that important bits of information are explicitly identified within the document. This identification exposes some of the meaning of the document, albeit at a fairly low level, so that various kinds of actions – primarily "show me this related document" – are offered to users and made easy for them to carry out.

The gap that separates these two notions of *document* is the need for the human authoring of the Web document. More to the point, it's the need for a human to identify the meaningful components of the document and the actions that make sense for those components. There is a real opportunity to advance the computing field here, by bringing these two worlds together: by enabling an ordinary document, built with any application, to automatically offer users access to some of the meaningful bits of its content, and by helping users carry out appropriate actions on these objects.

### Bridging the Gap through Structure Detection

This premise led to a collection of projects within Apple's Advanced Technology Group – within the Intelligent Systems Program, in particular – on the idea of *structure detection*. The work was based on the observation that, while automatically computing a high-level understanding of an arbitrary document is beyond our present ability, many meaningful bits of information are computationally quite easy to recognize: recognizing an e-mail address ("fred@apple.com") or a URL ("http://www.apple.com") takes little more than a context-free grammar, if not merely a regular expression parser. A first step to bridging the document gap described above is then to construct a means of passing text from a user's document into a parser for matching against a collection of recognizers, each of which is looking for some meaningful type of information. These identifications imply simple interpretations of the bits of information that were found: URLs are found by the URL grammar, e-mail addresses are found by the e-mail address grammar, and so on. Then, actions appropriate to each kind of object can be offered, supporting users in their work on those objects and on the document as a whole.

Our overall intent here – to examine document content, identify likely user actions, and provide simple ways of selecting and executing those actions – is not unlike that of the authors of other "intelligent" critic and advisory systems [e.g., 4, 6]. However, our work on structure detection differs from these systems in a number of ways:

- Syntactically-regular information structures, and the tasks that follow from them, can be found in almost any user domain. Hence, the total number of structures and tasks for which structure detection assistance would be helpful is too large for any single person or organization to try to satisfy. Therefore, we have paid special attention to the importance of allowing application developers and even end-users to define and extend the set of detectors and actions. This drove us to design

a plug-in architecture for object recognizers and actions, as discussed in [7].

- We have remained open to large numbers of applications – ideally, any application available on the Macintosh platform – and have provided users with a consistent human interface across those applications.
- Viewing this work from an agent perspective, we have worked to keep the user well in control of the system's actions. This has called for a clear and explicit connection between the information that was found and the actions taken by users on that information. We have also kept the grain size of the actions relatively small, so that they would be easily understandable and, if need be, undoable.
- We wanted to keep our approach practical enough that it could ship as a commercial product, rather than simply being a research project.

Our most successful instantiation of the idea of structure detection (thus far) has been *Apple Data Detectors* (ADD): a system extension for Mac OS 8 that is now commercially available[1]. This first version of structure detection has been applied to the domain of Internet information management: finding structures like e-mail addresses, URLs, host names, and newsgroup names in user documents and automating actions on these structures, like creating a new e-mail message addressed to a discovered e-mail address or opening a web browser on a discovered URL. This capability has been implemented via Mac OS 8's *contextual menus*: Users select a region of text and activate the contextual menu by pressing the keyboard's Control key and the mouse button. This initiates the grammatical analysis of the selected text, and presents a hierarchical menu consisting of the structures found by the grammars and, for each of those structures, the actions that make sense for it. Users can examine all the discovered structures and their associated actions, and invoke whichever action they choose (Figure 1). More on the origin of Apple Data Detectors, its implementation, and how it made the transfer from research to product can be found in [7]. Some other related approaches to the structure detection question can also be found in [5] and [8].

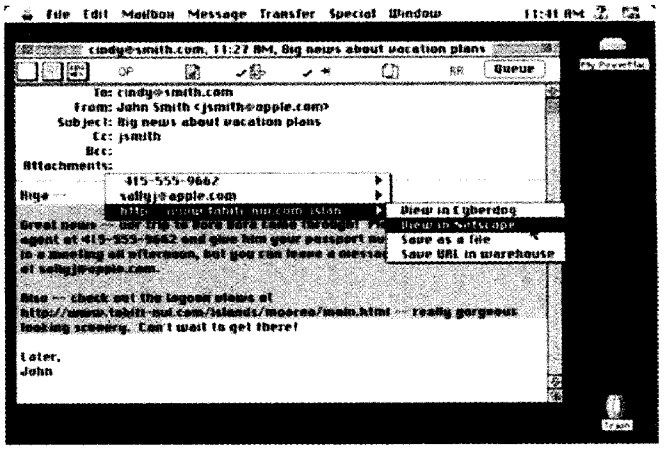



**Figure 1:** The results of invoking Apple Data Detectors on a text selection.

## LiveDoc: Beyond Data Detectors

Many of the goals we had set for shifting the model of documents from passive streams of characters to manipulable collections of meaningful objects were met quite nicely by Apple Data Detectors. Nevertheless, we saw other opportunities for structure detection, growing out of both limitations of this first system and opportunities for expanding its breadth and depth.

Consider, first, the user interface to ADD. Users must select a range of text in a document, and invoke ADD through the contextual menu system. At that point, the selected text is sent to ADD's parser for analysis, which produces the menu of discovered structures and corresponding actions. This design has a number of implications on what kinds of interaction services can be offered to users:

- The basic premise of ADD is that it finds and selects manipulable bits of information for the user. However, in practice, the user must still find a structure that they would like to do something with, so that they can select, for analysis by ADD, a region of text containing that structure. It's true that ADD allows users to be inexact in selecting the region of text containing the structure of interest, since its grammars will find the desired structures in a stream of other, irrelevant characters. However, the tasks of discovering structures that might be operated upon and selecting the parts of the document around those structures are still imposed upon the user by ADD.
- It is not uncommon for useful structures to be nested within other useful structures. The URL "http://www.apple.com/default.html" contains within it the host name "www.apple.com"; a similar situation holds for the host name embedded in an e-mail address. These multiple structures can make the contextual menu generated by a typical text selection quite long, forcing the user to choose between limiting the set of active detectors (which keeps the task of finding the desired structure in the contextual menu a manageable one) and having to search through a large number of detected structures in the contextual menu (which makes ADD's services applicable in the largest number of situations).
- Listing the detected structures in the contextual menu doesn't really make them manipulable, through such direct manipulation techniques as drag and drop.

We encountered still other limitations resulting from our desire to increase the flexibility and power of ADD analyses. For instance, there really isn't any semantic interpretation of the discovered structures in ADD: Actions are associated with structures through a lookup table, not through any rich semantic representation of, for instance, what a URL is, what it might be used for, and what constraints exist on its use. As a result, the set of actions that can be offered to a user is fixed: it can neither include nor omit actions based on the semantics of the immediate interaction context. Further, since ADD's processing is tied to and activated by the contextual menu system, it

[1] as a a free plug-in for Mac OS 8; it is available via http://applescript.apple.com/data_detectors.

must complete its analysis in the very short period of time in which users are willing to wait for a pop-up menu to appear[2] – about half a second. This is typically enough time for ADD to run a set of precompiled grammars and build a menu from a lookup table. However, it's easy to imagine more complex analyses of documents that could not be completed in this short amount of time.

Finally, we saw other opportunities for alternate analysis techniques, which could augment and extend the analysis model of ADD. ADD was built around a single parser, roughly equivalent to a context-free grammar (CFG); all structures found by ADD must be describable by a CFG. This is fine for simple structures such as phone numbers and e-mail addresses, but other more complex structures, such as a meeting announcement, are commonly found in a form not describable by a CFG, and so cannot be recognized by ADD. Some of these structures might be found by a more expressive grammar formalism, and others by recognizers that find structures by powerful but ad hoc analysis techniques instead of grammars. In addition, this textual grammar system cannot be easily extended to handle non-textual structures, such as images, drawings, circuit diagrams, or other kinds of visual information. All of these extensions indicate the need for a richer analysis model than was provided in ADD.

As a result, we began a follow-on project to ADD, known as *LiveDoc*, which would carry the ideas of structure detection forward to another level. In LiveDoc, the structure detection process is run in the background on the visible document's text, whenever that document is presented or updated. The results of LiveDoc's analysis are then presented by visually highlighting the discovered structures with a patch of color around the structure. Holding down a function key places the document in "LiveDoc mode" and presents the highlighted structures; releasing the function key returns the document to normal. Pointing at a highlight and pressing the mouse button then displays the menu of actions that can be applied to the structure, as shown in Fig 2.

Experientially, the design of LiveDoc draws on the Web in obvious ways: certain meaningful parts of a document are highlighted, and clicking on them causes certain actions to occur. LiveDoc differs from the Web, of course, in that we have substituted the automatic analysis of the document for the hand-authored links of web documents, so that any document in a LiveDoc-enabled application (more on this later) gains these characteristics. We are also able to assign more than one action to an object, something that lies outside the standard Web paradigm.

LiveDoc's use of background processing and automatic highlighting of discovered structures offers other advantages. Structures relevant to the user are automatically presented to the user while a document is in LiveDoc mode; interesting structures need not be searched for and highlighted manually. The problem of overly-long contextual menus is avoided, since a menu shows only the actions relevant to the structure it is associated with. Similarly, nested structures can be handled by nesting the mouse-sensitive regions around the structure: clicking on the host name part of a URL can present a menu of actions relevant to host names, while clicking outside the host name region presents actions relevant to URLs. This is shown in Figure 2: the host name of the e-mail address and URL ("sci.org") is shown with a darker highlight than those of the e-mail address and URL themselves. Finally, note that the visual representation of these structures means that, given appropriate software support, they can be treated as directly accessible components of the interface: they can take part in drag and drop interactions, and in other forms of direct manipulation.

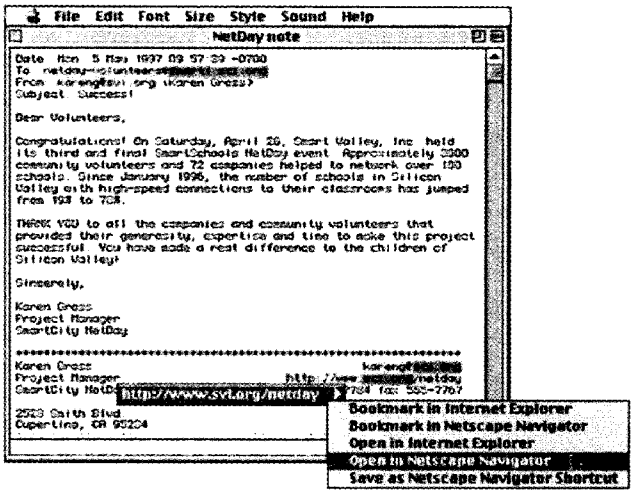



**Figure 2:** A sample interaction with LiveDoc. Note the highlighting of the discovered structures, the menu of actions available or the selected structure, and the nested highlighting of nested structures.

What is described above is, of course, only a general design for LiveDoc. To understand how this design can be implemented, it's necessary to look more closely at the system's architecture, and at its instantiation in several different working systems.

### The LiveDoc architecture: A General Description

Architecturally, LiveDoc is built around the LiveDoc Manager (Figure 3). This component acts as an intermediary between the application making use of LiveDoc and the various internals of LiveDoc itself. In particular, the Analyzer System is made up of a set of detectors that analyze the content of the document passed to LiveDoc, a set of actions (typically, but not necessarily, implemented as AppleScripts) that carry out the various operations on the discovered structures, a table that specifies the mapping between detectors and actions, and an Analyzer Server that coordinates all these functions.

To make use of LiveDoc, applications must implement a small number of calls to the LiveDoc Manager, and a small number of

2. In fact, the contextual menu system enforces a timeout after this amount of time: contextual menu plug-ins, like ADD, that have not completed their analysis when this time expires are halted by the contextual menu manager. When this happens, a "More actions..." option is added to the contextual menu; selecting this runs all the plug-ins to completion, and brings up a dialog box containing the results of their analyses.

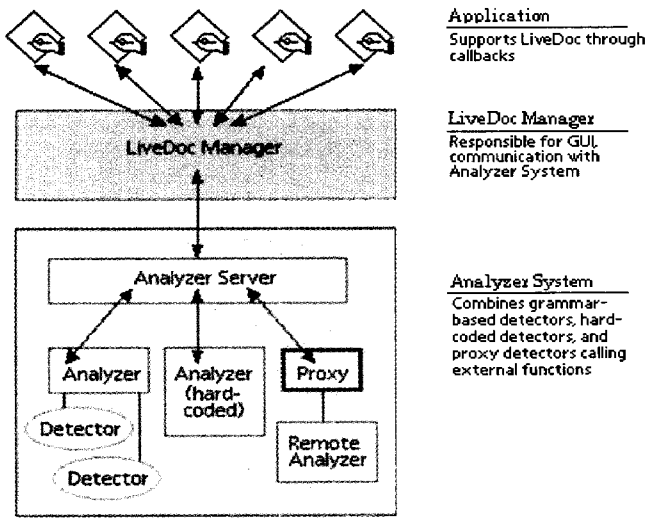



**Figure 3:** The high-level LiveDoc architecture

callback handlers to respond to calls from the LiveDoc Manager. The most important of these handlers inform the LiveDoc Manager of changes to the content of the document window, perhaps by the user's adding or deleting content, or by the scrolling or resizing of the window. The receipt of these calls by the LiveDoc Manager signals the Analyzer Server to analyze the text provided by the calling application; this will typically be the text currently visible in the applications' front-most window. Once the analysis is completed and the structures in the text have been identified, the LiveDoc Manager constructs the various highlights for the discovered structures and their corresponding menus of actions. LiveDoc knows where these structures appear in the text passed to it – an e-mail address might appear in characters 150 through 162 of the window's contents – but it has no idea where in the window those characters physically appear, and, thus, where the highlights should appear: this is information held by the application, not by LiveDoc. Hence, LiveDoc must ask the application for the information about the structures it has found via a callback. Once this information is available, the highlights and their associated mouse-sensitive regions can be constructed.

The LiveDoc Manager also controls the events that occur when the user presses the function key to enter LiveDoc mode, and when the mouse button is pressed while over a LiveDoc item. The LiveDoc Manager updates the display to present the highlight information over the discovered structures when the function key is pressed, and to remove the highlights when the function key is released. The LiveDoc Manager also receives the notification that the mouse button has been pressed over a highlighted item; it then gets the list of actions appropriate to the selected item and presents a menu of them to the user. If one of these items is selected, the action corresponding to the selection is run, producing the desired action.

### Implementation 1: LiveSimpleText

Our initial explorations of LiveDoc were implemented in Lisp, primarily to gain a rapid understanding of some of the implementation issues we would ultimately face and to explore various human interface ideas. However, we soon needed to turn our attention to a system-level API for LiveDoc that would be both efficient and not terribly burdensome to developers who would have to modify their applications to take advantage of LiveDoc.

To test our first implementation of the LiveDoc API, we decided to modify a simple text editor application, SimpleText, to be a LiveDoc client. Although the API was designed as a Macintosh toolbox manager, so that it could be incorporated into existing applications without requiring developers to compile and link our code with theirs, this initial implementation did require linking portions of LiveDoc code with that of SimpleText. (This would be changed in later work where the LiveDoc manager was built as a shared library, allowing the LiveDoc Manager to be called by multiple LiveDoc clients.) We also needed a set of analyzers that could provide the document analysis services to the Analyzer Server. We decided to use the Apple Data Detectors context-free grammar engine, and to additionally implement a fast string search algorithm, as described by Aho and Corasick [1]; this analyzer rapidly finds all instances of strings in a document by comparing the text of the document to a set of dictionary entries. In doing so, we were able to test the multiple-analyzer part of the Analyzer Server and confirm its utility. As part of this test, we gave each analyzer its own interface affordance by varying the colors of the highlights: green for items found using a context free grammar (the Apple Data Detectors analysis engine) and pink for those items found using the string search.

The background processing of LiveDoc raises the issue of the proper way in which to update the display when changes are made to the document. Whenever the text in the window is changed, by either the system or the user, LiveDoc must re-analyze the text, since recognizable structures may have appeared or disappeared. But when should this analysis be done? It makes little sense to analyze a document while the user is working, since each change will require a re-evaluation of the text. Our current implementation works with a timeout, which starts a re-analysis when the keyboard has been inactive for a short period of time (about one second) after a change to the display. In this way, LiveDoc does not analyze the document while the user is typing, but resumes when there is a pause in the user's actions. We have considered various algorithms that might minimize the cost of this analysis – perhaps only analyzing the part of the window that had been scrolled, for instance – but our current implementation reanalyzes the entire content region of a window when a change is detected.

Overall, we believe there are some very compelling aspects of the LiveDoc interface as compared to Apple Data Detectors. As shown in Figure 2, LiveDoc displays its discovered structures in place, in the context in which they occur. It associates a menu of options with the object found where, in Apple Data Detectors, a single menu appears for all of the items found in the selection. Finally, LiveDoc works quietly in the background and displays the results of its analysis on demand, rather than performing the analysis on demand. Having said that, there are some user inter-

face issues that we have not explicitly evaluated as part of this work, and that would benefit from further study. The use of highlighting is one of these: adding the notion of a sometimes-visible layer to the front of the display is a considerable change to the graphical interface, and, while others have pursued similar uses of translucency (e.g., [2]), its overall utility and understandability is worthy of study.

Similarly, the use of increasing degrees of saturation of the highlights should be examined: are these differences clear enough to be discernible, and do they require a precision in mouse control that users both possess and are willing to apply? Note also that this issue of manual and visual dexterity is complicated by the fact that using a function key as the means of entering and leaving LiveDoc mode means that we have made the interface a two-handed one: one for the function key and one for the mouse. This[3] raises disability and accessibility issues, and more general ease-of-use issues as well. Finally, we should note the current invisibility of the background document processing, and the need for the status of this processing to be visible to the user: How does a user know when the analysis is completed, and all the structures that can be discovered have been discovered? Some equivalent to a progress indicator bar, which reports how far along the analysis is, might be a useful addition. This may not be a problem when and if processors become fast enough to do these analyses very quickly, but we are hesitant to rely on sheer processor speed as the solution to what is really an interface design problem (especially since faster processors will most likely encourage developers to design more sophisticated, and probably more time-expensive, analyses).

### Dealing with Static-ness: APIs and Plug-Ins

A second aspect of this project addressed some deeper questions about the LiveDoc architecture: the need for an API to the LiveDoc Manager (and whether it can be eliminated), the inflexibility of the analysis architecture, and the opportunities raised by a greater degree of semantic representation about the discovered structures.

LiveSimpleText worked well as a prototype, but we were still concerned about requiring developers to change their applications to gain access to LiveDoc's capabilities. We know from experience that developers are justifiably reluctant to change their applications just to implement a new feature provided by the toolbox, so we experimented with some alternatives that we hoped would ease this restriction.

Our first approach was to modify *TextEdit*, a set of Macintosh toolbox routines that provide an application with minimal text editing and display capabilities. Although TextEdit has a limit of 32,000 characters, too small for the needs of most modern word processing applications, most application dialog boxes that have editable fields use TextEdit, and some applications also use TextEdit for their editors or text display fields. If we could "trap" the calls from an application to TextEdit and pass them through LiveDoc, we could provide LiveDoc capabilities to any application that uses TextEdit, without requiring any modification to the application. Hence, we built a Macintosh system component that trapped and patched all the TextEdit calls, and then tested this with a variety of applications. In doing this, we discovered that most applications that use TextEdit take some programming shortcuts that kept our patches from working properly. We did however, find that the e-mail client Eudora used TextEdit appropriately, and we were able to provide LiveDoc functionality to an unmodified version of Eudora through this approach. Unfortunately, before we could deploy what we called "Eudora-Live", Eudora underwent an architectural change in moving from Eudora 2.0 to Eudora 3.0, and no longer used TextEdit.

A second promising approach was to use the framework of OpenDoc[4] as a vehicle for LiveDoc. At the time we undertook this work, OpenDoc parts and part editors were just beginning to appear, and it appeared that we could leverage some of OpenDoc's extensibility to implement LiveDoc as a plug-in to the OpenDoc architecture, so that OpenDoc text parts could acquire LiveDoc behavior without any source code modification. This was promising, but we ultimately discovered that it still was not possible to get information about the way in which text was rendered without the LiveDoc Manager consulting the application.

These attempts to build an API-less LiveDoc ultimately failed (as did other attempts not discussed here, which worked with other parts of the Macintosh architecture. Nevertheless, we are still optimistic about the possibility of such approaches to system extension, for LiveDoc as well as other kinds of extensions. In retrospect, it seems that what we were trying to do was to graft an object-oriented software model onto a platform that was not object-oriented. OpenDoc had the right sense of object-orientation, but its design happened to not expose certain aspects of the system that LiveDoc required. As operating systems continue to evolve in an object-oriented direction, we may find that the kinds of extensions we sought become the norm, rather than the exception.

### Futures: Extensibility and Semantics

We tried to design LiveDoc's architecture to be as open as possible. In doing so, when faced with a tradeoff between performance and extensibility, we generally leaned towards extensibility. As implemented in the systems described here, there are two aspects to extensibility in LiveDoc: the analyzers and the kinds of structures they can discover, and the actions that can be taken on those structures. While the analyzers we built operated on lexical data, much could be done by applying the LiveDoc model to graphical information or multi-media content. We can easily envision analyzers that recognize features in drawings and pictures, or analyzers that "listen" for relevant structures in streaming audio or video. While we tried to be agnostic about these data types in the design of the LiveDoc architecture, it is inevitable that some pieces of the API would have to be rethought if such detectors became available.

The behaviors associated with actions should also be flexible and extensible, and work more closely with the document structure than they do at present. Our initial implementation of LiveDoc as LiveSimpleText assumed that actions would be handled by

3. Like the contextual menu system, it should be noted...

4. Cf. http://www.cilabs.com.

external applications, such as a Web browser presenting the page pointed to by a URL. However, other styles of interaction exist: Imagine a detector that finds the formula of an organic molecule in a document, and an action that presents a three-dimensional rendering of that molecule within the context of the document itself, rather than in a separate application. Our API allows for such behavior, but more sophisticated models of documents are needed to pursue these possibilities in detail.

Finally, LiveDoc itself knows virtually nothing of the real meaning of the items that it identifies. At best, LiveDoc might know that an object is a telephone number or a company name, but even those terms are simply labels assigned by the authors of the grammars, and have no extrinsic meaning. Moreover, there is no way to describe a composition of discovered objects in terms of meaningful relationships among them: all that the LiveDoc analysis produces is a list of discovered objects with a simple type identification. It's possible to discover relatively complex objects with the CFG model – a grammar might describe a "meeting" as a combination of a date, two times (presumably the starting and ending times of the meeting) and the name of a conference room – but there are still no semantics associated with this "meeting" object or any of its components. Hence, as noted earlier, the actions associated with a structure are static, rather than dynamically reflecting the nature of the user's interaction context. Further, we have done nothing of significance with the idea of making these objects directly manipulable; they are still only targets for a "mouse-down" interaction. These questions of direct manipulation and the marriage of syntax with semantics are important ones; we consider them in more detail in our discussion of "drop zones" [3].

## Summary

LiveDoc was a useful experiment, in several ways. It provided a testbed for exploring issues in system architecture and human interface, and how the two can – indeed, must – work together to produce a useful interaction technique. In addition, LiveDoc played a significant role in our productizing the general notion of structure detection via Apple Data Detectors. Having LiveDoc as a demo tool showed that there was a long-term technology stream at the heart of the simple notion of structure detection. The highly visual nature of the LiveDoc interface also was a very effective marketing tool: it caught the attention of the product-side people who saw it, and helped open the doors to consideration and adoption of the basic technology.

In any case, the value of the approach demonstrated by LiveDoc seems clear. More and more applications are providing some ability to recognize domain-relevant structures and allow users to execute relevant actions on them via a mouse click. Many e-mail clients automatically recognize e-mail addresses and URLs, as do some newsgroup readers and even some general-purpose document editors. This is fine with us: their domain-specific use of these techniques confirms the relevance of the general idea, and argues for a broader architectural approach to that general idea. We hope and expect that applications and the system software beneath them will continue to move in the direction outlined by LiveDoc: away from a lifeless stream of characters, and toward a collection of interrelated, meaningful, and manipulable objects. Having a clear system architecture and a well-conceived human interface, like that described here, will be essential to the success of this vision of documents, and this style of interacting with them.

## References


1. Aho, A. V., & Corasick, M. J. (1975). Efficient string matching: An aid to bibliographic search. *Communications of the ACM*, 18(6), pp. 333-340.
2. Bier, E. A., Stone, M. C., Fishkin, K., Buxton, W., & Baudel, T. (1994). A taxonomy of see-through tools. *Proceedings of CHI '94*, pp. 358-364. New York: ACM Press.
3. Bonura, T., & Miller, J. R. (1998). Drop Zones: An extension to LiveDoc. *SIGCHI Bulletin*, this volume.
4. Cypher, A. EAGER: Programming repetitive tasks by example. (1991). *Proc. CHI '91*, pp. 33-39. New York: ACM Press
5. Dey, A. K., Abowd, G. D., & Wood, A. (1998). CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services. *Proceedings of Intelligent User Interfaces '98*. New York: ACM Press.
6. Fischer, G., Lemke, A. C., Mastaglio, T., & Morch, A. (1991). Critics: An Emerging Approach to Knowledge-Based Human Computer Interaction. *International Journal of Man-Machine Studies*, 35(5), pp. 695-721.
7. Nardi, B. A., Miller, J. R., & Wright, D. J. (1998). Collaborative, programmable intelligent agents. *Communications of the ACM*, Vol. 41 No. 3, March 1998.
8. Pandit, M. & Kalbag, S. (1997). The selection recognition agent: Instant access to relevant information and operations. *Proc. Intelligent User Interfaces '97*. New York: ACM Press.


## Acknowledgments

We would like to acknowledge the contributions of Bruce Horn who was instrumental in the design of the LiveDoc API and implementation of LiveSimpleText. We also thank Laile DiSilvestro for her contributions to the OpenDoc work and implementing the TextEdit patches for EudoraLive.

## About the Authors

Jim Miller, until recently, was the program manager for Intelligent Systems in Apple's Advanced Technology Group. He is currently exploring consumer applications of Internet technology as part of *Miramontes Computing*.

Thomas Bonura is a Senior Scientist at Apple Computer, Inc. currently working on applications of speech technologies to the Macintosh user experience. His research interests are in user interface design and implementation and knowledge based systems.

## Authors' Addresses

Jim Miller
Miramontes Computing
828 Sladky Avenue, Mountain View CA 94040, USA
email: jmiller@millerclan.com
Tel: +1-650-967-2102

Thomas Bonura
Apple Computer, Inc.
1 Infinite Loop, MS 301-3KM, Cupertino CA 95014, USA
bonura@apple.com; or bonura@acm.org

# Drop Zones

## An Extension to LiveDoc

**Thomas Bonura and James R. Miller**

### Introduction

LiveDoc [6] is an extension to the Macintosh user experience that allows documents to reveal structured information in such a way that it can be readily identified and used to achieve specific actions. Various kinds of recognizers, including context-free grammars, are used to describe the structures to be found; these structures can be made up of either a single lexical term (either a variable structure like a phone number, or a collection of static strings, like company names) or multiple terms (for instance, a meeting can be defined as a combination of date, time, and venue structures). Small pieces of code can then be associated with each structure to instruct applications to carry out specific user actions on the discovered structures – perhaps to tell a telephony application to "Dial this phone number." These actions can then be offered to users by visually highlighting the discovered structures and attaching pop-up menus to the highlights. (See also [7] for an alternate interface formalism implementing the same notion of structure detection.)

This system can be very effective when working with structures that are simple and easy to recognize. However, some limitations to the approach have become clear in the time since our initial implementation of LiveDoc:

- *Limited interpretational power.* When a structure is difficult to characterize because of a high degree of variability in its form (such as the many different ways in which the information describing a meeting can be presented), the brittleness of LiveDoc's grammar-based analysis limits its effectiveness: LiveDoc may find some or all of the individual constituents of the meeting, but be unable to identify the composite structure because of where and how those constituents are located in the document. Unfortunately, this sort of structural complexity typifies much of the information for which people need computational assistance.
- *Too many choices.* There is a design tension implicit in LiveDoc: between encouraging developers to implement large numbers of actions for a given structure (so that LiveDoc can support as many user tasks as possible) and keeping the menus of actions small (so that the task of finding and selecting the desired action is kept simple). Ideally, these menus of actions should be kept small but relevant; the problem is that 'relevance' is really determined by the meaning of the context in which the user is currently working. For example, the name "Apple Computer, Inc." could be associated with such actions as, "Find the corporate headquarters on a map", "Get Apple's corporate phone number", "Get the current trading price of Apple stock", "Get the people in my address book associated with Apple" and so forth. All of these actions might be useful in one situation or another, but a user working on a financial task is far more likely to be concerned with the current price of Apple stock than the location of its corporate headquarters. Hence, the effective management of LiveDoc's action menus will come only with access to, and some understanding of, the contexts in which a user might be working.

Ultimately, both of these problems have their foundation in LiveDoc's lack of any sense of semantics of the objects on which it operates. The meaning of an object identified by LiveDoc is encapsulated within the grammar that recognizes it and the static list of actions that can be carried out on the structure: LiveDoc contains no explicit representation of this meaning. This makes it unwieldy, if not impossible, to define the mean-

ing of arbitrary sets of items or to describe anything about these structures that is not procedural. For instance, there is no way to capture abstract relations among structures, like the fact that people possess both telephone numbers and e-mail addresses.

Correcting these limitations required a different approach, one that would not only address the human interface concerns mentioned above but, more significantly, would position LiveDoc as an enabling technology for communicating with computational agents. This effort, called *Drop Zones*, is more than a new interface to LiveDoc. Rather, it is a framework centered on representing the meaning of LiveDoc objects, composing those objects might into other higher-level objects, and enabling users to take action on those compositions.

### Drop Zones

A Drop Zone provides users with an interface for managing LiveDoc objects in the context of a set of typical user tasks. Most importantly, the underlying framework for Drop Zones allows for the explicit definition of semantics about the objects, separated from the grammars that define the objects as textual strings. Once this has been done, a user interface to these objects and their semantic interpretations can be provided.

An interaction with the Drop Zone interface is shown in Figures 1 and 2. The window named 'test' in Figure 1 belongs to a LiveDoc-enabled word processor, *LiveSimpleText* (see [6]), and shows a number of structures within the document in view having been recognized by the analyzers. The window labeled *Activities* is a Drop Zone interface to a set of interpreters or 'assistants'. Each of these assistants, *E-mail*, *Telephony*, *Finance* and *Appointment*, implements a knowledge base that can operate on appropriate sets of LiveDoc structures. These assistants make their capabilities visible when the user selects various structures identified by LiveDoc and drags them to the assistants.

The important aspect of the Drop Zone interface is that it allows the user to work with the objects of interest in specific, understandable contexts. Simply working with the semantics of a set of individually meaningful objects, such as a personal name, a time and a telephone number, is too open-ended to permit much useful assistance to the user: there are too many ways in which these objects might be combined. However, thinking about the name of a person and a phone number from the perspective of making a telephone call easily leads to the interpretation, 'Call this person at this number'. Similarly, thinking about this information from the perspective of an address book easily leads to the interpretation , 'Add this person to my address book'.

Drop Zones offer a way to make these contexts a tangible part of the user interface. A Drop Zone assistant can be thought of as an interpreter that takes features identified by LiveDoc, interprets their meaning with respect to its context, and recommends appropriate actions. Consider Figure 2, in which the user has selected the structure Tom Bonura, which LiveDoc has identified with its personalName recognizer. When an object is selected, it is sent to the Drop Zone control system. Each of the assistants determines if it is able to accept and act upon the set

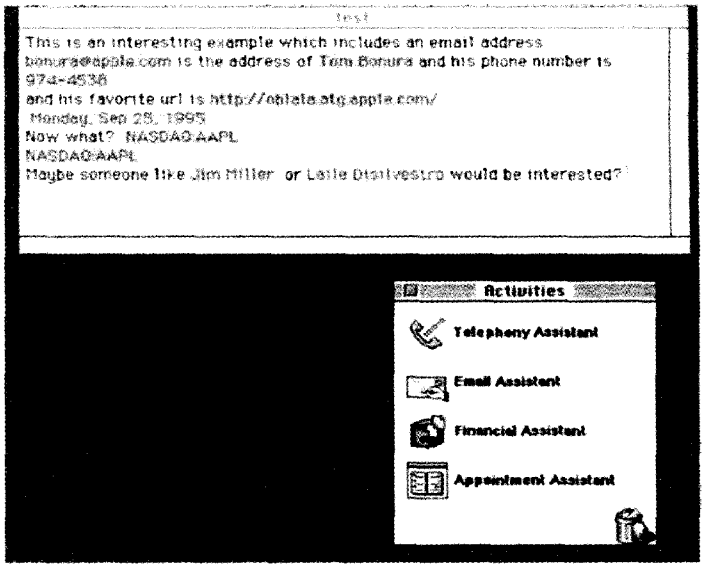



**Figure 1:** Drop zone is shown in the window labeled 'Activities'. The window at the top called 'Test' is a LiveDoc window showing proper names, e-mail addresses phone number, URL, date and stock market ticker codes

of currently selected objects. Any assistant that can do something meaningful with the objects will, when asked, highlight itself in a manner consistent with the Macintosh Drag Manager (typically, by drawing an enhanced border around itself when the mouse is above it). In this case, only the E-mail Assistant can accept the name of a person (that is, an object of type personalName), and so it indicates its availability to the user by drawing a highlight rectangle around itself when the object being dragged is over it.

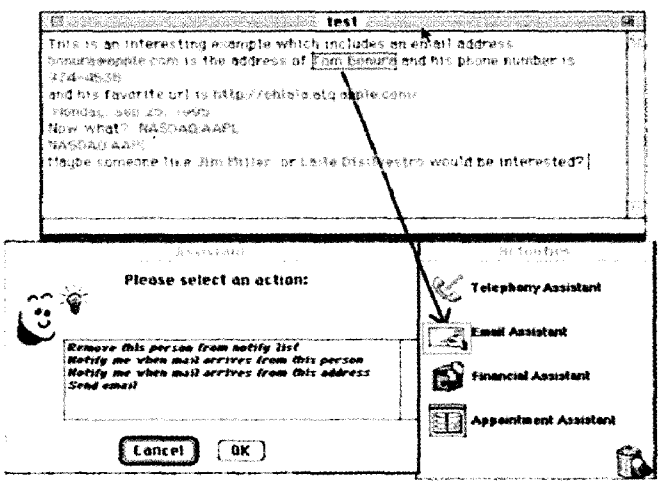



**Figure 2:** A user interaction with Drop Zones

After the user drops the name on the E-mail Assistant, a set of actions that make sense for people are presented in the *Assistant* window, and the user can choose among them. In this case, there are four possible actions, the first two of which operate on people's names and the last two of which operate on people's e-mail addresses. What is interesting in this example is that the e-mail actions are available, even though the object being offered to the assistant is a personalName. The E-mail Assistant's ability to do this is based on its access to a body of semantic information about the various types of objects present in the system, and its own design to provide e-mail-oriented assistance to the user.

Other interfaces to these assistant capabilities are, of course, possible. In particular, it is easy to imagine modifying the LiveDoc menus so that these inferred actions are presented along with those directly associated with the object clicked upon. The risk here – and the reason that we chose the alternative shown in Figure 2 – is that, as the number of selected objects and the size of the system's knowledge base increases, there may be an explosion of inferred actions expanding the size of the object's action menu beyond manageability. It's worth noting that Dey, Abowd, and Wood [3] have taken a more expansive approach to this problem, and include any action that can be inferred from any object. It will be interesting to see how well their design choice works under the circumstances described here.

### Semantics and Representation

As mentioned above, objects identified by LiveDoc have no explicit semantic value, other than their intrinsic meaning to the user of their names. It is easy to imagine the kinds of assistance that could be offered by a system that knew things like 'an important meeting is one that is called by anyone in your management chain' and 'you should always accept invitations to important meetings.' However, knowledge like this can't be stated in the simple kinds of lexical grammars used by LiveDoc. A major part of the design of Drop Zones is then to represent semantic information like this – of, about, and related to the structures being recognized by LiveDoc – in a knowledge base. The current implementation uses a hybrid knowledge representation language built around a frame-based object system, augmented by relational axioms to express facts about these objects. In this way, we keep object semantics separate from syntax (i.e., the grammars used for structure recognition in LiveDoc), and maintain the flexibility to assert facts unrelated to the actual objects found in the user's document (like the 'important meeting' example above).

When objects are selected, they are inspected by the assistants in the Drop Zone. These assistants are built around a collection of facts and axioms that determine whether and how they can operate in some meaningful way on various kinds of objects. These facts and axioms, and the inferences they permit, allow Drop Zone assistants to make much richer interpretations of structures and offer much more relevant actions than does LiveDoc's rigid model of structure grammars and static lists of actions.

Consider the situation in which a user drags a telephone number to the E-mail Assistant, presumably so that the user can send an e-mail message to the person who possesses that phone number. A literal examination of the object reveals nothing of use to the assistant: sending e-mail messages requires an e-mail address, not a phone number. However, as part of its design as an assistant for e-mail tasks, the E-mail Assistant includes an axiom that can derive e-mail addresses from phone numbers, by finding a person with the given phone number, and then obtaining the e-mail address of that person. That is, it needs to use the phone number passed to it to unify the expression:

```
(and
 (PHONE-NUMBER
    ?Person ?ThePhoneNumber)
 (e-mail-ADDRESS
    ?Person ?TheAddress))
```

and to produce the binding of the appropriate e-mail address to the variable ?TheAddress.

The problem now becomes: Where does the E-mail Assistant find the set of people whose phone numbers and e-mail addresses can be examined? The Drop Zones representational system provides two ways through which an assistant can gain access to this information. Mappings can be built between the objects inside the Drop Zones representational system (e.g., there is an object called PERSON, which has such attributes as PhoneNumber and EmailAddress) and databases or other applications. Such a mapping, in combination with a scripting language or some other programmatic way of manipulating applications, enables the E-mail Assistant to look inside an address book application for a person with the stated phone number. Another call to the address book application, guided by another mapping rule, will return the e-mail address for the identified person. Alternatively, these facts can be held directly within the Drop Zones representational system, as relations of terms, such as:

(PHONE-NUMBER 'Tom Bonura' 974-4538)

These terms can be provided by developers or, through an appropriate human interface, end-users. As a result, these bits of information can be provided to the system as they are needed, and, by preserving the assistants' knowledge bases across invocations, made available to future uses of the assistants.

Needless to say, this part of the Drop Zones work runs directly into the historical problems of knowledge representation and knowledge acquisition. The difficulty of this problem should not be underestimated, especially given its importance to Drop Zones. However, the complexity of this problem as it relates to Drop Zones is limited by the narrow scope of the assistants. The bodies of knowledge relevant to a typical assistant are small, and relatively self-contained. The representational task is then much more like that of a well-constrained expert system [e.g., 4] than a large and open-ended knowledge system [5], where the problems of representational consistency and inter-

connectedness of different knowledge units are both critical to the system's success and extremely difficult to resolve.

### Composing Terms

A representation of semantics can be useful in other ways. Consider the problem of trying to invoke an action on a collection of differing terms in a document. For instance, the sender of the e-mail message in Figure 3, based on a human's reading of the message, is clearly interested in getting together for lunch. LiveDoc has identified a number of structures in the e-mail message, which, as usual, are shown with colored highlights. However, the bits of information that make up the details of this proposed meeting are spread throughout the message, and, as a result, they fail to match the rigid syntactic 'Meeting' grammar built into LiveDoc.

Drop Zones goes beyond LiveDoc in allowing the user to select some subset of those terms and drag them as a group to the meeting assistant for interpretation. Because the Meeting Assistant contains an axiom specifying that a meeting is indicated by a date, a time, a venue, and a person's name, this collection of objects is recognized as a meeting by the Meeting Assistant. This assistant will then highlight itself when the objects are dragged over it, and an action like 'Add this meeting to your calendar' can be offered to the user via the Assistant window.

### Communicating with Agents

The concept of an 'intelligent agent' has been an active research topic in recent years, although problems still hamper the development and deployment of agent-based systems. One of the most serious problems faced by these systems is how to enable users to communicate with them. Task delegation to an agent is by its very nature an ambiguous undertaking; if it were not, we could hardly call it delegation. Users must describe their intended task to the agent, which must then devise a means of addressing it (or determine that addressing it is beyond its abilities), and communicate back to the user what it has done. Throughout this process, of course, the user needs to be aware of what the agent is about to do, what it is currently doing, and what it has just done. The user should also be able to interrupt the agent at any stage of its action, and either prevent it from taking some action or undo an action that has been taken.

To this end, many researchers have explored systems that are activated by recognizing patterns in user behavior and offering to complete the user's intended actions [e.g., 1]. Such approaches avoid direct communication with the agent, in the hope that the agent will be able to recognize an action worth taking, without explicit direction by the user. The risk here, of course, is that an agent will misrecognize or fail to recognize a pattern in the user's behavior. Other approaches use some limited form of natural language understanding as an agent communication language (e.g., [2]). The viability of this approach is limited by the complexity of the natural language understanding problem and the difficulty of extending it to all the possible problems that a user might wish to address. Users are additionally challenged by, in most cases, having a limited understanding of what kinds of actions the agent can take on

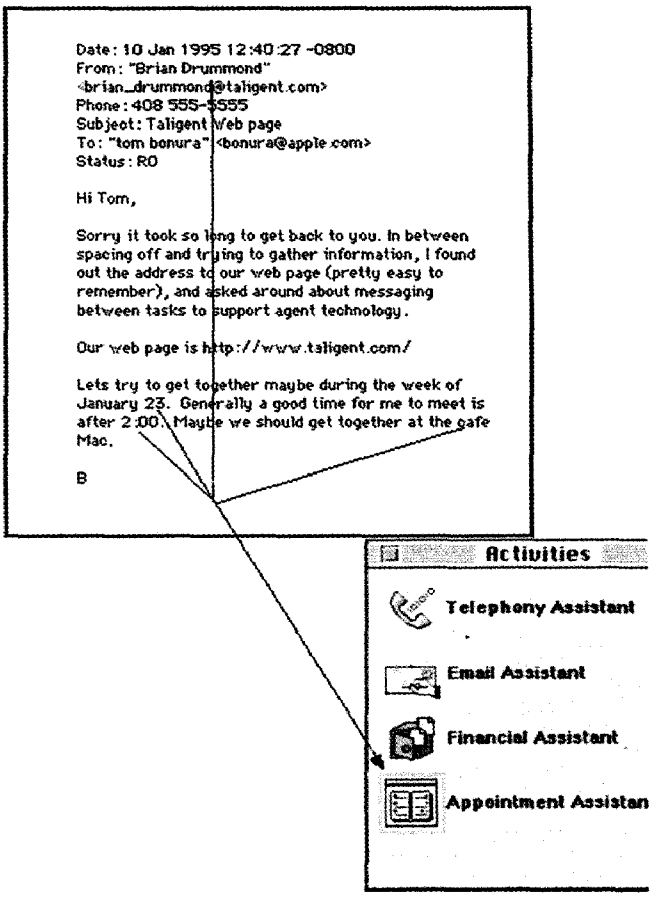



**Figure 3:** Selecting multiple items from a LiveDoc window for interpretation

their behalf, and of what kinds of language terms and constructs they can use in communicating with the agent.

We have designed the Drop Zone interface as a way of communicating intent to an agent by specifying the kinds of objects that need to be manipulated and the context in which they should be interpreted. The system provides feedback on this object selection, indicating whether it can operate on the selected items; it also provides feedback indicating what services it can provide. The ambiguity and open-endedness of a natural language interface is avoided, and a clear picture is given to the user of what services the agent can provide to the user, before those actions have been taken. Direct manipulation interfaces have often been characterized as in some way antithetical to agents (see [8]); Drop Zones shows how the interactional strengths of direct manipulation can serve as a gateway to the delegational ability of intelligent systems.

### Conclusion

The Drop Zone architecture is a powerful extension of LiveDoc's capabilities. It provides a direct manipulation interface for specifying actions to be taken on terms identified by Live-

Doc, and for selecting and confirming the actions desired by the user. Drop Zones use knowledge representation and limited inference to determine what actions users might profitably apply to various types of objects under different interaction contexts. They provide users with clear feedback indicating when an assistant can operate on the selection and what actions can be applied to the selection. What is especially significant about this approach is that, since knowledge can be added to the system dynamically, as either new facts or complex axioms, the behavior of the system is highly flexible and adaptable. We have spoken here and elsewhere [6] of the need to move to a more sophisticated model of documents, one that views documents as collections of meaningful objects rather than simple streams of characters. The combination of LiveDoc and Drop Zones offers a significant step in that direction.

## References


1. Benyon, D., and Murray, D. (1993). Developing adaptive systems to fit individual aptitudes. *Proceedings of the 1993 International Workshop on Intelligent User Interfaces.* Orlando, Florida.
2. Cohen, P., Cheyer, A., Wang, M. and Baeg, S. (1994). An open agent architecture. O. Etzioni (Ed.), *Proceedings of the AAAI Spring Symposium Series on Software Agents,* (Stanford, California, March 1994). American Association for Artificial Intelligence, p. 1-8.
3. Dey, A. K., Abowd, G. D., & Wood, A. (1998). CyberDesk: A Framework for Providing Self-Integrating Context-Aware Services. *Proceedings of Intelligent User Interfaces '98.* New York: ACM Press.
4. Hayes-Roth, F. (1983) *Building Expert Systems (Vol. 1).* New York: Addison-Wesley.
5. Lenat, D. B., & Guha, R. V. (1990). *Building Large Knowledge-Based Systems: Representation and Inference in the Cyc Project.* New York: Addison-Wesley.
6. Miller, J. R., & Bonura, T. (1998). From documents to objects: An overview of LiveDoc. *SIGCHI Bulletin,* this volume.
7. Nardi, B. A., Miller, J. R., & Wright, D. J. (1998). Collaborative, programmable intelligent agents. *Communications of the ACM,* Vol. 41, No. 3 March,1998.
8. Shneiderman, B., & Maes, P. (1997). Direct manipulation vs. interface agents. *interactions,* 4(6), pp. 42-61.


## About the Authors

Thomas Bonura is a Senior Scientist at Apple Computer, Inc. currently working on applications of speech technologies to the Macintosh user experience. His research interests are in user interface design and implementation and knowledge based systems.

Jim Miller, until recently, was the program manager for Intelligent Systems in Apple's Advanced Technology Group. He is currently exploring consumer applications of Internet technology as part of *Miramontes Computing.*

## Authors' Addresses


Thomas Bonura
Apple Computer, Inc.
1 Infinite Loop, MS 301-3KM
Cupertino CA 95014
bonura@apple.com or bonura@acm.org

Jim Miller
Miramontes Computing
828 Sladky Avenue
Mountain View CA 94040
email: jmiller@millerclan.com
Tel: +1-650-967-2102

# Exhibit 20

# Common Intents

An intent allows you to start an activity in another app by describing a simple action you'd like to perform (such as "view a map" or "take a picture") in an **`Intent`** (/reference/android/content/Intent) object. This type of intent is called an *implicit* intent because it does not specify the app component to start, but instead specifies an *action* and provides some *data* with which to perform the action.

When you call **`startActivity()`** (/reference/android/content/Context#startActivity(android.content.Intent)) or **`startActivityForResult()`** (/reference/android/app/Activity#startActivityForResult(android.content.Intent, int)) and pass it an implicit intent, the system resolves the intent (/guide/components/intents-filters#Resolution) to an app that can handle the intent and starts its corresponding **`Activity`** (/reference/android/app/Activity). If there's more than one app that can handle the intent, the system presents the user with a dialog to pick which app to use.

This page describes several implicit intents that you can use to perform common actions, organized by the type of app that handles the intent. Each section also shows how you can create an intent filter (/guide/components/intents-filters#Receiving) to advertise your app's ability to perform the same action.

**Caution:** If there are no apps on the device that can receive the implicit intent, your app will crash when it calls **`startActivity()`** (/reference/android/content/Context#startActivity(android.content.Intent)). To first verify that an app exists to receive the intent, call **`resolveActivity()`** (/reference/android/content/Intent#resolveActivity(android.content.pm.PackageManager)) on your **`Intent`** (/reference/android/content/Intent) object. If the result is non-null, there is at least one app that can handle the intent and it's safe to call **`startActivity()`** (/reference/android/content/Context#startActivity(android.content.Intent)). If the result is null, you should not use the intent and, if possible, you should disable the feature that invokes the intent.

If you're not familiar with how to create intents or intent filters, you should first read Intents and Intent Filters (/guide/components/intents-filters).

To learn how to fire the intents listed on this page from your development host, see Verify Intents with the Android Debug Bridge (#AdbIntents).

### Google Voice Actions

Google Voice Actions (https://developers.google.com/voice-actions/) fires some of the intents listed on this page in response to voice commands. For more information, see Intents fired by Google Voice Actions (https://developers.google.com/voice-actions/system/#system_actions_reference).

## Alarm Clock

### Create an alarm

To create a new alarm, use the `ACTION_SET_ALARM` (/reference/android/provider/AlarmClock#ACTION_SET_ALARM) action and specify alarm details such as the time and message using extras defined below.

(https://developers.google.cc
actions/system/#system_act

**Google Voice Actions**

- *"set an alarm for 7 am"*

**Note:** Only the hour, minutes, and message extras are available in Android 2.3 (API level 9) and lower. The other extras were added in later versions of the platform.

#### Action

`ACTION_SET_ALARM` (/reference/android/provider/AlarmClock#ACTION_SET_ALARM)

#### Data URI

None

#### MIME Type

None

**Extras**

`EXTRA_HOUR` (/reference/android/provider/AlarmClock#EXTRA_HOUR)

The hour for the alarm.

`EXTRA_MINUTES` (/reference/android/provider/AlarmClock#EXTRA_MINUTES)

The minutes for the alarm.

`EXTRA_MESSAGE` (/reference/android/provider/AlarmClock#EXTRA_MESSAGE)

A custom message to identify the alarm.

`EXTRA_DAYS` (/reference/android/provider/AlarmClock#EXTRA_DAYS)

An `ArrayList` (/reference/java/util/ArrayList) including each week day on which this alarm should be repeated. Each day must be declared with an integer from the `Calendar` (/reference/java/util/Calendar) class such as `MONDAY` (/reference/java/util/Calendar#MONDAY).

For a one-time alarm, do not specify this extra.

`EXTRA_RINGTONE` (/reference/android/provider/AlarmClock#EXTRA_RINGTONE)

A `content:` URI specifying a ringtone to use with the alarm, or `VALUE_RINGTONE_SILENT` (/reference/android/provider/AlarmClock#VALUE_RINGTONE_SILENT) for no ringtone.

To use the default ringtone, do not specify this extra.

`EXTRA_VIBRATE` (/reference/android/provider/AlarmClock#EXTRA_VIBRATE)

A boolean specifying whether to vibrate for this alarm.

**EXTRA_SKIP_UI** (/reference/android/provider/AlarmClock#EXTRA_SKIP_UI)

A boolean specifying whether the responding app should skip its UI when setting the alarm. If true, the app should bypass any confirmation UI and simply set the specified alarm.

**Example intent:**

KOTLIN (#KOTLIN)JAVA

```
public void createAlarm(String message, int hour, int minutes) {
    Intent intent = new Intent(AlarmClock.ACTION_SET_ALARM)
            .putExtra(AlarmClock.EXTRA_MESSAGE, message)
            .putExtra(AlarmClock.EXTRA_HOUR, hour)
            .putExtra(AlarmClock.EXTRA_MINUTES, minutes);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

**Note:**

In order to invoke the **ACTION_SET_ALARM** (/reference/android/provider/AlarmClock#ACTION_SET_ALARM) intent, your app must have the **SET_ALARM** (/reference/android/Manifest.permission#SET_ALARM) permission:

```
<uses-permission android:name="com.android.alarm.permission.SET_ALARM" />
```

**Example intent filter:**

```
<activity ...>
    <intent-filter>
        <action android:name="android.intent.action.SET_ALARM" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

## Create a timer

To create a countdown timer, use the `ACTION_SET_TIMER` (/reference/android/provider/AlarmClock#ACTION_SET_TIMER) action and specify timer details such as the duration using extras defined below.

(https://developers.google.co
actions/system/#system_act

**Google Voice Actions**

- *"set timer for 5 minutes"*

**Note:** This intent was added in Android 4.4 (API level 19).

### Action

`ACTION_SET_TIMER` (/reference/android/provider/AlarmClock#ACTION_SET_TIMER)

### Data URI

None

### MIME Type

None

### Extras

**EXTRA_LENGTH** (/reference/android/provider/AlarmClock#EXTRA_LENGTH)

The length of the timer in seconds.

**EXTRA_MESSAGE** (/reference/android/provider/AlarmClock#EXTRA_MESSAGE)

A custom message to identify the timer.

**EXTRA_SKIP_UI** (/reference/android/provider/AlarmClock#EXTRA_SKIP_UI)

A boolean specifying whether the responding app should skip its UI when setting the timer. If true, the app should bypass any confirmation UI and simply start the specified timer.

**Example intent:**

KOTLIN (#KOTLIN)JAVA

```
public void startTimer(String message, int seconds) {
    Intent intent = new Intent(AlarmClock.ACTION_SET_TIMER)
            .putExtra(AlarmClock.EXTRA_MESSAGE, message)
            .putExtra(AlarmClock.EXTRA_LENGTH, seconds)
            .putExtra(AlarmClock.EXTRA_SKIP_UI, true);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

**Note:**

In order to invoke the **ACTION_SET_TIMER** (/reference/android/provider/AlarmClock#ACTION_SET_TIMER) intent, your app must have the **SET_ALARM** (/reference/android/Manifest.permission#SET_ALARM) permission:

ARENDI_G329277

```
<uses-permission android:name="com.android.alarm.permission.SET_ALARM" />
```

**Example intent filter:**

```
<activity ...>
    <intent-filter>
        <action android:name="android.intent.action.SET_TIMER" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

## Show all alarms

To show the list of alarms, use the `ACTION_SHOW_ALARMS` (/reference/android/provider/AlarmClock#ACTION_SHOW_ALARMS) action.

Although not many apps will invoke this intent (it's primarily used by system apps), any app that behaves as an alarm clock should implement this intent filter and respond by showing the list of current alarms.

**Note:** This intent was added in Android 4.4 (API level 19).

### Action

`ACTION_SHOW_ALARMS` (/reference/android/provider/AlarmClock#ACTION_SHOW_ALARMS)

### Data URI

None

**MIME Type**

None

**Example intent filter:**

```
<activity ...>
    <intent-filter>
        <action android:name="android.intent.action.SHOW_ALARMS" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

# Calendar

## Add a calendar event

To add a new event to the user's calendar, use the `ACTION_INSERT` (/reference/android/content/Intent#ACTION_INSERT) action and specify the data URI with `Events.CONTENT_URI` (/reference/android/provider/CalendarContract.Events#CONTENT_URI). You can then specify various event details using extras defined below.

**Action**

`ACTION_INSERT` (/reference/android/content/Intent#ACTION_INSERT)

**Data URI**

`Events.CONTENT_URI` (/reference/android/provider/CalendarContract.Events#CONTENT_URI)

**MIME Type**

`"vnd.android.cursor.dir/event"`

**Extras**

`EXTRA_EVENT_ALL_DAY` (/reference/android/provider/CalendarContract#EXTRA_EVENT_ALL_DAY)

A boolean specifying whether this is an all-day event.

`EXTRA_EVENT_BEGIN_TIME` (/reference/android/provider/CalendarContract#EXTRA_EVENT_BEGIN_TIME)

The start time of the event (milliseconds since epoch).

`EXTRA_EVENT_END_TIME` (/reference/android/provider/CalendarContract#EXTRA_EVENT_END_TIME)

The end time of the event (milliseconds since epoch).

`TITLE` (/reference/android/provider/CalendarContract.EventsColumns#TITLE)

The event title.

`DESCRIPTION` (/reference/android/provider/CalendarContract.EventsColumns#DESCRIPTION)

The event description.

`EVENT_LOCATION` (/reference/android/provider/CalendarContract.EventsColumns#EVENT_LOCATION)

The event location.

`EXTRA_EMAIL` (/reference/android/content/Intent#EXTRA_EMAIL)

A comma-separated list of email addresses that specify the invitees.

Many more event details can be specified using the constants defined in the `CalendarContract.EventsColumns` (/reference/android/provider/CalendarContract.EventsColumns) class.

**Example intent:**

KOTLIN (#KOTLIN)JAVA

```
public void addEvent(String title, String location, long begin, long end) {
    Intent intent = new Intent(Intent.ACTION_INSERT)
            .setData(Events.CONTENT_URI)
            .putExtra(Events.TITLE, title)
            .putExtra(Events.EVENT_LOCATION, location)
            .putExtra(CalendarContract.EXTRA_EVENT_BEGIN_TIME, begin)
            .putExtra(CalendarContract.EXTRA_EVENT_END_TIME, end);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

**Example intent filter:**

```
<activity ...>
    <intent-filter>
        <action android:name="android.intent.action.INSERT" />
        <data android:mimeType="vnd.android.cursor.dir/event" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

# Camera

## Capture a picture or video and return it

To open a camera app and receive the resulting photo or video, use the `ACTION_IMAGE_CAPTURE` (/reference/android/provider/MediaStore#ACTION_IMAGE_CAPTURE) or `ACTION_VIDEO_CAPTURE` (/reference/android/provider/MediaStore#ACTION_VIDEO_CAPTURE) action. Also specify the URI location where you'd like the camera to save the photo or video, in the `EXTRA_OUTPUT` (/reference/android/provider/MediaStore#EXTRA_OUTPUT) extra.

**Action**

`ACTION_IMAGE_CAPTURE` (/reference/android/provider/MediaStore#ACTION_IMAGE_CAPTURE) or
`ACTION_VIDEO_CAPTURE` (/reference/android/provider/MediaStore#ACTION_VIDEO_CAPTURE)

**Data URI Scheme**

None

**MIME Type**

None

**Extras**

`EXTRA_OUTPUT` (/reference/android/provider/MediaStore#EXTRA_OUTPUT)

The URI location where the camera app should save the photo or video file (as a `Uri` (/reference/android/net/Uri) object).

When the camera app successfully returns focus to your activity (your app receives the `onActivityResult()` (/reference/android/app/Activity#onActivityResult(int, int, android.content.Intent)) callback), you can access the photo or video at the URI you specified with the `EXTRA_OUTPUT` (/reference/android/provider/MediaStore#EXTRA_OUTPUT) value.

**Note:** When you use `ACTION_IMAGE_CAPTURE` (/reference/android/provider/MediaStore#ACTION_IMAGE_CAPTURE) to capture a photo, the camera may also return a downscaled copy (a thumbnail) of the photo in the result `Intent` (/reference/android/content/Intent), saved as a `Bitmap`

(/reference/android/graphics/Bitmap) in an extra field named "**data**".

**Example intent:**

KOTLIN (#KOTLIN)JAVA

```
static final int REQUEST_IMAGE_CAPTURE = 1;
static final Uri locationForPhotos;

public void capturePhoto(String targetFilename) {
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    intent.putExtra(MediaStore.EXTRA_OUTPUT,
            Uri.withAppendedPath(locationForPhotos, targetFilename));
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(intent, REQUEST_IMAGE_CAPTURE);
    }
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_IMAGE_CAPTURE && resultCode == RESULT_OK) {
        Bitmap thumbnail = data.getParcelableExtra("data");
        // Do other work with full size photo saved in locationForPhotos
        ...
    }
}
```

For more information about how to use this intent to capture a photo, including how to create an appropriate `Uri` (/reference/android/net/Uri) for the output location, read Taking Photos Simply (/training/camera/photobasics) or Taking Videos Simply (/training/camera/videobasics).

**Example intent filter:**

```
<activity ...>
    <intent-filter>
        <action android:name="android.media.action.IMAGE_CAPTURE" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

When handling this intent, your activity should check for the `EXTRA_OUTPUT` (/reference/android/provider/MediaStore#EXTRA_OUTPUT) extra in the incoming `Intent` (/reference/android/content/Intent), then save the captured image or video at the location specified by that extra and call `setResult()` (/reference/android/app/Activity#setResult(int, android.content.Intent)) with an `Intent` (/reference/android/content/Intent) that includes a compressed thumbnail in an extra named `"data"`.

## Start a camera app in still image mode

To open a camera app in still image mode, use the `INTENT_ACTION_STILL_IMAGE_CAMERA` (/reference/android/provider/MediaStore#INTENT_ACTION_STILL_IMAGE_CAMERA) action.

**Action**

`INTENT_ACTION_STILL_IMAGE_CAMERA` (/reference/android/provider/MediaStore#INTENT_ACTION_STILL_IMAGE_CAMERA)

**Data URI Scheme**

None

**MIME Type**

(https://developers.google.cc
actions/system/#system_act

**Google Voice Actions**

- *"take a picture"*

None

**Extras**

None

**Example intent:**

KOTLIN (#KOTLIN)JAVA

```
public void capturePhoto() {
    Intent intent = new Intent(MediaStore.INTENT_ACTION_STILL_IMAGE_CAMERA);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(intent, REQUEST_IMAGE_CAPTURE);
    }
}
```

**Example intent filter:**

```
<activity ...>
    <intent-filter>
        <action android:name="android.media.action.STILL_IMAGE_CAMERA" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

## Start a camera app in video mode

To open a camera app in video mode, use the **INTENT_ACTION_VIDEO_CAMERA** (/reference/android/provider/MediaStore#INTENT_ACTION_VIDEO_CAMERA) action.

(https://developers.google.co
actions/system/#system_act

**Google Voice Actions**

- *"record a video"*

### Action

**INTENT_ACTION_VIDEO_CAMERA** (/reference/android/provider/MediaStore#INTENT_ACTION_VIDEO_CAMERA)

### Data URI Scheme

None

### MIME Type

None

### Extras

None

### Example intent:

KOTLIN (#KOTLIN)JAVA

```
public void capturePhoto() {
    Intent intent = new Intent(MediaStore.INTENT_ACTION_VIDEO_CAMERA);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(intent, REQUEST_IMAGE_CAPTURE);
    }
}
```

### Example intent filter:

ARENDI_G329286

```
<activity ...>
    <intent-filter>
        <action android:name="android.media.action.VIDEO_CAMERA" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

# Contacts/People App

## Select a contact

To have the user select a contact and provide your app access to all the contact information, use the `ACTION_PICK` (/reference/android/content/Intent#ACTION_PICK) action and specify the MIME type to `Contacts.CONTENT_TYPE` (/reference/android/provider/ContactsContract.Contacts#CONTENT_TYPE).

The result `Intent` (/reference/android/content/Intent) delivered to your `onActivityResult()` (/reference/android/app/Activity#onActivityResult(int, int, android.content.Intent)) callback contains the `content:` URI pointing to the selected contact. The response grants your app temporary permissions to read that contact using the Contacts Provider (/guide/topics/providers/contacts-provider) API even if your app does not include the `READ_CONTACTS` (/reference/android/Manifest.permission#READ_CONTACTS) permission.

**Tip:** If you need access to only a specific piece of contact information, such as a phone number or email address, instead see the next section about how to select specific contact data (#PickContactData).

### Action

`ACTION_PICK` (/reference/android/content/Intent#ACTION_PICK)

### Data URI Scheme

None

### MIME Type

`Contacts.CONTENT_TYPE` (/reference/android/provider/ContactsContract.Contacts#CONTENT_TYPE)

### Example intent:

KOTLIN (#KOTLIN)JAVA

```
static final int REQUEST_SELECT_CONTACT = 1;

public void selectContact() {
    Intent intent = new Intent(Intent.ACTION_PICK);
    intent.setType(ContactsContract.Contacts.CONTENT_TYPE);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(intent, REQUEST_SELECT_CONTACT);
    }
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_SELECT_CONTACT && resultCode == RESULT_OK) {
        Uri contactUri = data.getData();
        // Do something with the selected contact at contactUri
        ...
    }
}
```

For information about how to retrieve contact details once you have the contact URI, read Retrieving Details for a Contact (/training/contacts-provider/retrieve-details).

When you retrieve the contact URI using the above intent, you generally don't need the `READ_CONTACTS` (/reference/android/Manifest.permission#READ_CONTACTS) permission to read basic details for that contact, such as display name and whether the contact is starred. However, if you're trying to read more specific data (/guide/components/PickContactDat) about a given contact—such as their phone number or email address—you need the `READ_CONTACTS` permission.

## Select specific contact data

To have the user select a specific piece of information from a contact, such as a phone number, email address, or other data type, use the `ACTION_PICK` (/reference/android/content/Intent#ACTION_PICK) action and specify the MIME type to one of the content types listed below, such as `CommonDataKinds.Phone.CONTENT_TYPE` (/reference/android/provider/ContactsContract.CommonDataKinds.Phone#CONTENT_TYPE) to get the contact's phone number.

**Note:** In many cases, your app needs to have the `READ_CONTACTS` (/reference/android/Manifest.permission#READ_CONTACTS) permission in order to view specific information about a particular contact.

If you need to retrieve only one type of data from a contact, this technique with a `CONTENT_TYPE` from the `ContactsContract.CommonDataKinds` (/reference/android/provider/ContactsContract.CommonDataKinds) classes is more efficient than using the `Contacts.CONTENT_TYPE` (/reference/android/provider/ContactsContract.Contacts#CONTENT_TYPE) (as shown in the previous section) because the result provides you direct access to the desired data without requiring you to perform a more complex query to Contacts Provider (/guide/topics/providers/contacts-provider).

The result `Intent` (/reference/android/content/Intent) delivered to your `onActivityResult()` (/reference/android/app/Activity#onActivityResult(int, int, android.content.Intent)) callback contains the `content:` URI pointing to the selected contact data. The response grants your app temporary permissions to read that contact data even if your app does not include the `READ_CONTACTS` (/reference/android/Manifest.permission#READ_CONTACTS) permission.

### Action

`ACTION_PICK` (/reference/android/content/Intent#ACTION_PICK)

### Data URI Scheme

None

### MIME Type

`CommonDataKinds.Phone.CONTENT_TYPE` (/reference/android/provider/ContactsContract.CommonDataKinds.Phone#CONTENT_TYPE)

Pick from contacts with a phone number.

`CommonDataKinds.Email.CONTENT_TYPE` (/reference/android/provider/ContactsContract.CommonDataKinds.Email#CONTENT_TYPE)

Pick from contacts with an email address.

`CommonDataKinds.StructuredPostal.CONTENT_TYPE`
(/reference/android/provider/ContactsContract.CommonDataKinds.StructuredPostal#CONTENT_TYPE)

Pick from contacts with a postal address.

Or one of many other `CONTENT_TYPE` values under `ContactsContract` (/reference/android/provider/ContactsContract).

### Example intent:

KOTLIN (#KOTLIN)JAVA

```
static final int REQUEST_SELECT_PHONE_NUMBER = 1;

public void selectContact() {
    // Start an activity for the user to pick a phone number from contacts
```

```
    Intent intent = new Intent(Intent.ACTION_PICK);
    intent.setType(CommonDataKinds.Phone.CONTENT_TYPE);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(intent, REQUEST_SELECT_PHONE_NUMBER);
    }
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_SELECT_PHONE_NUMBER && resultCode == RESULT_OK) {
        // Get the URI and query the content provider for the phone number
        Uri contactUri = data.getData();
        String[] projection = new String[]{CommonDataKinds.Phone.NUMBER};
        Cursor cursor = getContentResolver().query(contactUri, projection,
                null, null, null);
        // If the cursor returned is valid, get the phone number
        if (cursor != null && cursor.moveToFirst()) {
            int numberIndex = cursor.getColumnIndex(CommonDataKinds.Phone.NUMBER);
            String number = cursor.getString(numberIndex);
            // Do something with the phone number
            //...
        }
    }
}
```

## View a contact

To display the details for a known contact, use the **`ACTION_VIEW`** (/reference/android/content/Intent#ACTION_VIEW) action and specify the contact with a **`content:`** URI as the intent data.

There are primarily two ways to initially retrieve the contact's URI:

- Use the contact URI returned by the **ACTION_PICK** (/reference/android/content/Intent#ACTION_PICK), shown in the previous section (this approach does not require any app permissions).
- Access the list of all contacts directly, as described in Retrieving a List of Contacts (/training/contacts-provider/retrieve-names) (this approach requires the **READ_CONTACTS** (/reference/android/Manifest.permission#READ_CONTACTS) permission).

**Action**

**ACTION_VIEW** (/reference/android/content/Intent#ACTION_VIEW)

**Data URI Scheme**

`content:<URI>`

**MIME Type**

None. The type is inferred from contact URI.

**Example intent:**

KOTLIN (#KOTLIN)JAVA

```
public void viewContact(Uri contactUri) {
    Intent intent = new Intent(Intent.ACTION_VIEW, contactUri);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

## Edit an existing contact

To edit a known contact, use the `ACTION_EDIT` (/reference/android/content/Intent#ACTION_EDIT) action, specify the contact with a `content:` URI as the intent data, and include any known contact information in extras specified by constants in `ContactsContract.Intents.Insert` (/reference/android/provider/ContactsContract.Intents.Insert).

There are primarily two ways to initially retrieve the contact URI:

- Use the contact URI returned by the `ACTION_PICK` (/reference/android/content/Intent#ACTION_PICK), shown in the previous section (this approach does not require any app permissions).
- Access the list of all contacts directly, as described in Retrieving a List of Contacts (/training/contacts-provider/retrieve-names) (this approach requires the `READ_CONTACTS` (/reference/android/Manifest.permission#READ_CONTACTS) permission).

**Action**

`ACTION_EDIT` (/reference/android/content/Intent#ACTION_EDIT)

**Data URI Scheme**

`content:<URI>`

**MIME Type**

The type is inferred from contact URI.

**Extras**

One or more of the extras defined in `ContactsContract.Intents.Insert` (/reference/android/provider/ContactsContract.Intents.Insert) so you can populate fields of the contact details.

**Example intent:**

KOTLIN (#KOTLIN)JAVA

ARENDI_G329293

```
public void editContact(Uri contactUri, String email) {
    Intent intent = new Intent(Intent.ACTION_EDIT);
    intent.setData(contactUri);
    intent.putExtra(Intents.Insert.EMAIL, email);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

For more information about how to edit a contact, read Modifying Contacts Using Intents (/training/contacts-provider/modify-data).

## Insert a contact

To insert a new contact, use the `ACTION_INSERT` (/reference/android/content/Intent#ACTION_INSERT) action, specify `Contacts.CONTENT_TYPE` (/reference/android/provider/ContactsContract.Contacts#CONTENT_TYPE) as the MIME type, and include any known contact information in extras specified by constants in `ContactsContract.Intents.Insert` (/reference/android/provider/ContactsContract.Intents.Insert).

**Action**

`ACTION_INSERT` (/reference/android/content/Intent#ACTION_INSERT)

**Data URI Scheme**

None

**MIME Type**

`Contacts.CONTENT_TYPE` (/reference/android/provider/ContactsContract.Contacts#CONTENT_TYPE)

**Extras**

One or more of the extras defined in `ContactsContract.Intents.Insert` (/reference/android/provider/ContactsContract.Intents.Insert).

**Example intent:**

KOTLIN (#KOTLIN) JAVA

```
public void insertContact(String name, String email) {
    Intent intent = new Intent(Intent.ACTION_INSERT);
    intent.setType(Contacts.CONTENT_TYPE);
    intent.putExtra(Intents.Insert.NAME, name);
    intent.putExtra(Intents.Insert.EMAIL, email);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

For more information about how to insert a contact, read Modifying Contacts Using Intents (/training/contacts-provider/modify-data).

# Email

## Compose an email with optional attachments

To compose an email, use one of the below actions based on whether you'll include attachments, and include email details such as the recipient and subject using the extra keys listed below.

**Action**

`ACTION_SENDTO` (/reference/android/content/Intent#ACTION_SENDTO) (for no attachment) or
`ACTION_SEND` (/reference/android/content/Intent#ACTION_SEND) (for one attachment) or
`ACTION_SEND_MULTIPLE` (/reference/android/content/Intent#ACTION_SEND_MULTIPLE) (for multiple attachments)

### Data URI Scheme

None

### MIME Type

`"text/plain"`

`"*/*"`

### Extras

`Intent.EXTRA_EMAIL` (/reference/android/content/Intent#EXTRA_EMAIL)

A string array of all "To" recipient email addresses.

`Intent.EXTRA_CC` (/reference/android/content/Intent#EXTRA_CC)

A string array of all "CC" recipient email addresses.

`Intent.EXTRA_BCC` (/reference/android/content/Intent#EXTRA_BCC)

A string array of all "BCC" recipient email addresses.

`Intent.EXTRA_SUBJECT` (/reference/android/content/Intent#EXTRA_SUBJECT)

A string with the email subject.

`Intent.EXTRA_TEXT` (/reference/android/content/Intent#EXTRA_TEXT)

A string with the body of the email.

`Intent.EXTRA_STREAM` (/reference/android/content/Intent#EXTRA_STREAM)

A `Uri` (/reference/android/net/Uri) pointing to the attachment. If using the `ACTION_SEND_MULTIPLE` (/reference/android/content/Intent#ACTION_SEND_MULTIPLE) action, this should instead be an `ArrayList` (/reference/java/util/ArrayList) containing multiple `Uri` (/reference/android/net/Uri) objects.

**Example intent:**

KOTLIN (#KOTLIN)JAVA

```
public void composeEmail(String[] addresses, String subject, Uri attachment) {
    Intent intent = new Intent(Intent.ACTION_SEND);
    intent.setType("*/*");
    intent.putExtra(Intent.EXTRA_EMAIL, addresses);
    intent.putExtra(Intent.EXTRA_SUBJECT, subject);
    intent.putExtra(Intent.EXTRA_STREAM, attachment);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

If you want to ensure that your intent is handled only by an email app (and not other text messaging or social apps), then use the `ACTION_SENDTO` (/reference/android/content/Intent#ACTION_SENDTO) action and include the `"mailto:"` data scheme. For example:

KOTLIN (#KOTLIN)JAVA

```
public void composeEmail(String[] addresses, String subject) {
    Intent intent = new Intent(Intent.ACTION_SENDTO);
```

ARENDI_G329297

```
    intent.setData(Uri.parse("mailto:")); // only email apps should handle this
    intent.putExtra(Intent.EXTRA_EMAIL, addresses);
    intent.putExtra(Intent.EXTRA_SUBJECT, subject);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

**Example intent filter:**

```
<activity ...>
    <intent-filter>
        <action android:name="android.intent.action.SEND" />
        <data android:type="*/*" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
    <intent-filter>
        <action android:name="android.intent.action.SENDTO" />
        <data android:scheme="mailto" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

# File Storage

## Retrieve a specific type of file

To request that the user select a file such as a document or photo and return a reference to your app, use the **`ACTION_GET_CONTENT`** (/reference/android/content/Intent#ACTION_GET_CONTENT) action and specify your desired MIME type. The file reference returned to your

app is transient to your activity's current lifecycle, so if you want to access it later you must import a copy that you can read later. This intent also allows the user to create a new file in the process (for example, instead of selecting an existing photo, the user can capture a new photo with the camera).

The result intent delivered to your `onActivityResult()` (/reference/android/app/Activity#onActivityResult(int, int, android.content.Intent)) method includes data with a URI pointing to the file. The URI could be anything, such as an `http:` URI, `file:` URI, or `content:` URI. However, if you'd like to restrict selectable files to only those that are accessible from a content provider (a `content:` URI) and that are available as a file stream with `openFileDescriptor()` (/reference/android/content/ContentResolver#openFileDescriptor(android.net.Uri, java.lang.String)), you should add the `CATEGORY_OPENABLE` (/reference/android/content/Intent#CATEGORY_OPENABLE) category to your intent.

On Android 4.3 (API level 18) and higher, you can also allow the user to select multiple files by adding `EXTRA_ALLOW_MULTIPLE` (/reference/android/content/Intent#EXTRA_ALLOW_MULTIPLE) to the intent, set to `true`. You can then access each of the selected files in a `ClipData` (/reference/android/content/ClipData) object returned by `getClipData()` (/reference/android/content/Intent#getClipData()).

**Action**

`ACTION_GET_CONTENT` (/reference/android/content/Intent#ACTION_GET_CONTENT)

**Data URI Scheme**

None

**MIME Type**

The MIME type corresponding to the file type the user should select.

**Extras**

`EXTRA_ALLOW_MULTIPLE` (/reference/android/content/Intent#EXTRA_ALLOW_MULTIPLE)

A boolean declaring whether the user can select more than one file at a time.

`EXTRA_LOCAL_ONLY` (/reference/android/content/Intent#EXTRA_LOCAL_ONLY)

A boolean that declares whether the returned file must be available directly from the device, rather than requiring a download from a remote service.

**Category (optional)**

`CATEGORY_OPENABLE` (/reference/android/content/Intent#CATEGORY_OPENABLE)

To return only "openable" files that can be represented as a file stream with `openFileDescriptor()` (/reference/android/content/ContentResolver#openFileDescriptor(android.net.Uri, java.lang.String)).

**Example intent to get a photo:**

KOTLIN (#KOTLIN)JAVA

```
static final int REQUEST_IMAGE_GET = 1;

public void selectImage() {
    Intent intent = new Intent(Intent.ACTION_GET_CONTENT);
    intent.setType("image/*");
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(intent, REQUEST_IMAGE_GET);
    }
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_IMAGE_GET && resultCode == RESULT_OK) {
        Bitmap thumbnail = data.getParcelable("data");
        Uri fullPhotoUri = data.getData();
        // Do work with photo saved at fullPhotoUri
        ...
```

```
    }
}
```

**Example intent filter to return a photo:**

```
<activity ...>
    <intent-filter>
        <action android:name="android.intent.action.GET_CONTENT" />
        <data android:type="image/*" />
        <category android:name="android.intent.category.DEFAULT" />
        <!-- The OPENABLE category declares that the returned file is accessible
             from a content provider that supports OpenableColumns (/reference/android/provider/OpenableColumns)
             and ContentResolver.openFileDescriptor() (/reference/android/content/ContentResolver#openFileDescriptor(android.net.Ur
        <category android:name="android.intent.category.OPENABLE" />
    </intent-filter>
</activity>
```

## Open a specific type of file

Instead of retrieving a copy of a file that you must import to your app (by using the `ACTION_GET_CONTENT` (/reference/android/content/Intent#ACTION_GET_CONTENT) action), when running on Android 4.4 or higher, you can instead request to *open* a file that's managed by another app by using the `ACTION_OPEN_DOCUMENT` (/reference/android/content/Intent#ACTION_OPEN_DOCUMENT) action and specifying a MIME type. To also allow the user to instead create a new document that your app can write to, use the `ACTION_CREATE_DOCUMENT` (/reference/android/content/Intent#ACTION_CREATE_DOCUMENT) action instead. For example, instead of selecting from existing PDF documents, the `ACTION_CREATE_DOCUMENT` (/reference/android/content/Intent#ACTION_CREATE_DOCUMENT) intent allows users to select where they'd like to create a new document (within another app that manages the document's storage)—your app then receives the URI location of where it can write the new document.

Whereas the intent delivered to your `onActivityResult()` (/reference/android/app/Activity#onActivityResult(int, int, android.content.Intent)) method from the `ACTION_GET_CONTENT` (/reference/android/content/Intent#ACTION_GET_CONTENT) action may return a URI of any type, the result intent from `ACTION_OPEN_DOCUMENT` (/reference/android/content/Intent#ACTION_OPEN_DOCUMENT) and `ACTION_CREATE_DOCUMENT` (/reference/android/content/Intent#ACTION_CREATE_DOCUMENT) always specify the chosen file as a `content:` URI that's backed by a `DocumentsProvider` (/reference/android/provider/DocumentsProvider). You can open the file with `openFileDescriptor()` (/reference/android/content/ContentResolver#openFileDescriptor(android.net.Uri, java.lang.String)) and query its details using columns from `DocumentsContract.Document` (/reference/android/provider/DocumentsContract.Document).

The returned URI grants your app long-term read access to the file (also possibly with write access). So the `ACTION_OPEN_DOCUMENT` (/reference/android/content/Intent#ACTION_OPEN_DOCUMENT) action is particularly useful (instead of using `ACTION_GET_CONTENT` (/reference/android/content/Intent#ACTION_GET_CONTENT)) when you want to read an existing file without making a copy into your app, or when you want to open and edit a file in place.

You can also allow the user to select multiple files by adding `EXTRA_ALLOW_MULTIPLE` (/reference/android/content/Intent#EXTRA_ALLOW_MULTIPLE) to the intent, set to `true`. If the user selects just one item, then you can retrieve the item from `getData()` (/reference/android/content/Intent#getData()). If the user selects more than one item, then `getData()` (/reference/android/content/Intent#getData()) returns null and you must instead retrieve each item from a `ClipData` (/reference/android/content/ClipData) object that is returned by `getClipData()` (/reference/android/content/Intent#getClipData()).

**Note:** Your intent **must** specify a MIME type and **must** declare the `CATEGORY_OPENABLE` (/reference/android/content/Intent#CATEGORY_OPENABLE) category. If appropriate, you can specify more than one MIME type by adding an array of MIME types with the `EXTRA_MIME_TYPES` (/reference/android/content/Intent#EXTRA_MIME_TYPES) extra—if you do so, you must set the primary MIME type in `setType()` (/reference/android/content/Intent#setType(java.lang.String)) to "`*/*`".

### Action

`ACTION_OPEN_DOCUMENT` (/reference/android/content/Intent#ACTION_OPEN_DOCUMENT) or
`ACTION_CREATE_DOCUMENT` (/reference/android/content/Intent#ACTION_CREATE_DOCUMENT)

### Data URI Scheme

None

### MIME Type

The MIME type corresponding to the file type the user should select.

### Extras

`EXTRA_MIME_TYPES` (/reference/android/content/Intent#EXTRA_MIME_TYPES)

An array of MIME types corresponding to the types of files your app is requesting. When you use this extra, you must set the primary MIME type in `setType()` (/reference/android/content/Intent#setType(java.lang.String)) to `"*/*"`.

`EXTRA_ALLOW_MULTIPLE` (/reference/android/content/Intent#EXTRA_ALLOW_MULTIPLE)

A boolean that declares whether the user can select more than one file at a time.

`EXTRA_TITLE` (/reference/android/content/Intent#EXTRA_TITLE)

For use with `ACTION_CREATE_DOCUMENT` (/reference/android/content/Intent#ACTION_CREATE_DOCUMENT) to specify an initial file name.

`EXTRA_LOCAL_ONLY` (/reference/android/content/Intent#EXTRA_LOCAL_ONLY)

A boolean that declares whether the returned file must be available directly from the device, rather than requiring a download from a remote service.

### Category

`CATEGORY_OPENABLE` (/reference/android/content/Intent#CATEGORY_OPENABLE)

To return only "openable" files that can be represented as a file stream with `openFileDescriptor()` (/reference/android/content/ContentResolver#openFileDescriptor(android.net.Uri, java.lang.String)).

**Example intent to get a photo:**

KOTLIN (#KOTLIN)JAVA

```
static final int REQUEST_IMAGE_OPEN = 1;

public void selectImage() {
    Intent intent = new Intent(Intent.ACTION_OPEN_DOCUMENT);
    intent.setType("image/*");
    intent.addCategory(Intent.CATEGORY_OPENABLE);
    // Only the system receives the ACTION_OPEN_DOCUMENT, so no need to test.
    startActivityForResult(intent, REQUEST_IMAGE_OPEN);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == REQUEST_IMAGE_OPEN && resultCode == RESULT_OK) {
        Uri fullPhotoUri = data.getData();
        // Do work with full size photo saved at fullPhotoUri
        ...
    }
}
```

Third party apps cannot actually respond to an intent with the `ACTION_OPEN_DOCUMENT` (/reference/android/content/Intent#ACTION_OPEN_DOCUMENT) action. Instead, the system receives this intent and displays all the files available from various apps in a unified user interface.

To provide your app's files in this UI and allow other apps to open them, you must implement a `DocumentsProvider` (/reference/android/provider/DocumentsProvider) and include an intent filter for `PROVIDER_INTERFACE` (/reference/android/provider/DocumentsContract#PROVIDER_INTERFACE) (`"android.content.action.DOCUMENTS_PROVIDER"`). For example:

ARENDI_G329304

```
<provider ...
    android:grantUriPermissions="true"
    android:exported="true"
    android:permission="android.permission.MANAGE_DOCUMENTS">
    <intent-filter>
        <action android:name="android.content.action.DOCUMENTS_PROVIDER" />
    </intent-filter>
</provider>
```

For more information about how to make the files managed by your app openable from other apps, read the Storage Access Framework (/guide/topics/providers/document-provider) guide.

# Local Actions

## Call a car

To call a taxi, use the `ACTION_RESERVE_TAXI_RESERVATION` (/reference/com/google/android/gms/actions/ReserveIntents#ACTION_RESERVE_TAXI_RESERVATION) action.

**Note:** Apps must ask for confirmation from the user before completing the action.

(https://developers.google.co
actions/system/#system_act

**Google Voice Actions**

- *"get me a taxi"*
- *"call me a car"*

(Wear OS only)

### Action

`ACTION_RESERVE_TAXI_RESERVATION` (/reference/com/google/android/gms/actions/ReserveIntents#ACTION_RESERVE_TAXI_RESERVATION)

### Data URI

None

### MIME Type

None

### Extras

None

### Example intent:

KOTLIN (#KOTLIN)JAVA

```
public void callCar() {
    Intent intent = new Intent(ReserveIntents.ACTION_RESERVE_TAXI_RESERVATION);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

### Example intent filter:

```
<activity ...>
    <intent-filter>
        <action android:name="com.google.android.gms.actions.RESERVE_TAXI_RESERVATION" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

# Maps

## Show a location on a map

To open a map, use the **ACTION_VIEW** (/reference/android/content/Intent#ACTION_VIEW) action and specify the location information in the intent data with one of the schemes defined below.

### Action

**ACTION_VIEW** (/reference/android/content/Intent#ACTION_VIEW)

### Data URI Scheme

`geo:`*`latitude`*`,`*`longitude`*

Show the map at the given longitude and latitude.
Example: "`geo:47.6,-122.3`"

`geo:`*`latitude`*`,`*`longitude`*`?z=`*`zoom`*

Show the map at the given longitude and latitude at a certain zoom level. A zoom level of 1 shows the whole Earth, centered at the given *lat*,*lng*. The highest (closest) zoom level is 23.

Example: "`geo:47.6,-122.3?z=11`"

`geo:0,0?q=lat,lng(label)`

Show the map at the given longitude and latitude with a string label.
Example: "`geo:0,0?q=34.99,-106.61(Treasure)`"

`geo:0,0?q=my+street+address`

Show the location for "my street address" (may be a specific address or location query).

Example: "**geo:0,0?q=1600+Amphitheatre+Parkway%2C+CA**"

★ **Note:** All strings passed in the **geo** URI must be encoded. For example, the string **1st & Pike, Seattle** should become **1st%20%26%20Pike%2C%20Seattle**. Spaces in the string can be encoded with **%20** or replaced with the plus sign (**+**).

### MIME Type

None

### Example intent:

KOTLIN (#KOTLIN)JAVA

```
public void showMap(Uri geoLocation) {
    Intent intent = new Intent(Intent.ACTION_VIEW);
    intent.setData(geoLocation);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

### Example intent filter:

```
<activity ...>
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <data android:scheme="geo" />
        <category android:name="android.intent.category.DEFAULT" />
```

```
    </intent-filter>
</activity>
```

# Music or Video

## Play a media file

To play a music file, use the **`ACTION_VIEW`** (/reference/android/content/Intent#ACTION_VIEW) action and specify the URI location of the file in the intent data.

### Action

**`ACTION_VIEW`** (/reference/android/content/Intent#ACTION_VIEW)

### Data URI Scheme

`file:`*`<URI>`*

`content:`*`<URI>`*

`http:`*`<URL>`*

### MIME Type

`"audio/*"`

`"application/ogg"`

`"application/x-ogg"`

`"application/itunes"`

**Or any other that your app may require.**

**Example intent:**

KOTLIN (#KOTLIN)JAVA

```
public void playMedia(Uri file) {
    Intent intent = new Intent(Intent.ACTION_VIEW);
    intent.setData(file);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

**Example intent filter:**

```
<activity ...>
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <data android:type="audio/*" />
        <data android:type="application/ogg" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

## Play music based on a search query

To play music based on a search query, use the `INTENT_ACTION_MEDIA_PLAY_FROM_SEARCH` (/reference/android/provider/MediaStore#INTENT_ACTION_MEDIA_PLAY_FROM_SEARCH) intent. An app may fire this intent in response to the user's voice command to play music. The receiving app for this intent performs a search within its inventory to match existing content to the given query and starts playing that content.

This intent should include the `EXTRA_MEDIA_FOCUS` (/reference/android/provider/MediaStore#EXTRA_MEDIA_FOCUS) string extra, which specifies the intended search mode. For example, the search mode can specify whether the search is for an artist name or song name.

(https://developers.google.co
actions/system/#system_act

**Google Voice Actions**

- *"play michael jackson billie jean"*

### Action

`INTENT_ACTION_MEDIA_PLAY_FROM_SEARCH` (/reference/android/provider/MediaStore#INTENT_ACTION_MEDIA_PLAY_FROM_SEARCH)

### Data URI Scheme

None

### MIME Type

None

### Extras

`MediaStore.EXTRA_MEDIA_FOCUS` (/reference/android/provider/MediaStore#EXTRA_MEDIA_FOCUS) **(required)**

Indicates the search mode (whether the user is looking for a particular artist, album, song, or playlist). Most search modes take additional extras. For example, if the user is interested in listening to a particular song, the intent might have three additional extras: the song title, the artist, and the album. This intent supports the following search modes for each value of `EXTRA_MEDIA_FOCUS` (/reference/android/provider/MediaStore#EXTRA_MEDIA_FOCUS):

***Any*** **-** `"vnd.android.cursor.item/*"`

Play any music. The receiving app should play some music based on a smart choice, such as the last playlist the user listened to.

Additional extras:

- `QUERY` (/reference/android/app/SearchManager#QUERY) (required) - An empty string. This extra is always provided for backward compatibility: existing apps that do not know about search modes can process this intent as an unstructured search.

***Unstructured*** **-** `"vnd.android.cursor.item/*"`

Play a particular song, album or genre from an unstructured search query. Apps may generate an intent with this search mode when they can't identify the type of content the user wants to listen to. Apps should use more specific search modes when possible.

Additional extras:

- `QUERY` (/reference/android/app/SearchManager#QUERY) (required) - A string that contains any combination of: the artist, the album, the song name, or the genre.

***Genre*** **-** `Audio.Genres.ENTRY_CONTENT_TYPE` (/reference/android/provider/MediaStore.Audio.Genres#ENTRY_CONTENT_TYPE)

Play music of a particular genre.

Additional extras:

- `"android.intent.extra.genre"` (required) - The genre.
- `QUERY` (/reference/android/app/SearchManager#QUERY) (required) - The genre. This extra is always provided for backward compatibility: existing apps that do not know about search modes can process this intent as an unstructured search.

***Artist*** **-** `Audio.Artists.ENTRY_CONTENT_TYPE` (/reference/android/provider/MediaStore.Audio.Artists#ENTRY_CONTENT_TYPE)

Play music from a particular artist.

Additional extras:

- `EXTRA_MEDIA_ARTIST` (/reference/android/provider/MediaStore#EXTRA_MEDIA_ARTIST) (required) - The artist.
- `"android.intent.extra.genre"` - The genre.
- `QUERY` (/reference/android/app/SearchManager#QUERY) (required) - A string that contains any combination of the artist or the genre. This extra is always provided for backward compatibility: existing apps that do not know about search modes can process this intent as an unstructured search.

***Album*** **-** `Audio.Albums.ENTRY_CONTENT_TYPE` (/reference/android/provider/MediaStore.Audio.Albums#ENTRY_CONTENT_TYPE)

Play music from a particular album.

Additional extras:

- `EXTRA_MEDIA_ALBUM` (/reference/android/provider/MediaStore#EXTRA_MEDIA_ALBUM) (required) - The album.
- `EXTRA_MEDIA_ARTIST` (/reference/android/provider/MediaStore#EXTRA_MEDIA_ARTIST) - The artist.
- `"android.intent.extra.genre"` - The genre.
- `QUERY` (/reference/android/app/SearchManager#QUERY) (required) - A string that contains any combination of the album or the artist. This extra is always provided for backward compatibility: existing apps that do not know about search modes can process this intent as an unstructured search.

***Song*** **-** `"vnd.android.cursor.item/audio"`

Play a particular song.

Additional extras:

- `EXTRA_MEDIA_ALBUM` (/reference/android/provider/MediaStore#EXTRA_MEDIA_ALBUM) - The album.

- `EXTRA_MEDIA_ARTIST` (/reference/android/provider/MediaStore#EXTRA_MEDIA_ARTIST) - The artist.
- `"android.intent.extra.genre"` - The genre.
- `EXTRA_MEDIA_TITLE` (/reference/android/provider/MediaStore#EXTRA_MEDIA_TITLE) (required) - The song name.
- `QUERY` (/reference/android/app/SearchManager#QUERY) (required) - A string that contains any combination of: the album, the artist, the genre, or the title. This extra is always provided for backward compatibility: existing apps that do not know about search modes can process this intent as an unstructured search.

***Playlist* - `Audio.Playlists.ENTRY_CONTENT_TYPE`**
(/reference/android/provider/MediaStore.Audio.Playlists#ENTRY_CONTENT_TYPE)

Play a particular playlist or a playlist that matches some criteria specified by additional extras.

Additional extras:

- `EXTRA_MEDIA_ALBUM` (/reference/android/provider/MediaStore#EXTRA_MEDIA_ALBUM) - The album.
- `EXTRA_MEDIA_ARTIST` (/reference/android/provider/MediaStore#EXTRA_MEDIA_ARTIST) - The artist.
- `"android.intent.extra.genre"` - The genre.
- `"android.intent.extra.playlist"` - The playlist.
- `EXTRA_MEDIA_TITLE` (/reference/android/provider/MediaStore#EXTRA_MEDIA_TITLE) - The song name that the playlist is based on.
- `QUERY` (/reference/android/app/SearchManager#QUERY) (required) - A string that contains any combination of: the album, the artist, the genre, the playlist, or the title. This extra is always provided for backward compatibility: existing apps that do not know about search modes can process this intent as an unstructured search.

**Example intent:**

If the user wants to listen to music from a particular artist, a search app may generate the following intent:

KOTLIN (#KOTLIN)JAVA

```
public void playSearchArtist(String artist) {
    Intent intent = new Intent(MediaStore.INTENT_ACTION_MEDIA_PLAY_FROM_SEARCH);
    intent.putExtra(MediaStore.EXTRA_MEDIA_FOCUS,
                    MediaStore.Audio.Artists.ENTRY_CONTENT_TYPE);
    intent.putExtra(MediaStore.EXTRA_MEDIA_ARTIST, artist);
    intent.putExtra(SearchManager.QUERY, artist);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

**Example intent filter:**

```
<activity ...>
    <intent-filter>
        <action android:name="android.media.action.MEDIA_PLAY_FROM_SEARCH" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

When handling this intent, your activity should check the value of the `EXTRA_MEDIA_FOCUS` (/reference/android/provider/MediaStore#EXTRA_MEDIA_FOCUS) extra in the incoming `Intent` (/reference/android/content/Intent) to determine the search mode. Once your activity has identified the search mode, it should read the values of the additional extras for that particular search mode. With this information your app can then perform the search within its inventory to play the content that matches the search query. For example:

```
protected void onCreate(Bundle savedInstanceState) {
    //...
    Intent intent = this.getIntent();
    if (intent.getAction().compareTo(MediaStore.INTENT_ACTION_MEDIA_PLAY_FROM_SEARCH) == 0) {

        String mediaFocus = intent.getStringExtra(MediaStore.EXTRA_MEDIA_FOCUS);
        String query = intent.getStringExtra(SearchManager.QUERY);

        // Some of these extras may not be available depending on the search mode
        String album = intent.getStringExtra(MediaStore.EXTRA_MEDIA_ALBUM);
        String artist = intent.getStringExtra(MediaStore.EXTRA_MEDIA_ARTIST);
        String genre = intent.getStringExtra("android.intent.extra.genre");
        String playlist = intent.getStringExtra("android.intent.extra.playlist");
        String title = intent.getStringExtra(MediaStore.EXTRA_MEDIA_TITLE);

        // Determine the search mode and use the corresponding extras
        if (mediaFocus == null) {
            // 'Unstructured' search mode (backward compatible)
            playUnstructuredSearch(query);

        } else if (mediaFocus.compareTo("vnd.android.cursor.item/*") == 0) {
            if (query.isEmpty()) {
                // 'Any' search mode
                playResumeLastPlaylist();
            } else {
                // 'Unstructured' search mode
                playUnstructuredSearch(query);
            }

        } else if (mediaFocus.compareTo(MediaStore.Audio.Genres.ENTRY_CONTENT_TYPE) == 0) {
            // 'Genre' search mode
            playGenre(genre);
```

```
        } else if (mediaFocus.compareTo(MediaStore.Audio.Artists.ENTRY_CONTENT_TYPE) == 0) {
            // 'Artist' search mode
            playArtist(artist, genre);

        } else if (mediaFocus.compareTo(MediaStore.Audio.Albums.ENTRY_CONTENT_TYPE) == 0) {
            // 'Album' search mode
            playAlbum(album, artist);

        } else if (mediaFocus.compareTo("vnd.android.cursor.item/audio") == 0) {
            // 'Song' search mode
            playSong(album, artist, genre, title);

        } else if (mediaFocus.compareTo(MediaStore.Audio.Playlists.ENTRY_CONTENT_TYPE) == 0) {
            // 'Playlist' search mode
            playPlaylist(album, artist, genre, playlist, title);
        }
    }
}
```

# New Note

## Create a note

To create a new note, use the **`ACTION_CREATE_NOTE`** (https://developers.google.com/android/reference/com/google/android/gms/actions/NoteIntents#ACTION_CREATE_NOTE) action and specify note details such as the subject and text using extras defined below.

**Note:** Apps must ask for confirmation from the user before completing the action.

### Action

ACTION_CREATE_NOTE (https://developers.google.com/android/reference/com/google/android/gms/actions/NoteIntents#ACTION_CREATE_NOTE)

### Data URI Scheme

None

### MIME Type

PLAIN_TEXT_TYPE (https://developer.android.com/reference/org/apache/http/protocol/HTTP.html#PLAIN_TEXT_TYPE)

"*/*"

### Extras

EXTRA_NAME (https://developers.google.com/android/reference/com/google/android/gms/actions/NoteIntents#EXTRA_NAME)

A string indicating the title or subject of the note.

EXTRA_TEXT (https://developers.google.com/android/reference/com/google/android/gms/actions/NoteIntents#EXTRA_TEXT)

A string indicating the text of the note.

### Example intent:

KOTLIN (#KOTLIN)JAVA

```
public void createNote(String subject, String text) {
    Intent intent = new Intent(NoteIntents.ACTION_CREATE_NOTE)
            .putExtra(NoteIntents.EXTRA_NAME, subject)
            .putExtra(NoteIntents.EXTRA_TEXT, text);
```

ARENDI_G329318

```
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

**Example intent filter:**

```
<activity ...>
    <intent-filter>
        <action android:name="com.google.android.gms.actions.CREATE_NOTE" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:mimeType="*/*" />
    </intent-filter>
</activity>
```

# Phone

## Initiate a phone call

To open the phone app and dial a phone number, use the **`ACTION_DIAL`** (/reference/android/content/Intent#ACTION_DIAL) action and specify a phone number using the URI scheme defined below. When the phone app opens, it displays the phone number but the user must press the *Call* button to begin the phone call.

To place a phone call directly, use the **`ACTION_CALL`** (/reference/android/content/Intent#ACTION_CALL) action and specify a phone number using the URI scheme defined below. When the phone app opens, it begins the phone call; the user does not need to press the *Call* button.

(https://developers.google.cc
actions/system/#system_act

**Google Voice Actions**

- *"call 555-5555"*
- *"call bob"*
- *"call voicemail"*

The `ACTION_CALL` (/reference/android/content/Intent#ACTION_CALL) action requires that you add the `CALL_PHONE` permission to your manifest file:

```
<uses-permission android:name="android.permission.CALL_PHONE" />
```

**Action**

- `ACTION_DIAL` (/reference/android/content/Intent#ACTION_DIAL) - Opens the dialer or phone app.
- `ACTION_CALL` (/reference/android/content/Intent#ACTION_CALL) - Places a phone call (requires the `CALL_PHONE` permission)

**Data URI Scheme**

- `tel:<phone-number>`
- `voicemail:<phone-number>`

**MIME Type**

None

Valid telephone numbers are those defined in the IETF RFC 3966 (http://tools.ietf.org/html/rfc3966). Valid examples include the following:

- `tel:2125551212`
- `tel:(212) 555 1212`

The Phone's dialer is good at normalizing schemes, such as telephone numbers. So the scheme described isn't strictly required in the `Uri.parse()` (/reference/android/net/Uri#parse(java.lang.String)) method. However, if you have not tried a scheme or are unsure whether it

can be handled, use the `Uri.fromParts()` (/reference/android/net/Uri#fromParts(java.lang.String, java.lang.String, java.lang.String)) method instead.

**Example intent:**

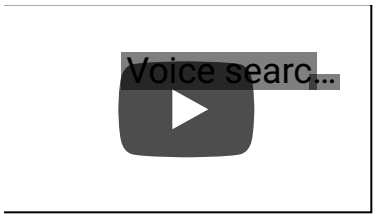
KOTLIN (#KOTLIN)JAVA

```
public void dialPhoneNumber(String phoneNumber) {
    Intent intent = new Intent(Intent.ACTION_DIAL);
    intent.setData(Uri.parse("tel:" + phoneNumber));
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

# Search

## Search using a specific app

To support search within the context of your app, declare an intent filter in your app with the `SEARCH_ACTION` action, as shown in the example intent filter below.




**Action**

`"com.google.android.gms.actions.SEARCH_ACTION"`

Support search queries from Google Voice Actions.

(https://developers.google.co
actions/system/#system_act

**Google Voice Actions**

- *"search for cat videos on myvideoapp"*

**Extras**

`QUERY` (/reference/android/app/SearchManager#QUERY)

A string that contains the search query.

**Example intent filter:**

```
<activity android:name=".SearchActivity">
    <intent-filter>
        <action android:name="com.google.android.gms.actions.SEARCH_ACTION"/>
        <category android:name="android.intent.category.DEFAULT"/>
    </intent-filter>
</activity>
```

## Perform a web search

To initiate a web search, use the `ACTION_WEB_SEARCH` (/reference/android/content/Intent#ACTION_WEB_SEARCH) action and specify the search string in the `SearchManager.QUERY` (/reference/android/app/SearchManager#QUERY) extra.

**Action**

`ACTION_WEB_SEARCH` (/reference/android/content/Intent#ACTION_WEB_SEARCH)

**Data URI Scheme**

None

**MIME Type**

None

**Extras**

`SearchManager.QUERY` (/reference/android/app/SearchManager#QUERY)

The search string.

**Example intent:**

KOTLIN (#KOTLIN)JAVA

```
public void searchWeb(String query) {
    Intent intent = new Intent(Intent.ACTION_WEB_SEARCH);
    intent.putExtra(SearchManager.QUERY, query);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

# Settings

## Open a specific section of Settings

To open a screen in the system settings when your app requires the user to change something, use one of the following intent actions to open the settings screen respective to the action name.

**Action**

`ACTION_SETTINGS` (/reference/android/provider/Settings#ACTION_SETTINGS)

`ACTION_WIRELESS_SETTINGS` (/reference/android/provider/Settings#ACTION_WIRELESS_SETTINGS)

ACTION_AIRPLANE_MODE_SETTINGS (/reference/android/provider/Settings#ACTION_AIRPLANE_MODE_SETTINGS)
ACTION_WIFI_SETTINGS (/reference/android/provider/Settings#ACTION_WIFI_SETTINGS)
ACTION_APN_SETTINGS (/reference/android/provider/Settings#ACTION_APN_SETTINGS)
ACTION_BLUETOOTH_SETTINGS (/reference/android/provider/Settings#ACTION_BLUETOOTH_SETTINGS)
ACTION_DATE_SETTINGS (/reference/android/provider/Settings#ACTION_DATE_SETTINGS)
ACTION_LOCALE_SETTINGS (/reference/android/provider/Settings#ACTION_LOCALE_SETTINGS)
ACTION_INPUT_METHOD_SETTINGS (/reference/android/provider/Settings#ACTION_INPUT_METHOD_SETTINGS)
ACTION_DISPLAY_SETTINGS (/reference/android/provider/Settings#ACTION_DISPLAY_SETTINGS)
ACTION_SECURITY_SETTINGS (/reference/android/provider/Settings#ACTION_SECURITY_SETTINGS)
ACTION_LOCATION_SOURCE_SETTINGS (/reference/android/provider/Settings#ACTION_LOCATION_SOURCE_SETTINGS)
ACTION_INTERNAL_STORAGE_SETTINGS (/reference/android/provider/Settings#ACTION_INTERNAL_STORAGE_SETTINGS)
ACTION_MEMORY_CARD_SETTINGS (/reference/android/provider/Settings#ACTION_MEMORY_CARD_SETTINGS)

See the Settings (/reference/android/provider/Settings) documentation for additional settings screens that are available.

### Data URI Scheme

None

### MIME Type

None

### Example intent:

KOTLIN (#KOTLIN)JAVA

```
public void openWifiSettings() {
    Intent intent = new Intent(Settings.ACTION_WIFI_SETTINGS);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
```

```
    }
}
```

# Text Messaging

## Compose an SMS/MMS message with attachment

To initiate an SMS or MMS text message, use one of the intent actions below and specify message details such as the phone number, subject, and message body using the extra keys listed below.

### Action

**ACTION_SENDTO** (/reference/android/content/Intent#ACTION_SENDTO) or
**ACTION_SEND** (/reference/android/content/Intent#ACTION_SEND) or
**ACTION_SEND_MULTIPLE** (/reference/android/content/Intent#ACTION_SEND_MULTIPLE)

### Data URI Scheme

`sms:<phone_number>`

`smsto:<phone_number>`

`mms:<phone_number>`

`mmsto:<phone_number>`

Each of these schemes are handled the same.

### MIME Type

`"text/plain"`

`"image/*"`

`"video/*"`

**Extras**

`"subject"`

A string for the message subject (usually for MMS only).

`"sms_body"`

A string for the text message.

`EXTRA_STREAM` (/reference/android/content/Intent#EXTRA_STREAM)

A `Uri` (/reference/android/net/Uri) pointing to the image or video to attach. If using the `ACTION_SEND_MULTIPLE` (/reference/android/content/Intent#ACTION_SEND_MULTIPLE) action, this extra should be an `ArrayList` (/reference/java/util/ArrayList) of `Uri` (/reference/android/net/Uri)s pointing to the images/videos to attach.

**Example intent:**

KOTLIN (#KOTLIN)JAVA

```
public void composeMmsMessage(String message, Uri attachment) {
    Intent intent = new Intent(Intent.ACTION_SENDTO);
    intent.setType(HTTP.PLAIN_TEXT_TYPE);
    intent.putExtra("sms_body", message);
    intent.putExtra(Intent.EXTRA_STREAM, attachment);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

ARENDI_G329326

If you want to ensure that your intent is handled only by a text messaging app (and not other email or social apps), then use the **ACTION_SENDTO** (/reference/android/content/Intent#ACTION_SENDTO) action and include the "**smsto:**" data scheme. For example:

**KOTLIN** (#KOTLIN)**JAVA**

```
public void composeMmsMessage(String message, Uri attachment) {
    Intent intent = new Intent(Intent.ACTION_SEND);
    intent.setData(Uri.parse("smsto:"));  // This ensures only SMS apps respond
    intent.putExtra("sms_body", message);
    intent.putExtra(Intent.EXTRA_STREAM, attachment);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

**Example intent filter:**

```
<activity ...>
    <intent-filter>
        <action android:name="android.intent.action.SEND" />
        <data android:type="text/plain" />
        <data android:type="image/*" />
        <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</activity>
```

**Note:** If you're developing an SMS/MMS messaging app, you must implement intent filters for several additional actions in order to be available as the *default SMS app* on Android 4.4 and higher. For more information, see the documentation at **`Telephony`** (/reference/android/provider/Telephony).

# Web Browser

## Load a web URL

To open a web page, use the **`ACTION_VIEW`** (/reference/android/content/Intent#ACTION_VIEW) action and specify the web URL in the intent data.

(https://developers.google.cc
actions/system/#system_act

**Google Voice Actions**

- *"open example.com"*

### Action

**`ACTION_VIEW`** (/reference/android/content/Intent#ACTION_VIEW)

### Data URI Scheme

`http:<URL>`
`https:<URL>`

### MIME Type

`"text/plain"`

`"text/html"`

`"application/xhtml+xml"`

`"application/vnd.wap.xhtml+xml"`

### Example intent:

KOTLIN (#KOTLIN)JAVA

```
public void openWebPage(String url) {
    Uri webpage = Uri.parse(url);
    Intent intent = new Intent(Intent.ACTION_VIEW, webpage);
    if (intent.resolveActivity(getPackageManager()) != null) {
        startActivity(intent);
    }
}
```

**Example intent filter:**

```
<activity ...>
    <intent-filter>
        <action android:name="android.intent.action.VIEW" />
        <!-- Include the host attribute if you want your app to respond
             only to URLs with your app's domain. -->
        <data android:scheme="http" android:host="www.example.com" />
        <category android:name="android.intent.category.DEFAULT" />
        <!-- The BROWSABLE category is required to get links from web pages. -->
        <category android:name="android.intent.category.BROWSABLE" />
    </intent-filter>
</activity>
```

**Tip:** If your Android app provides functionality similar to your web site, include an intent filter for URLs that point to your web site. Then, if users have your app installed, links from emails or other web pages pointing to your web site open your Android app instead of your web page.

# Verify Intents with the Android Debug Bridge

To verify that your app responds to the intents that you want to support, you can use the **adb** (/tools/help/adb) tool to fire specific intents:

1. Set up an Android device for development (/tools/device#setting-up), or use a virtual device (/tools/devices/emulator#avds).
2. Install a version of your app that handles the intents you want to support.
3. Fire an intent using **adb**:

   ```
   adb shell am start -a <ACTION> -t <MIME_TYPE> -d <DATA> \
     -e <EXTRA_NAME> <EXTRA_VALUE> -n <ACTIVITY>
   ```

   For example:

   ```
   adb shell am start -a android.intent.action.DIAL \
     -d tel:555-5555 -n org.example.MyApp/.MyActivity
   ```

4. If you defined the required intent filters, your app should handle the intent.

For more information, see ADB Shell Commands (/tools/help/shell#am).



Last updated 2019-12-27.

# Exhibit 21


# Newton Programmer's Guide

**For Newton 2.0**

**Addison-Wesley Publishing Company**

Reading, Massachusetts Menlo Park, California New York
Don Mills, Ontario Harlow, England Amsterdam Bonn
Sydney Singapore Tokyo Madrid San Juan
Paris Seoul Milan Mexico City Taipei

## Stationery

Stationery is a capability of the system that allows applications to be extended by other developers. The word "stationery" refers to the capability of having different kinds of data within a single application (such as plain notes and outlines in the Notepad) and/or to the capability of having different ways of viewing the same data (such as the Card and All Info views in the Names file). An application that supports stationery can be extended either by adding a new type of data to it (for example, adding recipe cards to the Notepad), or by adding a new type of viewer for existing data (a new way of viewing Names file entries or a new print format, for example).

To support stationery, an application must register with the system a frame, called a data definition, that describes the data with which it works. The different data definitions available to an application are listed on the pop-up menu attached to the New button. In addition, an application must register one or more view definitions, which describe how the data is to be viewed or printed. View definitions can include simple read-only views, editor-type views, or print formats. The different view definitions available in an application (not including print formats) are listed on the pop-up menu attached to the Show button.

Stationery is well integrated into the NewtApp framework, so if you use that framework for your application, using stationery is easy. The printing architecture also uses stationery, so all application print formats are registered as a kind of stationery.

For more information about using stationery, see Chapter 5, "Stationery."

## Intelligent Assistant

A key part of the Newton information architecture is the Intelligent Assistant. The Intelligent Assistant is a system service that attempts to complete actions for the user according to deductions it makes about the task that the user is currently performing. The Assistant is always instantly available to the user through the Assist button, yet remains nonintrusive.

The Assistant knows how to complete several built-in tasks; they are Scheduling (adding meetings), Finding, Reminding (adding To Do items), Mailing, Faxing, Printing, Calling, and getting time information from the Time Zones map. Each of these tasks has several synonyms; for example, the user can write "call," "phone," "ring," or "dial" to make a phone call.

Applications can add new tasks so that the Assistant supports their special capabilities and services. The Newton unified data model makes it possible for the Assistant to access data stored by any application, thus allowing the Assistant to be well integrated in the system.

For details on using the Intelligent Assistant and integrating support for it into your application, see Chapter 18, "Intelligent Assistant."

## Imaging and Printing

At the operating system level, the Newton imaging and printing software is based on an object-oriented, device-independent imaging model. The imaging model is monochrome since the current Newton screen is a black-and-white screen.

NewtonScript application programs don't call low-level imaging routines directly to do drawing or image manipulation. In fact, most drawing is handled for applications by the user interface components they incorporate, or when they call other routines that display information. However, there is a versatile set of high-level drawing routines that you can call directly to create and draw shapes, pictures, bitmaps, and text. When drawing, you can vary the pen thickness, pen pattern, fill pattern, and other attributes. For details on drawing, refer to Chapter 13, "Drawing and Graphics."

The Newton text imaging facility supports Unicode directly, so the system can be easily localized to display languages using different script systems. The system is extensible, so it's possible to add additional fonts, font engines, and printer drivers.

The high-level interface to printing on the Newton uses a model identical to that used for views. Essentially, you design a special kind of view called a print format to specify how printed information is to be laid out on the page. Print formats use a unique view template that automatically adjusts its size to the page size of the printer chosen by the user. When the user prints, the system handles all the details of rendering the views on the printer according to the layout you specified.

The Newton offers the feature of deferred printing. The user can print even though he or she is not connected to a printer at the moment. An object describing the print job is stored in the Newton Out Box application, and when a printer is connected later, the user can then select that print job for printing. Again, this feature is handled automatically by the system and requires no additional application programming work.

For information on how to add printing capabilities to an application, refer to Chapter 21, "Routing Interface."

## Sound

The Newton includes a monophonic speaker and can play sounds sampled at rates up to 22 kHz. You can attach sounds to particular events associated with a view, such as showing it, hiding it, and scrolling it. You can also use sound routines to play sounds synchronously or asynchronously at any other time.

Newton can serve as a phone dialer by dialing phone numbers through the speaker. The dialing tones are built into the system ROM, along with several other sounds that can be used in applications.

CHAPTER 16

# Find

This chapter describes how your application can support finding text, dates, or your own data types in its data. If you want users to be able to use the system's Find service to locate data in your application, you should be familiar with the material discussed in this chapter.

Before reading this chapter, you should understand the concept of the target of an action, explained in Chapter 15, "Filing." Familiarity with using views to image data, covered in Chapter 3, "Views," is also helpful. If your application stores data as soup entries, you should understand the contents of Chapter 11, "Data Storage and Retrieval."

This chapter is divided into two main parts:

- "About the Find Service" describes the core user interface to the Find service, along with variations and optional features. A compatibility section covers differences between the current version of the Find service and previous ones.
- "Using the Find Service" provides a technical overview of Find operations, with code examples to show how to implement support for this service in your application.

In addition, the "Find Reference" (page 13-1) in *Newton Programmer's Reference* provides complete descriptions of all Find service data structures, functions, and methods.

## About the Find Service

The Find service searches for occurrences of data items the user specifies on a Find slip. The Find slip may be supplied by the system or by the developer. Figure 16-1 illustrates the system-supplied Find slip.

Find

Figure 15-1 The system-supplied Find slip


The system-supplied Find slip contains an input line that specifies a search string, several buttons indicate the scope of the search, and a Look For picker (pop-up menu) that specifies the kind of search to perform. By choosing from the Look For picker (pop-up menu) you may specify whether the search string is a text item or a date, as shown in Figure 15-2.

Figure 15-2 Specifying text or date searches in the Find slip


Text searches are case insensitive and find only string beginnings. That is, a search for the string "smith" may return the items "Smith" and "smithers," but not "blacksmith." Date searches find items dated before, after, or on the date specified by the search string.

From the application developer's perspective, text finds and date finds are nearly identical. The only significant difference between them is the test an item must pass to be included in the result of the search.

The system-supplied Find slip always contains an Everywhere button and Selected button. If the current application supports the Find service, a button with the application's name appears in this slip as well.

CHAPTER 16

Find

Searching for data in the current application only is called a **Local find** operation. Figure 16-3 depicts a Local find in the Notepad application.

**Figure 16-3** A local Find operation


The Everywhere and Selected buttons specify that the system perform searches in applications other than the currently active one. Applications must register with the Find service to participate in such searches.

Tapping the Everywhere button tells the system to conduct searches in all currently available applications registered with the Find service. This kind of search is called a **Global find**. Applications need not be open to participate in a Global find.

A Global find is similar to a series of Local find operations initiated by the system. When the user requests a Global find, the system executes a Local find in each application registered with the Find service.

Tapping the Selected button causes a checklist to appear at the top of the Find slip. The list includes all currently available applications registered with the Find service. Tapping items in the list places a check mark next to applications in which the system should conduct a Local find. This kind of search is called a **Selected find**. The slip in Figure 16-4 depicts a search for the string "Daphne" in the Notes and Dates applications.

**Figure 16-4** Searching selected applications

Find

In addition, an application can support searches of multiple data sets. For example, a personal finance program might allow you to search the check ledger, the account list, and the credit charges list as separate searches, even though all the data resides in a single application. For more information on how to implement this in your application see "Adding Application Data Sets to Selected Finds" beginning on page 16-19.

In addition, you can replace the Find slip in the currently active application. Typically, you would do this to provide a customized user interface for specialized searches. For more information, see "Replacing the Built-in Find Slip" beginning on page 16-24.

After setting the parameters of the search with the Find slip, the user initiates the search by tapping the Find button. Alternatively, the user can cancel the search by tapping the close box to dismiss the Find slip.

While the search executes, the system provides user feedback through a Progress slip. This slip provides a Stop button that allows the user to cancel a search in progress. Figure 16-5 shows a typical Progress slip.

Figure 16-5 Progress slip


When the search is completed, the Find service displays an overview list of items that match the search criteria. Figure 16-6 shows the Find overview as it might appear after searching all applications for the string "Dagmar".

**Figure 16-6** The Find overview


The user can tap items in the Find overview to display them. As items are displayed, a status message at the top of the Find slip indicates which item is displayed and whether there are more results to display. Figure 16-7 depicts this status message.

**Figure 16-7** Find status message


When more than one item is found, the status message indicates that there are more items to display.

Between uses, the Find service stores the setting of the Look For picker. The next time this service is used, it reopens in the most recently set find mode. Note that in order to conserve memory, the list of found items is not saved between finds.

CHAPTER 18

# Intelligent Assistant

The Intelligent Assistant is a system service that attempts to complete actions specified by the user's written input. You can think of the Assistant as an alternate interface to Newton applications and services.

The Assistant can use the built-in applications to complete predefined tasks such as calling, faxing, printing, scheduling, and so on. You can also program the Assistant to execute any task that your application performs. In addition, you can display your application's online help from the Assistant.

This chapter describes how to make application behaviors and online help available from the Assistant. If you want to provide a textual interface to your application or its online help, you should become familiar with the Assistant and the concepts discussed in this chapter.

Before reading this chapter, you should be familiar with the concept of the target of an action, as explained in Chapter 15, "Filing," and you should understand the behavior or service that your application provides through the Assistant. Although it is not essential, it is helpful if you are familiar with lexical dictionaries. Lexical dictionaries are described in Chapter 9, "Recognition."

## About the Assistant

This section describes the Assistant's behavior in a variety of user scenarios, and then provides an overview of the templates, frames, and methods that provide this behavior.

### Introduction

When the user invokes the Assistant, the system passes the current text selection to it. If no text is selected, the system passes to the Assistant the contents of a buffer that holds the most recent text input.

The Assistant then attempts to match words in the input string with templates and dictionaries that classify the words as actions, targets, or unknown entities. The

templates may be supplied by the system or by your application. Some system-supplied templates make use of lexical dictionaries, which are also supplied by the system. For more information about lexical dictionaries, see Chapter 9, "Recognition."

Depending on the amount of information that parsing the input string provides, the Assistant either attempts to complete a task or prompts the user to supply additional information.

## Input Strings

The Assistant is preprogrammed with a list of words representing tasks that the built-in applications can perform. In addition to using these words, you can program the Assistant to associate your own words with tasks your application performs. The user cannot add words to the Assistant's vocabulary.

You can associate multiple verbs with a single action; for example, the system supplies a template that maps the words `"call"`, `"dial"`, `"phone"`, and `"ring"` to the same task.

The Assistant uses some of the same dictionaries as the recognition system when attempting to classify items in the input string. For example, it uses the system's built-in lexical dictionaries to classify strings as phone numbers.

The word order in the input phrase is not significant— for example, the input phrase `"Bob fax"` produces the same result as the phrase `"Fax Bob"`. This syntax-independent architecture allows easier localization of applications for international audiences.

**Note**
The input string passed to the Assistant must not contain more than 15 words. The Assistant does not attempt to interpret strings containing more than 15 words. ◆

## No Verb in Input String

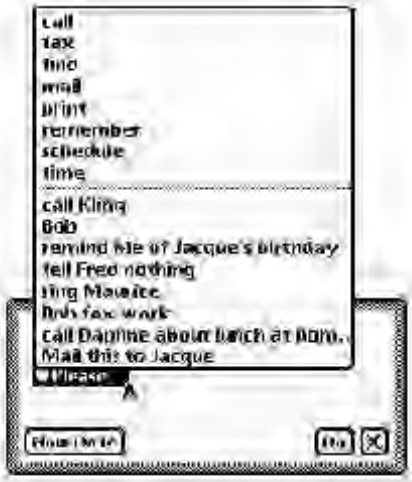
If the Assistant cannot determine the user's intended action, it displays an Assist slip that prompts the user to tap the Please picker for more options. The Please picker allows the user to specify an action when the Assistant cannot determine one by parsing the input string.

For example, using the string `"Bob"` as partial input, the Assistant can perform a number of actions: it can find Bob, fax Bob, call Bob, schedule a meeting with Bob, and so on. However, this input string does not indicate which of these actions is the user's actual intent. Figure 18-1 shows the Assist slip as it would appear if the string `"Bob"` were the only input provided to the Assistant.

Figure 18-1 Assist slip


When prompted by the Assist slip, the user must provide additional information on its input line or choose an action from the Please picker. The top portion of the Please picker displays all of the actions currently registered with the Assistant. The Please picker is shown in Figure 18-2.

Figure 18-2 The Please picker




The built-in tasks that the Assistant can complete include calling, faxing, finding, mailing, printing, remembering (adding To Do items), scheduling (adding meetings), getting time information from the built-in Time Zones application and displaying online help. Note that the top portion of this menu displays only one word for each action the Assistant can perform. For example, the word "call" appears here but the synonyms "ring" and "phone" do not. Recently used synonyms may appear in the bottom half of the slip, however.

To allow the user to repeat recent actions easily, the bottom portion of the Please picker displays the eight strings most recently passed to the Assistant. Thus, the string "Ring Katrina" appears here, even though the action of placing a phone call is represented only by the verb "call" in the top portion of the picker.

After making corrections to the input string, the user can tap the Do button in the Assist slip to pass the corrected string to the Assistant once again.

## Ambiguous or Missing Information

If the input string specifies an action, the Assistant does not display the Assist slip containing the Please picker, but may still need to obtain additional information in order to complete the task.

When an action is specified but required information is still missing, the Assistant tries to supply as much of the required information as possible. For example, if the input string is "Fax Bob", the Assistant can query the Names soup for information such as Bob's name and fax number. However, the user may still need to correct the input if the Assistant chooses the wrong Bob from the Names soup, cannot find Bob in the Names soup, or cannot find Bob's fax number in this soup.

The user can resolve ambiguities or provide additional information from within a task slip that the Assistant displays for this purpose.

## The Task Slip

The task slip provides the user with a final opportunity to correct input to the Assistant and confirm or dismiss execution of the task before the Assistant actually takes action. Although it's recommended that you always provide this opportunity to confirm, modify, or cancel the task before taking action, it's especially important to do so when execution of the task will open other applications or otherwise inconvenience the user.

Figure 18-3 Calling task slip

## Programmer's Overview

This section describes how the templates, frames and methods used by the Assistant interact to provide services to the user.

You can think of any user operation as a set of one or more actions to complete. A single action is represented to the Assistant by an **action template.** You can use action templates supplied by the system and you can also define your own action templates.

Some actions require data or objects on which to operate. For example, a phone-dialing action requires a phone number to dial. The data or object on which an action operates is represented to the Assistant by a **target template.** The system provides a variety of target templates, and you can define your own as well.

Each action template or target template defines a set of one or more strings known as its **lexicon**. In general, an action template's `lexicon` slot holds one or more verbs, while the `lexicon` slot of a target template holds one or more nouns.

When any of the words in a template's lexicon appear in the input string passed to the Assistant, the Assistant builds a frame based on that template. For example, the lexicon for the built-in `call_act` template includes the strings `"call"`, `"ring"`, and `"dial"`. When any of these strings appear in the Assistant's input string, the Assistant builds a frame based on the `call_act` template. Frames created from action templates are called **action frames;** frames created from target templates are called **target frames.**

A **task** is a behavior (such as calling, printing, faxing, and so on) that is made available to the user by the Assistant. Usually a task consists of multiple actions and targets, although a task can be as simple as a single action having no target.

You define a task to the Assistant by using the `RegTaskTemplate` function to register a **task template** with the system. A task template is like a recipe: it specifies everything required to perform a particular task, including

- the actions and targets required to provide a specified behavior
- the behavior to provide
- supporting methods and data structures

The task template must specify all of the actions and targets required to perform the task it defines. You provide this information as an array of action templates and target templates residing in your task template's `signature` slot. When the task template is registered with the Assistant, all of the templates in the `signature` slot are registered as well. Once your task template is registered, the Assistant can match user input with words in any of the lexicons of templates specified in the task template's `signature` slot.

Your task template designates one action template as its **primary action.** The primary action is the one that the Assistant associates with a specified verb in its input string. The primary action is represented by an action template that resides in the task template's `primary_act` slot. Because task templates do not provide lexicons, the designation of a primary action provides the Assistant with a way to associate a verb in the input string with a task template.

In addition to items required by the Assistant, your task template can hold any additional methods or data structures you require to perform the task.

Recall that the Assistant creates an action frame or target frame for each word in the input string that matches a lexicon item in a registered template. If the Assistant can match an action frame with the `primary_act` slot in a task template, it uses that template to create a **task frame.** The task frame holds all the information defined in the task template, as well as some additional slots that the Assistant provides.

The Assistant creates slots in the task frame to hold action frames and task frames that are created as templates in the `signature` slot are matched. These slots are named using symbols that reside in a `preConditions` slot that your task template provides.

Each element of the `preConditions` array specifies the name of the slot that is to hold the frame created from the template in the corresponding element of the `signature` array. For example, presume that the fourth element of the `signature` array holds the `call_act` template and the fourth element of the `preConditions` array holds the `'myCallAction` symbol. When the `call_act` template is matched, the Assistant creates in a slot named `'myCallAction` in the task frame and places in that slot the action frame created from the `call_act` template. Later in this chapter, the section "The Signature and PreConditions Slots" (page 18-10) discusses this process in more detail.

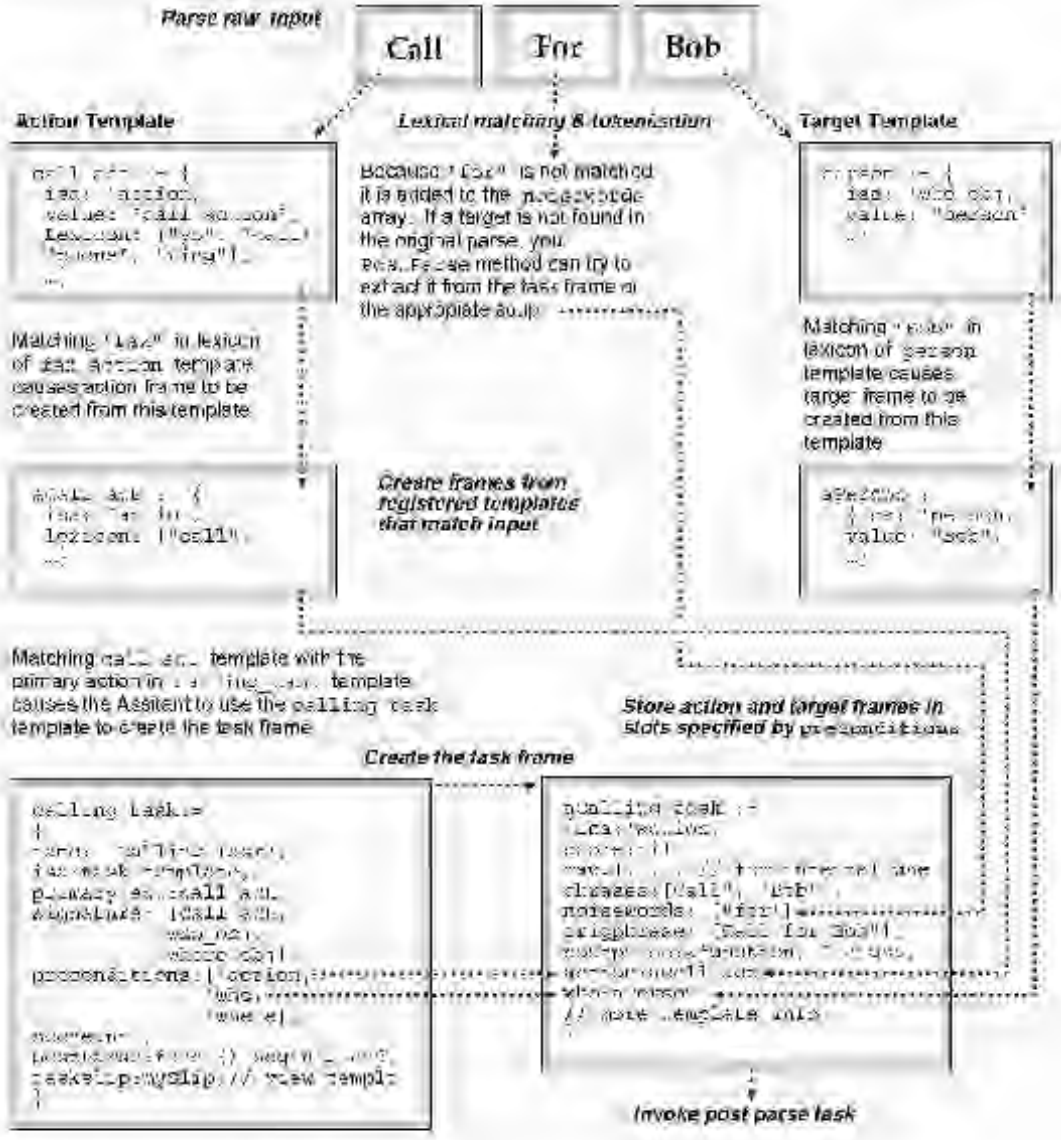
If the Assistant cannot match an action frame with any of the task templates currently registered, it displays the Assist slip, which prompts the user to specify an action.

If the Assistant matches more than one action frame with a currently registered task template's `primary_act` slot, it attempts to resolve the conflict by determining whether additional frames required by the task template are present. The conflict resolution process is described in "Resolving Template-Matching Conflicts" (page 18-13).

Once the Assistant has matched as many templates as possible and stored the resulting frames in the task frame, it sends the `PostParse` message to the task frame.

Figure 18-4 illustrates a simple example of how the Assistant builds a task frame by matching user input strings with registered templates.

Figure 18-4 Simplified overview of the Assistant's matching process



You define the `PostParse` method in your task template. Your `PostParse` method must perform any actions required to complete the primary task. The `PostParse` method usually acts as a dispatching method that executes the subtasks comprising the primary action. In doing so, the `PostParse` method may validate data and retrieve information that the Assistant was unable to find on its own.

**IMPORTANT**
Once your `PostParse` method is invoked, it is in complete control of the error-handling and user interface on the Newton. Your `PostParse` method must include any code necessary to complete the primary action, such as code required to display a task slip, validate input, and handle errors. ▲

## Matching Words With Templates

This section discusses the process of extracting words or phrases from user input and matching them to templates registered with the Assistant. In particular, this section provides more detail regarding unmatched words, partially matched phrases, and words that match multiple templates.

When the user taps the Assist button, the system passes the current input string to the `ParseUtter` function, which matches words in the input string with elements in the lexicons of templates currently registered with the Assistant. Normally, you do not need to call the `ParseUtter` function yourself; however, you can experiment with the Assistant by passing strings to this function in the NTK Inspector window.

When parsing the input string, the `ParseUtter` function matches entire words only. For example if the word `"telephone"` appears in a template's `lexicon` slot, that template is not matched when the word `"phone"` appears in the input string.

The Assistant's matching process is case insensitive; thus, if the word `"phone"` appears in a registered template's `lexicon` slot, that template is matched when `"Phone"` or `"phone"` appears in the input string.

If absolutely nothing in the input string is matched, the return result of the `ParseUtter` function is unspecified. However, if any word in the input string is matched, the `ParseUtter` function returns a frame containing frames created from the appropriate templates and additional information about the matched words. For more information about the result frame returned by the `ParseUtter` function, see the description of this function in *Newton Programmer's Reference*.

When none of the words in the input string matches an action template, the Assistant may use the information it did match to try to determine a likely action. For example, when the user enters the phrase `"buzz 555-1234"`, the Assistant does not match the word `"buzz"` to an action template but it can identify

"555-1234" as having the format of a telephone number. Based on that information, the Assistant creates a task frame from the built-in call_act template and displays a call slip to prompt the user for additional information.

When no action template is matched, or more than one is matched, the Assistant compares the number of matched target templates for each task with the total number of targets for that task. The Assistant creates the task frame from the task template having the highest percentage of targets matched and then invokes that task frame's PostParse method.

If two or more tasks have the same percentage of matched target templates, then the Assistant displays the slip containing the Please picker and prompts the user to choose an action. Under these circumstances, the Please picker contains only words representing the candidate templates.

To permit more natural interaction, the Assistant ignores words that do not appear in any registered template's lexicon. Rather than limiting the user to terse commands, the Assistant extracts meaningful words from phrases such as "Make a call to Bob at work" and ignores the others. For example, the words "call", "Bob" and "work" are meaningful to the Assistant because they appear in the lexicons of registered templates. (In this case, the templates are supplied and registered by the system.) On the other hand, because the words "a", "to", and "at" do not appear in the lexicons of any registered templates, they are not matched and are therefore ignored.

Unmatched words appear in the noiseWords slot of the frame that the ParseUtter function returns. Your PostParse method may be able to use the contents of this array to determine further information about the user's intended action. For example, if there are no entries for Bob in the Names soup, the word "bob" is not matched and is returned as an element of the noiseWords array. The word "to" is also likely to show up in this array. Because words are added to this array in the order they were parsed, your PostParse method can extract the word following "to" from the noiseWords array and attempt to use it as a target. The recommended action in this situation is to use this information to fill out a task slip that is displayed to the user for confirmation.

When a word appears in the lexicon of more than one template, it can cause the Assistant to match the wrong template. For example, two games might both register action templates having the word "play" in their lexicon, or you might attempt to register a template that duplicates a word in the lexicon of one of the system-supplied templates. A strategy for resolving these kinds of conflicts is described in "Resolving Template-Matching Conflicts" (page 18-13).

## The Signature and PreConditions Slots

Your task template must define two slots, called `signature` and `preConditions`, which store arrays of templates and symbols, respectively.

The `signature` slot holds action templates and target templates that must be matched to complete the primary action. The `preConditions` slot specifies the names of slots that the Assistant creates in the task frame as templates in the `signature` slot are matched.

Each element of the `preConditions` array is related to the corresponding element of the `signature` array. Specifically, an element of the `preConditions` array specifies the name of the slot that the Assistant creates in the task frame when the template in the corresponding element of the `signature` array is matched.

For example, to send a fax the Assistant needs frames representing the action of faxing, a fax number, the name of the person to whom the fax is sent, and the time at which the fax is to be sent. The `signature` slot in the following code fragment specifies by name the templates required to create these frames.

```
{...
// example: when fax_number is matched, Assistant creates
// a 'number slot in task frame & puts target frame in it
signature: [fax_action, fax_number, who_obj, when_obj],
preConditions: ['action, 'number, 'recipient, 'when],
...}
```

The corresponding elements of the `preConditions` slot specify the names of slots that the Assistant creates in the task frame to hold the frames created as the templates in the `signature` slot are matched. For example, the `preConditions` array in the previous code fragment specifies that the Assistant creates slots named `action`, `number`, `recipient` and `when` in the task frame as necessary.

Continuing with the example based on the previous code fragment, when the Assistant parses the input phrase `"fax Bob"`, it matches the word `"fax"` to the `fax_action` template in the first element of the `signature` array and creates an action frame from this template. The Assistant places this action frame in an `action` slot (named for the symbol in the first element of the `preConditions` array) that it creates in the task frame. Similarly, the Assistant creates a target frame when the word `"Bob"` is matched to the `who_obj` template in the third element of the `signature` array. The Assistant places this target frame in a `recipient` slot (named for the symbol in the third element of the `preConditions` array) that it creates in the task frame.

Words representing elements of the `signature` array do not necessarily need to appear in the input string in order to be matched to a template; for example, your `PostParse` method might supply Bob's fax number by finding it in the Names soup.

**IMPORTANT**
Soup formats are subject to change. Applications that rely on soup formats risk incompatibility with future versions of Newton software. To avoid future compatibility problems with soup format changes, you should use the methods provided by the built-in applications (if any exist) or the global functions `GetSysEntryData` and `SetSysEntryData` to get or change entries in any of the built-in soups. They allow you to get and set the values of slots in a soup entry. ▲

Familiarity with Chapter 1, "Overview," Chapter 5, "Stationery," and Chapter 11, "Data Storage and Retrieval," of this manual is particularly valuable in reading this chapter.

**Note**
Future Newton devices may not include all the built-in applications described in this chapter. ◆

# Names

This section describes the application program interface (API) to the Names application. The Names application manages a database of people and places. It presents information either as a business card, or as a list of all the available information. These two views are shown in Figure 19-1.

## About the Names Application

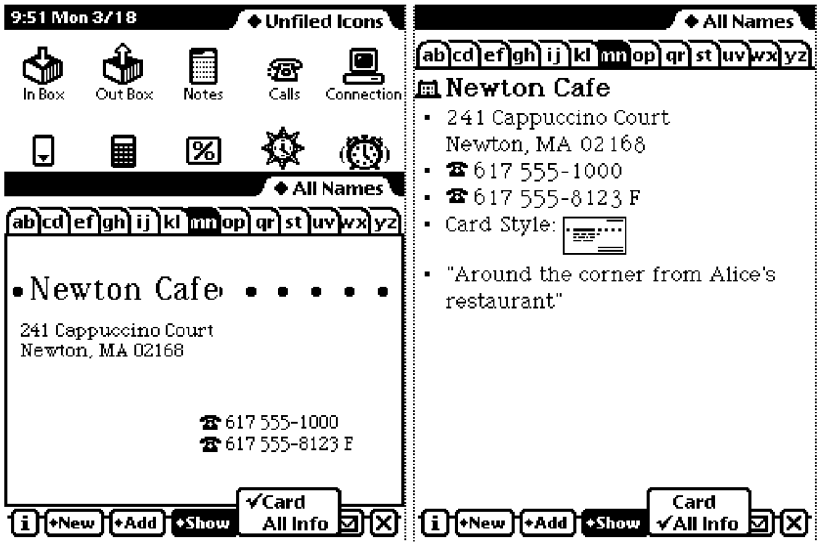
The Names application is built with the NewtApp framework using data definitions (commonly called "dataDefs") and view definitions (commonly called "viewDefs"). This architecture allows extensibility—the addition of new data views, card types, and card layout styles—without altering the Names application itself. For more information on dataDefs and viewDefs, see Chapter 4, "NewtApp Applications," and Chapter 5, "Stationery."

The Names application interface allows you to programmatically add complete cards and add information to an existing card. In addition, several Names methods let you access information in a Names soup entry.

The Names application can be extended by adding auxiliary buttons, as described in "Auxiliary Buttons" beginning on page 19-36.

The application is called Names, because that is what the user sees, but in programming the term "cardfile" is used interchangeably and appears in the code.

**Figure 19-1** Names application Card and All Info views




## Names Compatibility

All the Names methods, variables, and constants are new in this version. The `'group`, `'owner`, and `'worksite` types are new. The `'person` and `'company` types include many new slots.

The Names application converts incoming soup entries that contain 1.x data into soup entries that conform to the 2.0 data format. Conversion includes adding particular slots to soup entries and, in some cases, removing slots. The data conversion occurs when the Newton receives 1.x entries by beaming, synchronizing using Newton Connection, restoring from a backup, or updating a 1.x card to 2.0.

A user can beam a card created in version 2.0 to a Newton running an earlier version of the system and it will read and edit the card, and *vice versa*.

In addition to changes in the programmatic interface, the 2.0 version has extensive changes in the user interface to accommodate the increased number of card types, layout styles, and data.

# Using the Names Application

This section describes

- adding a new type of card to the Names application
- adding a new data item to a card
- adding a new card layout style
- using the Names methods
- using the Names soup
- using two protos with pickers for personae and emporia

## Adding a New Type of Card

The New button on the Names status bar creates its picker by looking at the registered dataDefs for the Names application. All dataDefs whose `superSymbol` slot is set to `'Names` show up in the New picker. When the user picks a choice, the `MakeNewEntry` routine defined for that dataDef is called.

Built-in choices on the New picker include Person, Company, and Group. You can create a new type of card for the Names application by supplying a new data definition.

In addition to the usual slots found in a dataDef frame, Names dataDefs contain two special slots, `overviewIcon` and `viewsToDisplay`.

These slots are described in "Names Data Definition Frame" (page 16-2) in *Newton Programmer's Reference*. For information on dataDefs in particular, and stationery in general, see Chapter 5, "Stationery."

## Adding a New Data Item

The Add button on the Names status bar allows the user to add new items of information to a card, such as a phone number or an address for a person. There is a Custom choice on the Add picker (pop-up menu), through which the user can create special data items that contain a simple text field. However, you can programmatically add new choices to the picker by creating and registering new view definitions with the Names application.

The Add button creates its picker from the viewDefs registered for the card type of the current card. Of these, only viewDefs whose `type` slot is set to `'editor` show up in the Add picker.

Names viewDefs must contain a special slot called `infoFrame`, in addition to those slots required of all viewDefs. The `infoFrame` slot is described in "Names View Definition Frame" (page 16-3), and the slots common to all viewDefs are described in "viewDef Frame" (page 4-1) in *Newton Programmer's Reference*.

Here is an example of an `infoFrame` for a Names viewDef defining a view that has two fields, `Make` and `Model`:

```
infoFrame:{checkPaths: '[carMake, carModel],
           checkPrefix: '[true, [pathExpr: carInfo]],
           stringData: nil,
           format: "^?0Make: ^0\n||^?1Model: ^1||" }
```

When chosen from the Add picker the first time, this view initially fills in the `carMake` and `carModel` slots in the soup entry with the user's entries. If chosen again, this view creates an array called `carInfo` containing one frame for each additional data set. These frames would look like this:

{`carMake:` *make*, `carModel:` *model*}

The reason this is necessary is that after information for the first car is entered, the soup entry will contain the slots `carMake` and `carModel`. The information for the second car could not also be stored in the `carMake` and `carModel` slots of the soup entry. Instead a `carInfo` slot is added to the soup entry, this slot holds a frame containing `carMake` and `carModel` slots.

When a view from the Add picker is instantiated, the system creates a slot called `selectedPath` in the view that was instantiated. This slot is set to the path expression where data should be entered (or to `nil` if the data should be entered directly into the soup entry). For example, when chosen from the Add picker the first time, the view in this example would have its `selectedPath` slot set to `nil`, meaning that the information should be put directly into the soup entry. When chosen from the picker the second time, the `selectedPath` slot is set to `[pathExpr: carInfo, 0]`, to indicate that the new car information should go into the first frame in the `carInfo` array. The third time, `selectedPath` is set to `[pathExpr: carInfo, 1]`, and so on.

## Adding a New Card Layout Style

When the "Card" layout is selected in the Show picker, the Names application looks at the `cardType` slot of the current card to determine which kind of business card layout to use for that card. You can create new viewDefs and register them with the Names application to use a custom card layouts. Card viewDefs must have the `type` slot set to `'bizcard`, and must contain a `bizCardNum` slot and a `bizCardIcon` slot.

The `bizCardNum` slot contains an integer that corresponds to the value stored in the `cardType` slot of the card entries. The values 0-6 correspond to the business card layouts that are built into the system. You should pick integers over 1,000 to use as a `bizCardNum`, and register your number with Newton DTS.

The `bizCardIcon` slot contains an icon representing the new layout, to be shown in the Card Style view, where the user can change the type of card layout to use for a particular card. This icon should be 38x23 since this is the size of the built-in icons.

## Adding New Layouts to the Names Application

The Show picker allows the user to chose from the two built-in layouts: Card and All Info. You can programmatically add a new layout by calling the Names method `AddLayout`. It takes a single parameter, which is the layout to add. This layout should be based on the `newtLayout` proto, and must include the following slots:

| | |
|---|---|
| `name` | A string shown in the Show picker. |
| `symbol` | A symbol, which includes your developer signature, uniquely identifying this layout. This symbol must be passed to the `EnsureInternal` function. |
| `type` | Set this slot to the symbol `'viewer`. |
| `protection` | Set this slot to the symbol `'private`. |

For more details see `AddLayout` (page 16-8), and its counterpart function `SafeRemoveLayout` (page 16-13) in *Newton Programmer's Reference*.

## Using the Names Methods and Functions

There are a number of methods provided by the Names application. To obtain a reference to the Names application in order to send these methods, use the following code:

```
GetRoot().cardfile
```

Note that future Newton devices may not include the Names application. You should therefore check for the existence of the Names application before trying to access it. Use the following code to test for this:

```
if GetRoot().cardfile then ...
```

The methods provided allow you to

- add a new card (`AddCard`)
- add data to an existing card (`AddCardData`)
- turn to a particular card if Names is open (`ShowFoundItem`)
- open the Names application to a particular card (`OpenTo`)
- replace ink data in a card with a string (`ReplaceInkData`)
- add an action to the Action picker (`RegNamesRouteScript`)

- get information from Names soup entries
  - for credit/phone card information (`BcCreditCards`)
  - for custom fields information (`BcCustomFields`)
  - for e-mail address information (`BcEmailAddress`)
  - for e-mail network information (`BcEmailNetwork`)
  - for phone number information (`BcPhoneNumber`)

These functions and methods are all described in *Newton Programmer's Reference*.

## Using the Names Soup

The Names application stores its data in the `ROM_CardFileSoupName` ("Names") soup. Entries in this soup are frames for either a person, an owner, a group, a company, or a worksite card.

The soup formats for each of these types of entries are described in "Names Soup Format" (page 16-15) in *Newton Programmer's Reference*. A list of these frames is available in the Summary; see "Names Soup" (page 19-49).

To avoid future compatibility problems with soup format changes, you should use the Names methods provided for getting or setting information in the Names soup. If none is available for getting the information you want, use the global functions `GetSysEntryData` and `SetSysEntryData` to get or change entries in any of the built-in soups. They allow you to get and set the values of slots in a soup entry. If you don't use these functions to get and set entry slots in the built-in soups, your application may break under future versions of system software.

## Using the Names Protos

The Names application uses two protos which are available to you: `protoPersonaPopup` and `protoEmporiumPopup`. These protos provide pickers that maintain lists of **personae** and **emporia.** Personae are people who use the Newton device, and emporia are places where the Newton device is used.

Note that you can get the information on the current `owner` and `worksite` from the `user configuration` data stored by the system. This data is described in "System Data" beginning on page 19-44.

### protoPersonaPopup

This proto is used for a picker that lets the user maintain and switch between different owner cards, or "**personae**." Here's an example:

```
areaCode: "503",
region:"OR",
airport:"PDX"}
```

### Using Longitude and Latitude Values

To calculate the latitude or longitude of a location, create and use the following function:

```
CalcLngLat := func(dgrs, min, secs, westOrSouth) begin
   local loc;
   loc := dgrs / 180 + min / (180 * 60) + secs
         / (180 * 60 * 60);
   loc := rinttol(loc * 0x10000000);
   if westOrSouth then
      loc := 0x20000000 - loc;
   loc;
end;
```

The built-in utility functions `LatitudeToString` and `LongitudeToString` return a string representation of an encoded integer latitude or longitude value. For information on these functions see *Newton Programmer's Reference*.

### Setting the Home City

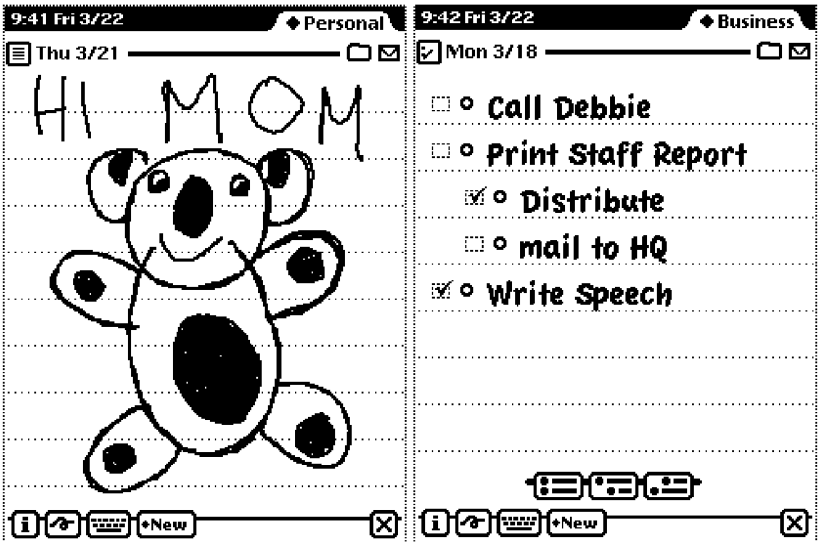
The `SetLocation` method sets the home city. It takes a single parameter *whichCity* which is the same as the *newCityFrame* parameter of the `NewCity` method; see "Adding a City to a Newton Device" beginning on page 19-29. The following code makes Los Angeles the home city:

```
GetRoot().worldClock:SetLocation( GetCityEntry
      ("Los Angeles") [0] );
```

## Notes

This section describes the Notes API. The Notes application uses three types of stationery: regular notes, checklists, and outlines. Figure 19-6 shows a note and a checklist; an outline (not shown) is like the checklist without the checkboxes.

**Figure 19-6** Notes note and Checklist views




## About the Notes Application

Notes is a simple application based on NewtApp that allows the user to create new stationery, scroll up and down, route and file notes, and scan an overview.

The Notes API is limited to a few methods which allow you to create new notes.

The Notes application can be extended by adding auxiliary buttons, as described in "Auxiliary Buttons" beginning on page 19-36.

The name of the application is Notes, which is what the user sees, but in programming the term `paperroll` is also used and appears in the code.

### Notes Compatibility

There are some anomalies in converting ink from system software 2.0 to earlier versions of the system. Version 2.0 ink text is converted to straight ink when viewed in 1.x versions. Paragraphs with mixed regular and ink text are converted so that the regular text loses any styles; for example in 1.x versions, it becomes 18-point user font. Any paragraph that contains ink text is reflowed in 1.x versions so that line layouts (breaks) are different from the original. This means that the paragraph may grow taller. Ink works converted from 2.0 to 1.x straight ink appear in the size originally drawn, not in the 2.0 scaled size.

# Exhibit 22

# THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

# Exhibit 23

# Intentionally Left Blank

# Exhibit 24

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

| | | |
|---|---|---|
| ARENDI S.A.R.L., Plaintiff, vs. MOTOROLA MOBILITY LLC f/k/a/ MOTOROLA MOBILITY, INC., Defendant. | ) | No. 12-1601-LPS |
| ARENDI S.A.R.L., Plaintiff, vs. GOOGLE LLC, Defendant. | ) | No. 13-919-LPS |

VIDEO-RECORDED GOOGLE MEET DEPOSITION

UPON ORAL EXAMINATION OF

EARL SACERDOTI, PhD

** CONFIDENTIAL, OUTSIDE COUNSELS' EYES ONLY **

9:03 A.M.
WEDNESDAY, JANUARY 20, 2021
(ALL PARTICIPANTS AT THEIR RESPECTIVE LOCATIONS)

Reported by: Tami Lynn Vondran, CRR, RMR, CCR
Washington CCR No. 2157, Oregon CSR No. 20-0477

A P P E A R A N C E S

(ALL PARTIES APPEARING REMOTELY)

FOR THE PLAINTIFF AND WITNESS:

BURTON DeWITT
Susman Godfrey LLP
1000 Louisiana Street, Suite 5100
Houston, Texas 77002
713.650.4335
bdewitt@susmangodfrey.com

FOR THE DEFENDANT:

ROBERT UNIKEL
Paul Hastings
71 South Wacker Drive, 45th Floor
Chicago, Illinois 60606
312.499.6030
robertunikel@paulhastings.com

ALSO PRESENT:

ALAN MORGAN, Videographer
MINDY MARSHALL, Paul Hastings

videographer. The court reporter is Tami Vondran.

I am not authorized to administer an oath. I am not related to any party in this action, nor am I financially interested in the outcome.

If there are any objections to proceeding, please state them at the time of your appearance, beginning with the noticing attorney.

Will all present please state their name and affiliation for the record.

MR. UNIKEL: Robert Unikel of Paul Hastings LLP on behalf of Defendants Google and Motorola. Also appearing on this remote deposition is Mindy Marshall, who's also with Paul Hastings LLP on behalf of Google and Motorola.

MR. DeWITT: Burton -- sorry. Burton DeWitt of Susman Godfrey LLP appearing on behalf of Plaintiff Arendi S.A.R.L. and Deponent Earl Sacerdoti.

THE COURT REPORTER: My name is Tami Vondran. I am a Washington State certified court reporter.

Counsel, if you have an objection to the remote deposition or to me swearing in the witness remotely, please state so now.

Hearing no objection and seeing the witness, I will swear in the witness.

EARL SACERDOTI, PhD,
sworn as a witness by the Certified Court Reporter,
testified as follows:

THE COURT REPORTER: Thank you.
Please proceed.

EXAMINATION

BY MR. UNIKEL:

Q. Good morning, Dr. Sacerdoti.

A. Good morning.

MR. UNIKEL: Before I begin asking you questions, I just want to make a statement for the record. This is a deposition that is being conducted in two cases, the case of Arendi versus Google and the case of Arendi versus Motorola. It is a single seven-hour on-the-record deposition that is being conducted and is usable in both cases equally. There will not be two separate portions of the deposition, one for Google and one for Motorola.

Just wanted to confirm with Mr. DeWitt that that is accurate and that we're all in agreement on that.

MR. DeWITT: In agreement. Thank you.

MR. UNIKEL: Great. Thank you.

apologize, but I'd have to think through what I intended to mean here to determine whether I actually meant the second computer or I meant the second computer program.

Q. Okay. So sitting here today, you're not sure in the third option, at least, whether you meant by a second computer, as it's written, or by a second computer program, is that -- am I understanding your correctly?

A. That's correct. I need to -- need to analyze this further and think about it further so I'm sure I know what I said.

Q. Okay.

A. I apologize for that. It's potentially an editing error.

Q. Okay. Looking then at the first two possibilities that you list here, how is having an input device set up independently by the operating system different than having an input device set up by a first computer program displaying the document?

A. As I stated a few minutes ago, one of ordinary skill in the art would not consider an operating system to be an application or a stand-alone computer program.

Q. Even if the -- that first computer program that's displaying the document is utilizing features of the operating system?

MR. DeWITT: Objection to form.

A. I think you already asked that question, and as I recall, I said that what you would be describing would be the -- would be an application invoking functionality of the operating system.

Q. (BY MR. UNIKEL) Isn't that the same thing as having the first computer program actually doing what the operating system is permitting it to do?

MR. DeWITT: Objection to form.

A. No, that's the first computer program using functionality provided by the operating system.

Q. (BY MR. UNIKEL) And that's a different thing, in your view, than having the first computer program actually perform the action itself?

A. In that case, the first computer program is invoking the functionality. It's not -- it's not containing the functionality.

Q. Well, how about the second option, how is having an input device set up by a second computer program responsible only for linking together the first computer program and a third computer program, and passing information between them, different than having the first computer program actually set up the input device?

A. Well, in that case, the input device would be

Q. And you say, quote, this is tantamount to arguing that architecture in computer systems is irrelevant. It was well known in the art at the time, and still is today, that the assignment of functions to particular programs is one of the most critical design factors that one of skill in the art must consider, unquote.

Do you see that?

A. I do.

Q. Do you still agree with this statement in paragraph -- these statements in paragraph 828?

A. I'm sorry. I've been reviewing the paragraph of Dr. Fox in which that sentence that I quoted was embedded.

Please ask your question again. I'm prepared to answer it now.

Q. Yes.

Is it still your view that, quote, It was well known in the art at the time, and still is today, that the assignment of functions to particular programs is one of the most critical design factors that one of skill in the art must consider, unquote?

A. Yes.

Q. And why do you believe that the assignment of functions to particular programs is one of the most

A. Does quite a number of things or permits the user to enter a user command that initiates an operation that does quite a few things. But, yes, there is a -- there is an element of Claim 1 that incorporates a reference to that input device configured by the first computer program.

Q. And I'm just trying to make sure I understand. When you state in paragraph 828 that the assignment of functions to particular programs is one of the most critical design factors, might one example of such an assignment be assigning to the first computer program in the '843 claims that it configure the input device?

A. Yes.

Q. And why is that kind of an assignment of function that -- the requirement that the first computer program set up the input device, why is something like that a critical design factor?

A. I don't think I can answer that by taking -- taking that portion of the limitation out of context from the entirety of Claim 1 of the '843 patent.

In its entirety, the claim describes an allocation of functions to -- to different computer programs and that allocation of functionality to the computer -- to the various computer programs, in my view, is a substantial aspect of the value of the '843

Exhibit S12, which is Luciw '735 patent.

(Exhibit No. S12 marked for identification.)

Q. (BY MR. UNIKEL) Let me know when you have that up in front of you.

A. I have that up in front of me now.

Q. And this is a United States Patent No. 5,644,735; correct?

A. Correct.

Q. And one of the inventors is a gentleman named William W. Luciw, L-u-c-i-w; correct?

A. Correct.

Q. Is that the patent that you are discussing in paragraph 252 of your report?

A. Yes, it is.

Q. Now, in the last -- at the end of paragraph 252, there is a sentence that begins "Moreover."

Do you see that?

A. Yes.

Q. It's near to the bottom.

A. Yes.

Q. Great.

You state, quote, Moreover, to the extent that Dr. Fox conceives of the phone slip window or its smart fields as the document, I disagree because it is my

opinion that neither is "a word processing, spreadsheet, or similar file into which text can be entered." For example, both the phone slip and the smart field embedded in it are transitory interfaces in windows that are established during and for the assist process (e.g., Luciw, 10:22-26). Like a search field, they are established for the entry of a command to a computer, unquote.

Do you see that language from your report?

A. I do.

Q. Do you agree still with that opinion as expressed in paragraph 252?

A. In the context of the rest of paragraph 252, and in the context of the rest of this discussion of the Luciw's patent, yes, I -- I agree with what's written there.

Q. And to make sure I know what you're referring to, in Exhibit S12, which is the Luciw patent, if you could find Figures 4B and 4C for me, please.

A. Yes, I've found them.

Q. All right. Does Figure 4B show either the phone slip window or its smart field that you are referring to in your expert report, paragraph 252?

MR. DeWITT: Objection to form.

A. So Figure 4B shows an example of an implicit

assist operation with a phone slip window having a smart name field.

Q. (BY MR. UNIKEL) And it is your opinion that what is shown as a phone slip window or its smart field is not a document within the meaning of the '843 patent; is that correct?

A. That's correct.

Q. Am I correct that a user can enter text --

MR. DeWITT: I've lost the -- we can go off the record. But I appear to have lost the live stream of the transcript.

MR. UNIKEL: Okay. Why don't we go off the record for a moment.

THE VIDEOGRAPHER: Going off the record. The time is 5:08 p.m.

(Brief break taken.)

THE VIDEOGRAPHER: We are back on the record. The time is 5:09 p.m.

Q. (BY MR. UNIKEL) Dr. Sacerdoti, before we had technical difficulties, we were discussing the Luciw patent; correct?

A. Correct.

Q. And am I correct that a user can, in fact, enter text into the phone slip window that you are referring to in paragraph 252 of your report, as you

understand it?

A. The Luciw patent discloses that the, quote -- I'm quoting from Column 10, line 26, "Once the particular window 170 is presented to the user, the name ISAAC can be handwritten into the particular smart field 175."

I would not consider handwritten -- handwriting to be understood as text necessarily.

Q. Am I correct that according to what you just read, a user can handwrite names into the smart field that you have just referred to -- I'm sorry, into the phone slip that you just referred to?

A. My understanding is that it's entered -- that the handwriting is entered into a smart field labeled name that is within the -- the slip that was -- that I just alluded to.

Q. And is it your understanding the system in Luciw is then designed to recognize the text that was handwritten into the smart field?

A. It's designed to recognize the handwriting that was written into the smart field.

Q. And according to your opinion in paragraph 252, you do not believe the phone slip window or its smart field can be a document within the meaning of the '843 patent; is that correct?

A. That's correct.

Q. And what about the phone slip window or its smart field in Luciw leads you to believe that it is neither a, quote, word processing, spreadsheet or similar file into which text can be entered?

MR. DeWITT: Objection to form.

A. The phone slip that is discussed in the Luciw patent that we've been referring to is a transitory interface in windows that are established during and for the assist process.

Q. (BY MR. UNIKEL) And what is the -- were you done? I'm sorry.

A. Yes.

Q. What is the import of the fact that it's a transient interface, as you describe it?

A. As I understood -- I'm going to have to go back and find the court's construction again so that I can quote it accurately.

Document has been construed as a word processing, spreadsheet or similar file into which text can be entered. And the phone slip, in my opinion, is not a similar document to a word processing or a spreadsheet file.

Q. What criteria did you use to come to that conclusion?

A. Would you please indulge me and point me back to the paragraph we were talking about?

Q. Sure.

A. I went to see the court's construction and now I'm lost.

Q. Paragraph 252, please.

A. Thank you very much.

Q. Sure.

A. So we need to look at all the ways in which a document is -- is employed or referenced within the totality of the claims of the '843 patent in order to answer that question.

So I'm going to apologize, but I'm going to need to find the '843 patent here --

Q. Okay.

A. -- and that will refresh my recollection of the claims.

Sorry. I apologize. I found a Hedloy patent, but it was the wrong one. Here we go. I've got a large number of unlabeled tabs at the top of my browser, at this point, with all these documents that we've been opening.

Q. Understood.

A. Okay. So in the preamble of the claim, it's describing a computer-implemented method for finding

data related to the contents of a document. The document -- the first limitation describes the document being displayed electronically using the first computer program.

There's an analysis for determining if the first information is of, at least, one of a plurality of types of information that can be searched for in the second limitation. And the information source that is searched in the operation disclosed in the fourth limitation must be external to that document.

Q. And just to be clear, I'm -- I just want to understand what the criteria that you applied to conclude that the phone slip or smart field in Luciw was not a document within the meaning of the '843 patent.

A. As I stated before, for one reason the -- that phone slip or the smart field within it are not, in my opinion, analogous to a word processing file or a spreadsheet file into which text can be entered.

I don't recall the court's discussion supporting its construction. But the implication is or was that the document, as the court has construed it, would be something that would be potentially saved into a file, which I understand is a nontransitory element of storage.

And the -- there's nothing disclosed in Luciw,

that I can recall, that discloses that phone slip being treated as a file in the sense that the -- the court's construction uses that -- uses that term.

Q. Is there any other reason that you believe that a phone slip of the type shown in Luciw is not analogous to a word processing, spreadsheet or similar file?

A. There may well be. I'm not -- I'm not -- as I'm sitting here, I'm not prepared to give you an exhaustive list of all the reasons why a phone slip might differ from a word processing, spreadsheet or a similar file.

Q. Okay. One last paragraph that I'd like you to take a look at, if you would, please, in your report paragraph 225.

A. Okay. I'm there.

Q. In paragraph 225 of your report, do you see you are discussing the Newton MessagePad Pro -- I'm sorry -- MessagePad 2000 prior art?

A. This paragraph refers to it in general as the Newton, I believe. Identified that as the specific version of the device somewhere prior in my -- yes, it's -- yes.

Q. And in particular -- and would you please take a read of paragraphs 224 and 225 to yourself, and let me

opinion as expressed here?

A. Yes.

Q. And which claim limitation are you referring to that you believe Newton does not meet?

A. I've presented this analysis in the context of the claim limitation while the document is being displayed, analyzing in a computer process, first information from the document to determine if the first information is at least one of a plurality of types of information that can be searched for in order to find second information related to the first information.

Q. And in the next sentence of paragraph 225, you state more specifically, quote, That is, that the Newton thinks that 555-1000 might be a phone number does not mean that a phone number can be searched for, unquote.

Do you see that?

A. I do.

Q. Again, do you still agree with your statement there?

A. I do.

Q. And why is it the case that if Newton identifies 555-1000 as a possible phone number that it does not necessarily mean that the phone number can be searched for?

A. Because those are independent functionalities

of a computing device.

Q. Well, how would the Newton determine that a possible phone number like 555-1000 is something that can be searched for?

A. There are at least two ways that that might be done that I can think of off the top of my head. One is to actually perform the search and have that be incorporated into the analysis, and the other is to have a known component of the Newton that is known to search for phone numbers.

Q. And it's your understanding that Newton did not do either of those things to determine whether a possible phone number could be searched for?

A. It's my understanding that Dr. Fox has not demonstrated that either of those alternatives, that I presented, is the case.

Q. And is it your view that identifying text as a phone number is not enough to meet the claim element that you've identified here?

A. On the face of it, it's not because there's a -- there's also a requirement that the type of information can be searched for -- excuse me, that the information can -- the information is of a type that can be searched for.

Q. And merely identifying a phone number as a

phone number is not enough, in your view?

A. As I said before, there would have to be some known functionality on the Newton that searched for phone numbers. You say, Ah-ha, it's a phone number, but if you don't -- if you don't have the capability of doing the search, then it's not going to meet the limitation.

MR. UNIKEL: All right. Why don't we take a very fast break to make sure that there's nothing else that I need to cover.

In the meantime, why don't we go off the record and then, Alan, you can tell me where we are on the record.

THE VIDEOGRAPHER: Going off the record. The time is 5:30 p.m.

(Brief break taken.)

THE VIDEOGRAPHER: We are back on the record. The time is 5:36 p.m.

(Exhibit No. S13 marked for identification.)

Q. (BY MR. UNIKEL) Sir, if you could please go to the shared exhibit folder and find the exhibit marked S13, the '843 IPR Arendi Preliminary Response, please.

A. I have it up.

Q. You are aware, are you not, that the '843 patent was subject to certain inter partes review

# Exhibit 25

**From:** John Lahad <jlahad@SusmanGodfrey.com>
**Sent:** Friday, March 8, 2019 11:25 PM
**To:** Unikel, Robert <robertunikel@paulhastings.com>
**Cc:** Max Straus <MStraus@susmangodfrey.com>; Eve H. Ormerod <eho@skjlaw.com>; Marek, Michelle <michellemarek@paulhastings.com>; Neal Belgam <nbelgam@skjlaw.com>; Marek, Michelle <michellemarek@paulhastings.com>; Moore, David E. <dmoore@potteranderson.com>; Palapura, Bindu A. <bpalapura@potteranderson.com>; Richard A. Wojtczak <rwojtczak@susmangodfrey.com>; Seth Ard <sard@susmangodfrey.com>; Steve Susman <SSusman@SusmanGodfrey.com>
**Subject:** [EXT] RE: Summary of Today's Meet-and-Confer Discussion

Rob,
Thanks for your time today. This is generally right, but I think there is a disconnect as to the last two paragraphs. I agreed to look at the 843 chart re the Explore functionality and follow up with you. That is all I agreed to do at this point.

As to Google's and Motorola's production of documents, you said additional documents were forthcoming. You did not say and I did not understand that the additional document productions would be conditioned on when "it becomes clear what particular app/program/device functionalities are being accused." You clearly have an idea of what is being accused at least as to the representative products/functionality we've charted. You can provide core technical documents as to those devices and that functionality. And it would be just as easy to produce similar documents for prior versions of same.

Happy to discuss this further. Have a pleasant weekend.
Thank you.

JOHN P. LAHAD
SUSMAN GODFREY LLP
713-653-7859 (OFFICE)
713-725-3557 (MOBILE)
CLICK HERE FOR BIO

**From:** Unikel, Robert <robertunikel@paulhastings.com>
**Sent:** Friday, March 8, 2019 5:20 PM
**To:** John Lahad <jlahad@SusmanGodfrey.com>
**Cc:** Max Straus <MStraus@susmangodfrey.com>; Eve H. Ormerod <eho@skjlaw.com>; Neal Belgam <nbelgam@skjlaw.com>; Marek, Michelle <michellemarek@paulhastings.com>; Moore, David E. <dmoore@potteranderson.com>; Palapura, Bindu A. <bpalapura@potteranderson.com>
**Subject:** Summary of Today's Meet-and-Confer Discussion

John et al.,
Thank you for taking the time to speak with us this morning to discuss the issues raised in Google's and Motorola's February 28, 2019 letters regarding Arendi's supplemental initial claim charts. This email serves as a summary of our March 8, 2019 meet and confer.

During today's call, you confirmed that the asserted claims listed in Arendi's cover letter to its supplemental initial claim charts are being asserted *in addition to* any claims that previously were asserted against Google and/or Motorola in 2013. To provide clarity, and to prevent any future misunderstandings, you agreed to provide a complete list of all of the claims that Arendi currently is asserting against Google and Motorola.

In regards to claims 1 and 12 of the '356 patent, which were found to be unpatentable in IPR2014-00450, you stated that these claims were included in the list of asserted claims due to a clerical error and that Arendi would *not* be pursuing claims 1 and 12 of the '356 patent against Google.

In regards to the '854 patent, which is being asserted against Google, you confirmed that you currently are pursuing claims 13, 15, 31, 50, 53, 56 79, and 101 despite the fact that in the institution decisions in IPR2014-00206 and IPR2016-00207, the PTAB expressly concluded that the means-plus-function elements of these claims lacked sufficient disclosure of corresponding structures. However, you agreed to reexamine the institution decisions, and the PTAB's findings concerning insufficient disclosure, and to consider further whether Arendi would continue to assert claims 13, 15, 31, 50, 53, 56 79, and 101 going forward.

In discussing Google's and Motorola's overarching concern regarding the contentions' lack of clarity as to the particular apps/programs being accused of infringement in connection with every accused device (we mentioned, e.g., our uncertainty as to whether apps like Facebook and Facebook Messenger were being accused as used on the accused devices), you confirmed that for the accused Google and Motorola devices, Arendi is only accusing devices based on use of those applications that were *preinstalled* on the Google and/or Motorola devices at the time those devices were delivered to users; you confirmed that you are *not* accusing devices based on the use of apps that were *not* preinstalled on the Google and/or Motorola devices.

With regard to our discussion of the supplementation and clarification of Arendi's exemplary initial claim charts, you indicated that Arendi is not willing to provide individual claim charts for every accused combination of device, operating system, and application/program. We informed you that Arendi is obligated to do so under the Delaware Rules (as explained and applied in the *Personal Audio* decision). Notwithstanding our disagreement, we agreed to seek and discuss a possible compromise solution, in which Google and Motorola might specify a group of combinations of accused devices, operating systems, and accused apps/programs from specific date ranges that Arendi would chart. Assuming that the range of combinations charted is sufficiently broad and diverse, and that the charts provided are sufficiently specific and detailed, such approach might address Google's and Motorola's concerns. We are attempting to formulate a proposal to resolve our dispute and will send that proposal to you as soon as we are able. To be clear, however, unless and until an adequate compromise is reached, Google and Motorola continue to object to Arendi's contentions, as served, and reserve all rights to seek appropriate relief concerning the identified issues with those contentions.

With regard to the contentions provided to Motorola, we pointed out that many of the charts include screen shots of non-Motorola apps/programs being run on non-Motorola devices, which is inappropriate. You agreed to provide revised charts that properly reference combinations that utilize only accused Motorola devices.

Finally, Google and Motorola confirmed that their document production is ongoing and will be supplemented as it becomes clear what particular app/program/device functionalities are being accused.

Please let me know if anything I have stated is incorrect.

Best regards,
Rob Unikel

---

**Robert Unikel** | **Partner**
Paul Hastings LLP | 71 South Wacker Drive, Suite 4500, Chicago, Illinois 60606 | Direct: +1.312.499.6030 | Main: +1.312.499.6000 | Fax: +1.312.499.6131 | robertunikel@paulhastings.com | www.paulhastings.com



**********
This message is sent by a law firm and may contain information that is privileged or confidential. If you received this transmission in error, please notify the sender by reply e-mail and delete the message and any attachments. If you reply to this message, Paul Hastings may collect personal information including your name, business name and other contact details, and IP address. For more information about Paul Hastings' information collection, privacy and security principles please click HERE. If you have any questions, please contact Privacy@paulhastings.com.

**********
This message is sent by a law firm and may contain information that is privileged or confidential. If you received this transmission in error, please notify the sender by reply e-mail and delete the message and any attachments. If you reply to this message, Paul Hastings may collect personal information including your name, business name and other contact details, and IP address. For more information about Paul Hastings' information collection, privacy and security principles please click HERE. If you have any questions, please contact Privacy@paulhastings.com.

# Exhibit 26

# THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

# Exhibit 27

US005859636A

# United States Patent [19]

**Pandit**

[11] **Patent Number: 5,859,636**

[45] **Date of Patent: Jan. 12, 1999**

[54] **RECOGNITION OF AND OPERATION ON TEXT DATA**

[75] Inventor: **Milind S. Pandit**, Beaverton, Oreg.

[73] Assignee: **Intel Corporation**, Santa Clara, Calif.

[21] Appl. No.: **579,568**

[22] Filed: **Dec. 27, 1995**

[51] **Int. Cl.**[6] ..... **G06F 3/00**

[52] **U.S. Cl.** ..... **345/335**; 707/501

[58] **Field of Search** ..... 345/339, 349, 345/352, 353, 335; 707/501

[56] **References Cited**

PUBLICATIONS

"TAPI—The Biggest Roundup Ever Published," Computer Technology, May 1995, p. 55.

"Apple Introduces Internet Address Detectors," Newsbytes, 29 Sep. 1997.

"E–Mail 'Net Differences–. . . ," CommunicationWeek, 6 Jan. 1997 p. 12.

"Internet Update," Newbytes, 10 Sep. 1997.

"McWorld–Apple, Partners Launch Live Objects," Newsbytes, 8 Aug. 1996.

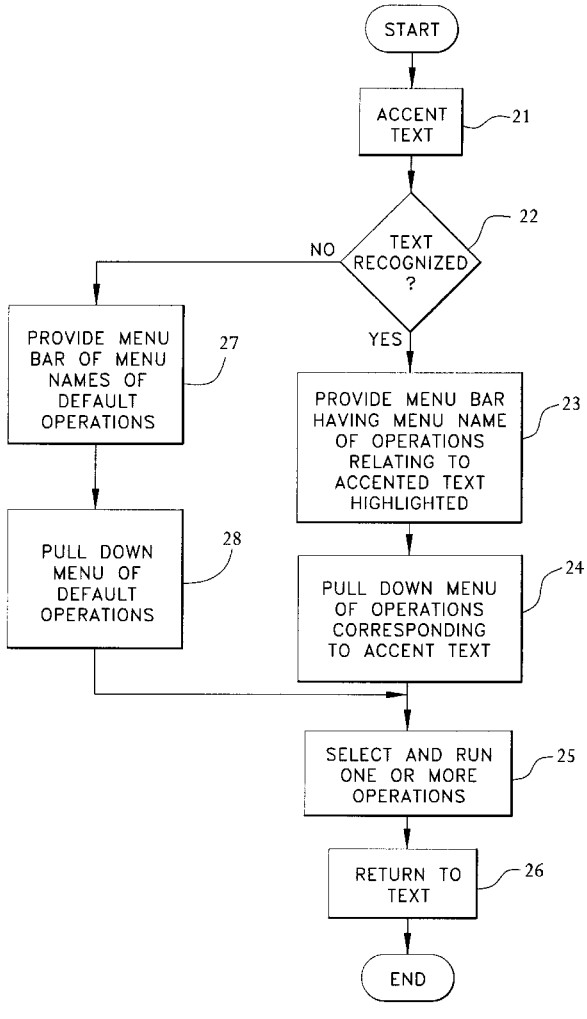
*Primary Examiner*—A. Katbab

*Attorney, Agent, or Firm*—Duane, Morris & Heckscher LLP

[57] **ABSTRACT**

Text of a predetermined class is recognized in a body of text. After recognition, operations relevant to the recognized text may be performed. For example, text such as telephone numbers, telefax numbers, and dates can be recognized in a body of text. Options are provided for selecting and running operations and programs relevant to the recognized text, such as, telephone dialers, telefaxing programs, writable databases etc. Libraries of subroutines are provided for each class of text to be recognized. Each library typically includes a plurality of operations which may be run on the particular class of text. The libraries are recognized at run time of an application, so that additional libraries and operations may be added without a need for recompiling. A single class of text is recognizable in a number of formats without limits as to the origin of the body of text.

**40 Claims, 9 Drawing Sheets**



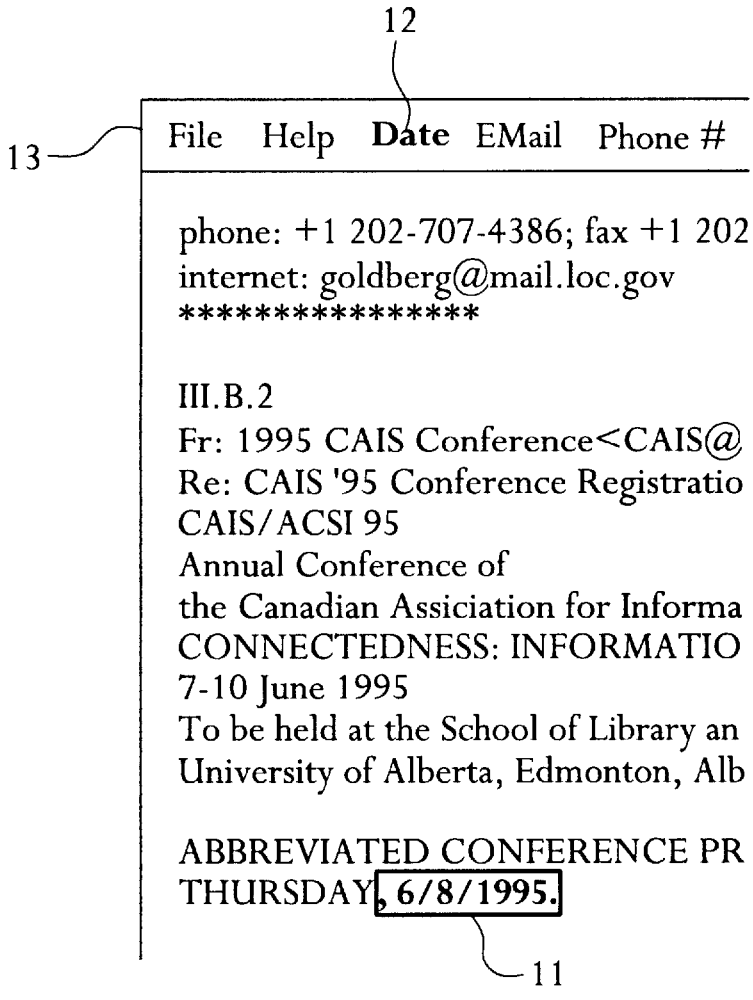

12
13
File Help Date EMail Phone #
phone: +1 202-707-4386; fax +1 202
internet: goldberg@mail.loc.gov
**********
III.B.2
Fr: 1995 CAIS Conference<CAIS@
Re: CAIS '95 Conference Registratio
CAIS/ACSI 95
Annual Conference of
the Canadian Assiciation for Informa
CONNECTEDNESS: INFORMATIO
7-10 June 1995
To be held at the School of Library an
University of Alberta, Edmonton, Alb
ABBREVIATED CONFERENCE PR
THURSDAY, 6/8/1995.
11

FIG. 1a

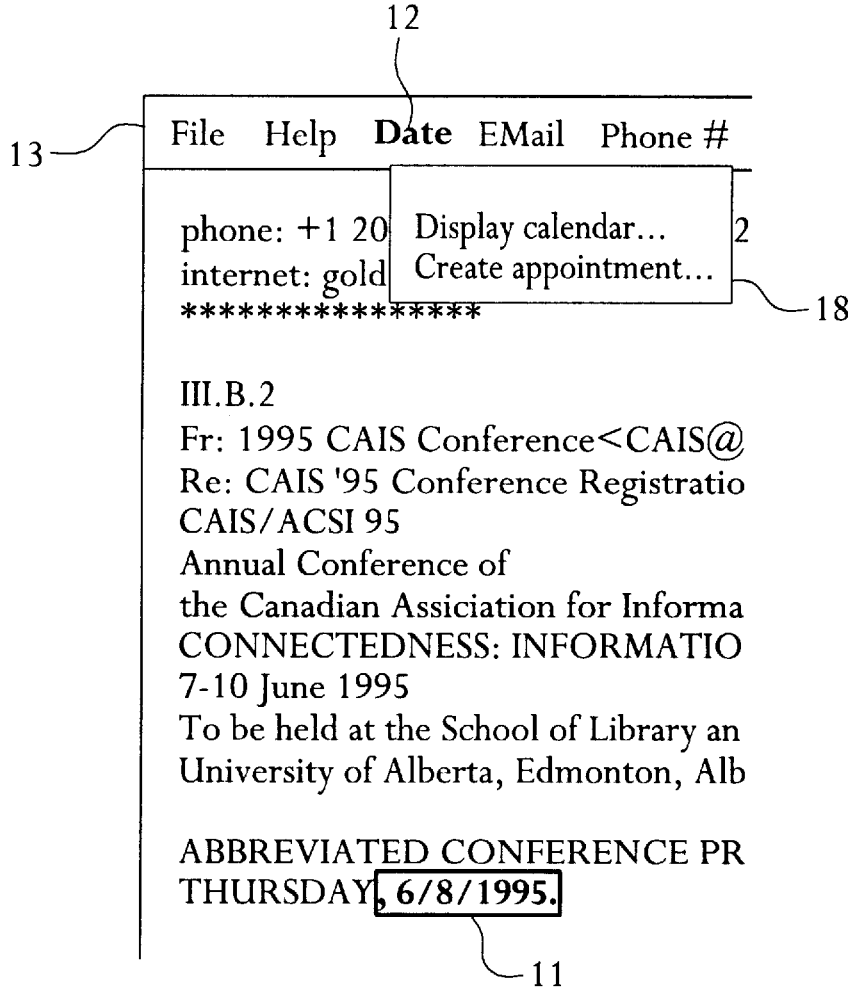

12
13
File Help Date EMail Phone #
phone: +1 20
internet: gold
**********
Display calendar...
Create appointment...
2
18
III.B.2
Fr: 1995 CAIS Conference<CAIS@
Re: CAIS '95 Conference Registratio
CAIS/ACSI 95
Annual Conference of
the Canadian Assiciation for Informa
CONNECTEDNESS: INFORMATIO
7-10 June 1995
To be held at the School of Library an
University of Alberta, Edmonton, Alb
ABBREVIATED CONFERENCE PR
THURSDAY, 6/8/1995.
11

FIG. 1b

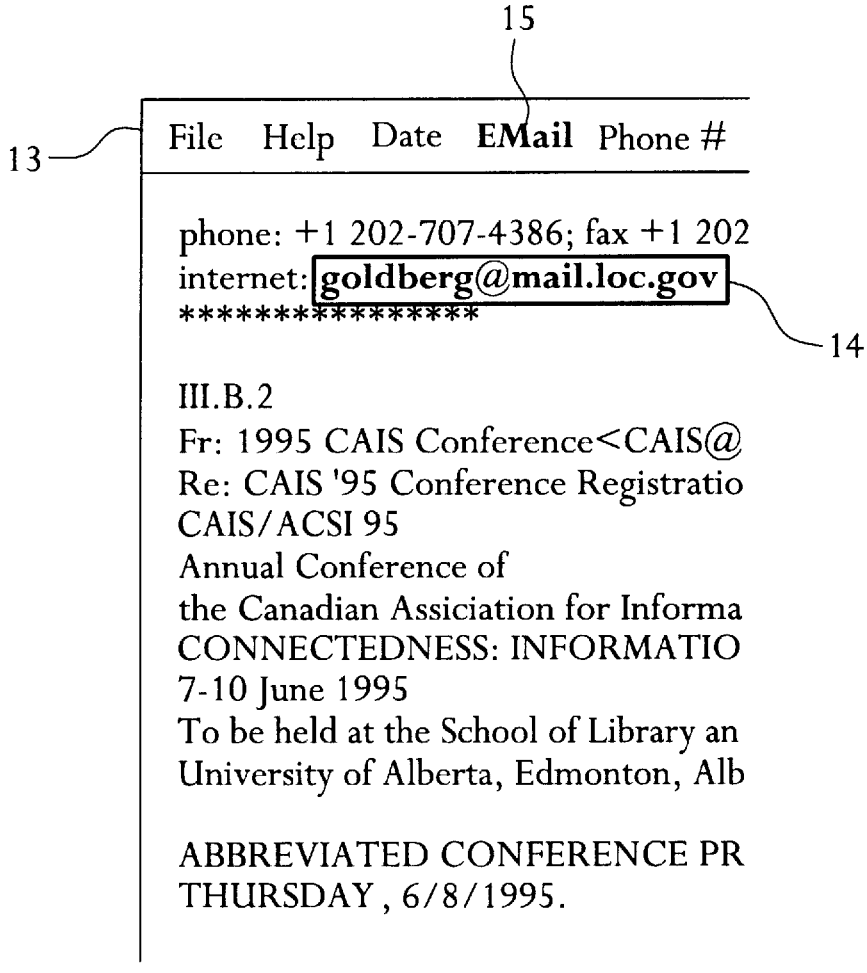

15
13
File Help Date EMail Phone #
phone: +1 202-707-4386; fax +1 202
internet: goldberg@mail.loc.gov
**********
14
III.B.2
Fr: 1995 CAIS Conference<CAIS@
Re: CAIS '95 Conference Registratio
CAIS/ACSI 95
Annual Conference of
the Canadian Assiciation for Informa
CONNECTEDNESS: INFORMATIO
7-10 June 1995
To be held at the School of Library an
University of Alberta, Edmonton, Alb
ABBREVIATED CONFERENCE PR
THURSDAY , 6/8/1995.

FIG. 1c

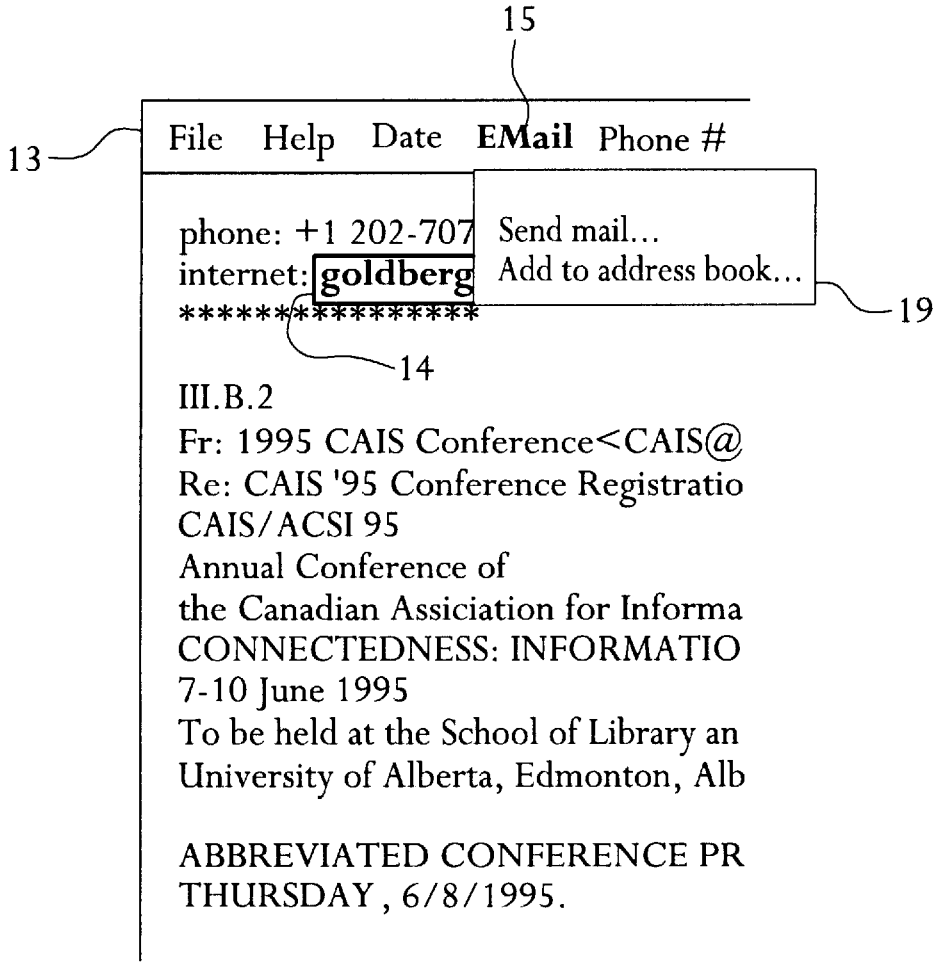

15
13
File Help Date EMail Phone #
phone: +1 202-707
internet: goldberg
**********
Send mail...
Add to address book...
19
14
III.B.2
Fr: 1995 CAIS Conference<CAIS@
Re: CAIS '95 Conference Registratio
CAIS/ACSI 95
Annual Conference of
the Canadian Assiciation for Informa
CONNECTEDNESS: INFORMATIO
7-10 June 1995
To be held at the School of Library an
University of Alberta, Edmonton, Alb
ABBREVIATED CONFERENCE PR
THURSDAY, 6/8/1995.

FIG. 1d

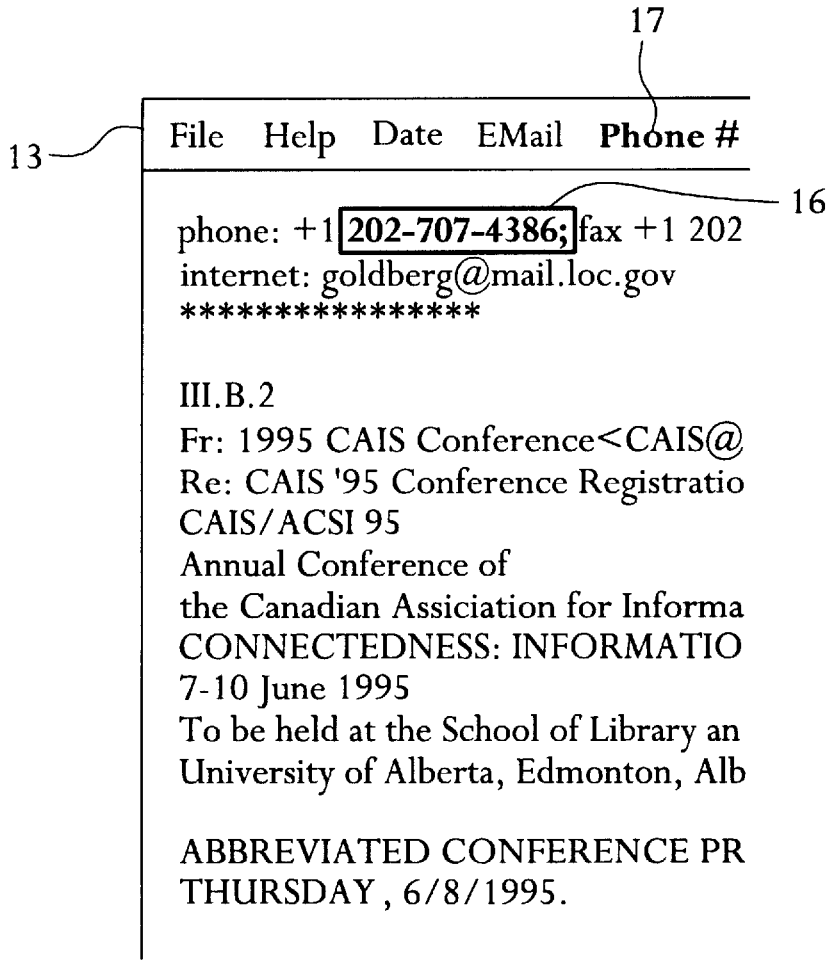

17
13
File Help Date EMail Phone #
16
phone: +1 202-707-4386; fax +1 202
internet: goldberg@mail.loc.gov
**********
III.B.2
Fr: 1995 CAIS Conference<CAIS@
Re: CAIS '95 Conference Registratio
CAIS/ACSI 95
Annual Conference of
the Canadian Assiciation for Informa
CONNECTEDNESS: INFORMATIO
7-10 June 1995
To be held at the School of Library an
University of Alberta, Edmonton, Alb
ABBREVIATED CONFERENCE PR
THURSDAY , 6/8/1995.

FIG. 1e

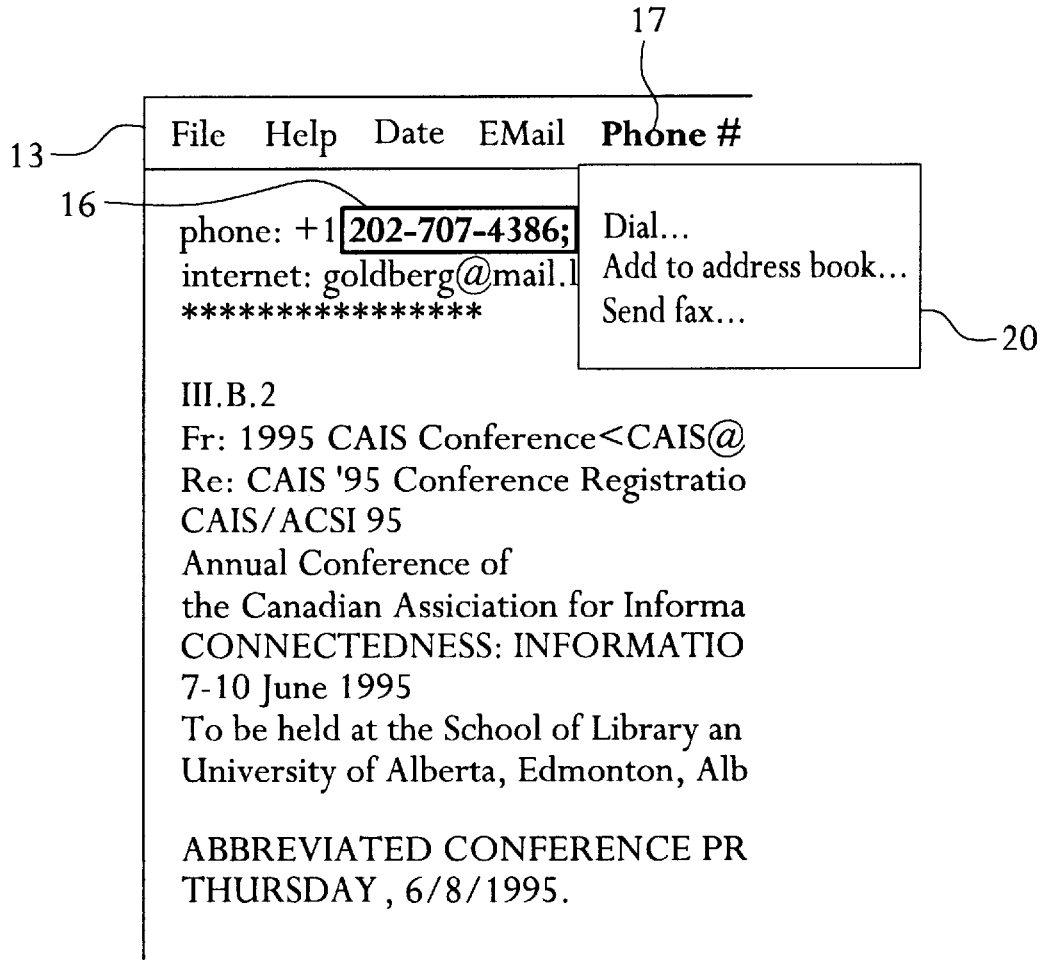

17
13
File Help Date EMail Phone #
16
phone: +1 202-707-4386;
internet: goldberg@mail.l
**********
Dial...
Add to address book...
Send fax...
20
III.B.2
Fr: 1995 CAIS Conference<CAIS@
Re: CAIS '95 Conference Registratio
CAIS/ACSI 95
Annual Conference of
the Canadian Assiciation for Informa
CONNECTEDNESS: INFORMATIO
7-10 June 1995
To be held at the School of Library an
University of Alberta, Edmonton, Alb
ABBREVIATED CONFERENCE PR
THURSDAY, 6/8/1995.

FIG. 1f

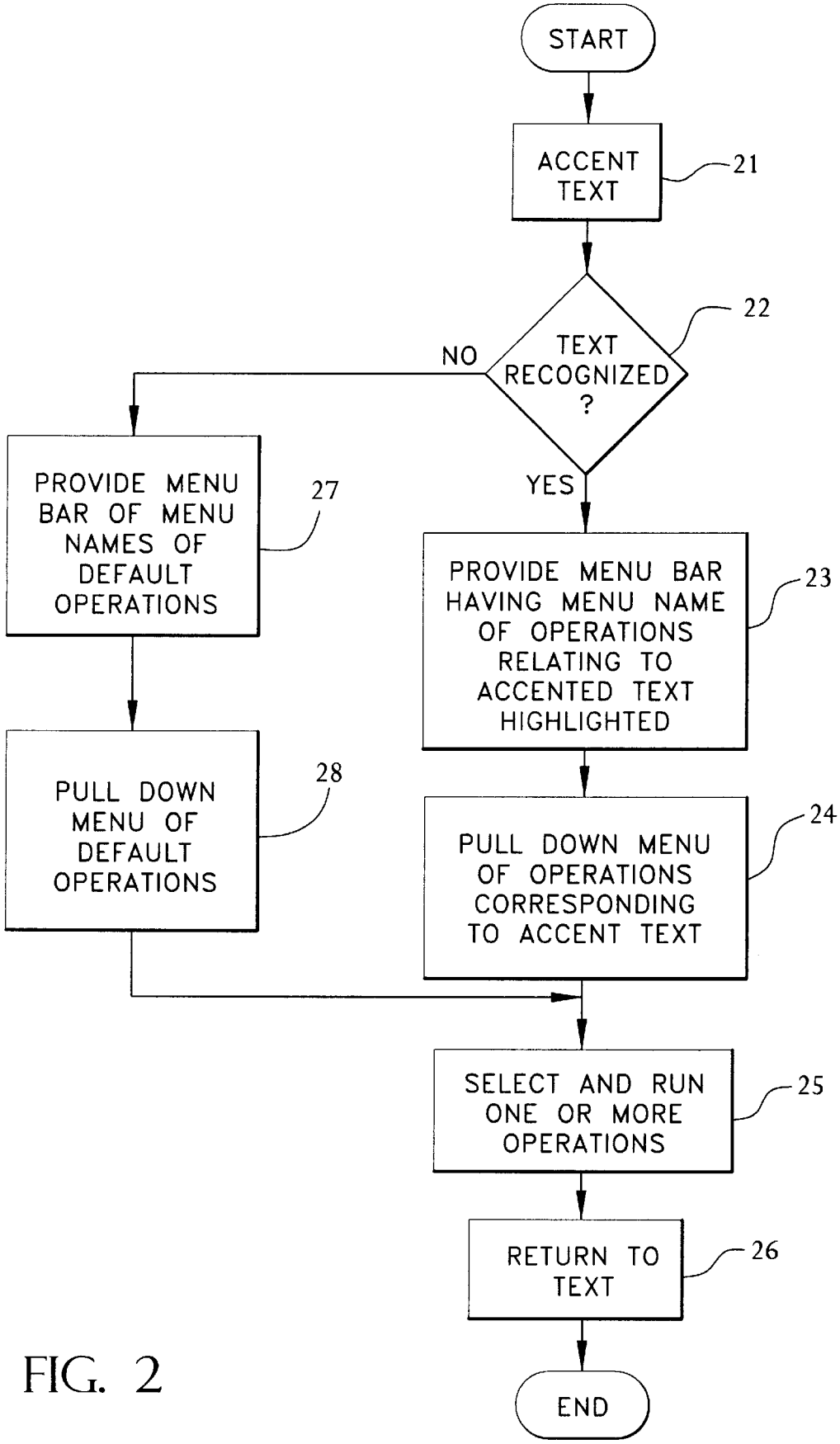

START
ACCENT TEXT
21
TEXT RECOGNIZED ?
22
NO
YES
PROVIDE MENU BAR OF MENU NAMES OF DEFAULT OPERATIONS
27
PROVIDE MENU BAR HAVING MENU NAME OF OPERATIONS RELATING TO ACCENTED TEXT HIGHLIGHTED
23
PULL DOWN MENU OF DEFAULT OPERATIONS
28
PULL DOWN MENU OF OPERATIONS CORRESPONDING TO ACCENT TEXT
24
SELECT AND RUN ONE OR MORE OPERATIONS
25
RETURN TO TEXT
26
END

FIG. 2

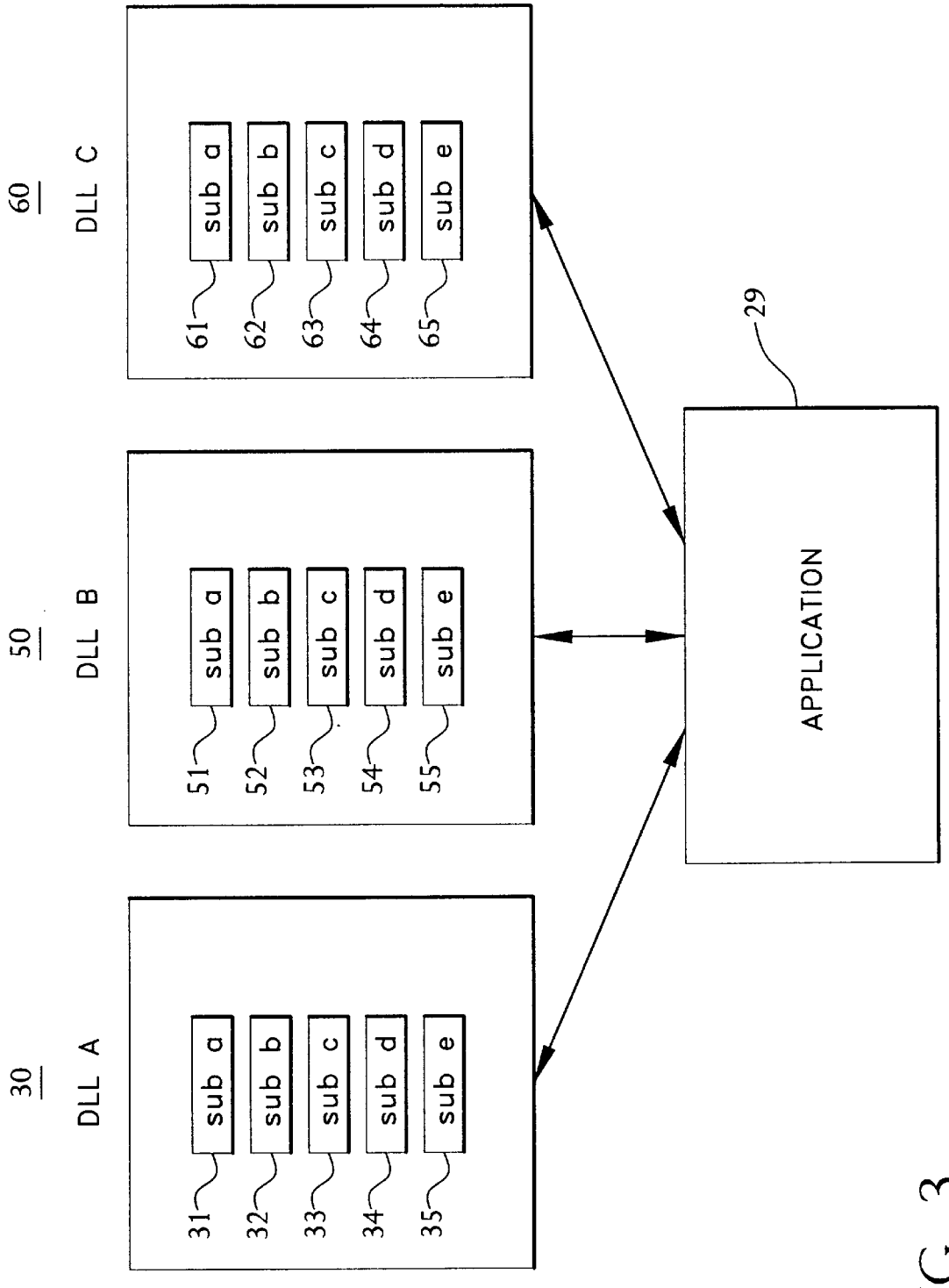

60
DLL C
sub a
sub b
sub c
sub d
sub e
61
62
63
64
65
50
DLL B
sub a
sub b
sub c
sub d
sub e
51
52
53
54
55
30
DLL A
sub a
sub b
sub c
sub d
sub e
31
32
33
34
35
29
APPLICATION

FIG. 3

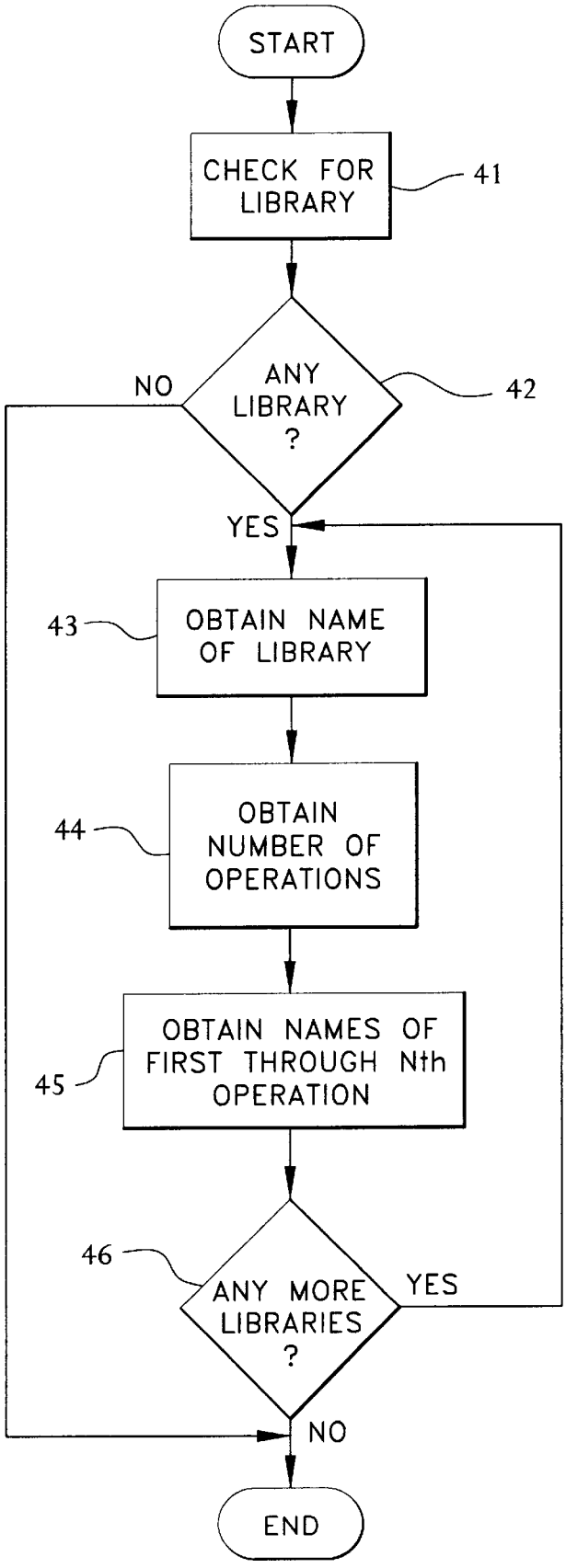

START
CHECK FOR LIBRARY
41
ANY LIBRARY ?
42
NO
YES
OBTAIN NAME OF LIBRARY
43
OBTAIN NUMBER OF OPERATIONS
44
OBTAIN NAMES OF FIRST THROUGH Nth OPERATION
45
ANY MORE LIBRARIES ?
46
YES
NO
END

FIG. 4

# RECOGNITION OF AND OPERATION ON TEXT DATA

## BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to the field of text data processing.

2. Description of the Related Art

Word recognition involves an ability to recognize selected words in a document or the like. One example of word recognition is the searching of large volumes of text, such as encyclopedias or legal case books, using key words or search terms. A user typically locates relevant segments of information from within large volumes by specifying a word or words which must appear in the segment in order for it to be retrieved. Generally, other limiters are used, such as commands which require that the searched for words appear in a same sentence or paragraph, or within a predetermined number of words from one another. Boolean connectors also figure prominently in this type of searching. Generally, therefore, this type of word recognition involves searching a large body of text for the presence of one or more words, which possibly are arranged in a predefined order. No operations are performed other than retrieval of a portion of text which includes the selected words and the words are not recognized as part of a general class.

Other word recognition and operation features are known. Generally, programs which provide for word recognition and for operations using or on the recognized words require the words to have been created by the program. For example, conventional applications that allow users to perform word recognition and operations on the words typically require that the words be created using the application. Words created using any other application are not recognizable and may not be operated on. Consequently, words created using a particular word processing or database program can be recognized and operated on only by the word processing or database program responsible for their creation. Examples of the conventional operations which can be performed on words by conventional word processing or database programs include spell checking, finding and replacing, etc.

The present invention will benefit any application which displays text to a user, regardless of the origin of the text. The invention expands the operations which may be performed using recognized text by allowing a user to intuitively exploit the presence of certain classes or types of text in any document by transforming the text into an interface to other functions or operations.

## SUMMARY OF THE INVENTION

The invention pertains to recognition of text in a body of text as belonging to a predetermined class and performing an operation relevant to the recognized text.

## BRIEF DESCRIPTION OF THE DRAWINGS

The following detailed description will be more fully understood with reference to the accompanying drawings in which:

FIGS. **1***a*–**1***f* are graphic representations of recognized text on video monitors in accordance with the invention;

FIG. **2** is a flow chart diagram of the operation of the test recognition and operation features of the invention;

FIG. **3** is a block diagram of the libraries of the invention; and

FIG. **4** is a flow chart diagram of the menu building features of the invention.



## DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The invention selectively recognizes text and performs relevant operations based on the recognition. Referring to FIG. **1***a* and FIG. **2**, for example, a date **11** in text appearing on a video monitor is accented (step **21** of FIG. **2**) for example by shading, underlining or pointing to and clicking on the text. The invention recognizes the accented text (step **22**), and provides a menu bar **13** in which the name of menu **12** corresponding to the class of text accented is highlighted or shown in bold type, thereby showing that the menu is enabled (step **23**). In the example of FIG. **1***a*, the Date menu **12** is shown in bold type, signifying that the invention includes a menu of operations and/or programs which are relevant to dates. A user can "click" on the Date menu name **12** or otherwise call the menu by one or more keystrokes on a keyboard associated with the video monitor to display or pull down the contents of the menu (step **24**). A view of an embodiment of a pulled-down Date menu **18** is shown in FIG. **1***b*. A user may directly call a calender or appointment database program from pulled-down menu **18**. Other programs may be included in pulled-down date menu **18** as discussed below.

As shown in FIGS. **1***c*–**1***f*, the invention is not limited to the recognition of dates in text and preferred embodiments of the invention can recognize e-mail addresses and telephone numbers. In fact, there is no limit on the type of text which can be recognized by the invention and additional embodiments can recognize such classes of text as Uniform Resource Locators, nouns, verbs, names, street addresses, etc.

The pull-down menus provided by the invention identify the operations and/or programs which relate to the class of text accented, highlighted or otherwise indicated. For example, referring again to FIG. **1***a* where date **11** has been accented and recognized by the invention, the pulled-down menu **18** can identify operations and/or programs relevant to dates, such as the calendar program and appointment programs shown as well as a To-Do list program, an anniversary database, a scheduling program etc. . . . A user is able to run one or more of the programs relevant to dates which are identified in the pulled-down menu in a known manner, such as by clicking on the name of the program as it appears in the pulled-down menu (step **25**) or through the execution of one or more keyboard key strokes. In the example shown, therefore, a user is able to record in, for example, a calendar program, an upcoming event mentioned in a body of text in which a date has been recognized. The user may then quickly return to the body of text (step **26**).

Referring to FIG. **1***c*, an e-mail address **14** is accented. In this example, a user may click on the highlighted menu name EMail **15** to pull-down the menu. The EMail menu preferably includes, for example, an identification of programs and operations related to EMail and EMail addresses.

An embodiment of pulled-down EMail menu **19** is shown in FIG. **1***d*. Included in pulled-down Email menu **19** are such programs as a writable Email or general address book database and an EMail template and transmitting program, preferably automatically addressed with the accented address recognized in the text, etc. Any other program related to EMail sending or address storage may be included as within the scope of this invention.

Referring now to FIG. **1***e*, a telephone number **16** is accented. The pull down menu named Phone #**17** is highlighted and preferably identifies the executable operations and/or programs which are relevant to telephone and telefax

numbers. As shown in FIG. **1***f* on pulled-down menu **20**, possible programs include a writable computer database of telephone and telefax numbers, a program which instructs a properly equipped computer to dial the number accented, a program which generates a template for the preparation of a fax message and which subsequently causes a properly equipped computer to transmit the message to the accented number, etc. Again, any program related to telephone or telefax numbers can be included in pulled-down menu **20** for direct accessing in accordance with the teachings of this disclosure.

Where the invention is capable of recognizing nouns or verbs, pull-down menus can, for example, identify executable programs which provide the meaning of the highlighted word, appropriate synonyms and the singular or plural version of the noun or conjugation of the verb.

As noted above the invention preferably includes a library enabling recognition of Uniform Resource Locators (URLs) in text. Consequently, preferred programs which appear on and can be run from the pull-down menu in response to the accenting and subsequent recognition of a URL include World-Wide Web browser programs, such as "NETSCAPE" or "NCSA MOSAIC."

In a preferred embodiment, in the event the accented text is not recognized, i.e., the text is not of the specific type or class recognizable by any of the libraries provided, a menu bar having a list of one or more menu names of default operations can be made to appear (step **27**). The invention preferably includes as default operations such programs as spell-checkers, grammar-checkers, a thesaurus, a dictionary, execution of an EMail program to transmit the text, programs to store the text and any other programs relating to words in general. Of course, the names of the default programs appear on one or more pull-down menus (step **28**) corresponding to the one or more menu names.

Referring now to FIG. **3**, the invention is implemented in one or more modular libraries of subroutines. The libraries can be Dynamically Linked Libraries as understood by those skilled in the art with "MICROSOFT" operating systems. Every subroutine performs a distinct task. In FIG. **3**, three libraries **30**, **50** and **60** are shown, although the number of libraries which are possible is limited only by the number of possible operations which may be performed. Each library preferably holds five sets of subroutines. For example, Libraries A, B and C each include subroutines a, b, c, d and e. Subroutines a and c are concerned with the class or type of text data recognized by the Library. Subroutines b, d and e are concerned with the operations performed by the Library on the recognized text.

Library A concerns, for example, recognizing and performing operations on dates in text as described above in reference to FIGS. **1***a* and **1***b*. Subroutine a (**31**) of Library A preferably detects a class or type of text data, in this case, dates. Generally, subroutine a is a parser capable of recognizing a class or type of text data in a number of formats. In the case of Library A, the parser subroutine a is able to recognize, for example, date text written in a number of variations, for example 1/1/99, 1/1/1999, 1-1-99, January 1, 1999 etc.

Subroutine c (**33**) of Library A provides the menu name corresponding to Library A and its text recognition and operating functions. For example, subroutine c provides the name Date or Dates **12**. This name appears in menu bar **13**, as shown in FIG. **1***a* and can be displayed in bold-face type or emphasized in some other way whenever date text has been recognized to signify that a number of operations are available for running on the date text.

Subroutine d (**34**) of Library A identifies the particular number of operations which can be performed on the date text and correlates to the number of operations implemented by subroutine b. Each operation is identified by a number between and including 1 and the value returned by subroutine d.

Given a number identifying an operation, subroutine e (**35**) of Library A identifies the name of the operation. Examples of the names of the operations which can be run on date text include Schedule, To-Do List, Anniversary, etc. Subroutine e provides the names of the operations as they appear in pull-down menu **18**.

Given a number identifying an operation, subroutine b (**32**) of Library A performs the identified operation on the recognized text data. For example, subroutine b can call scheduling programs, writable calendar databases, writable to-do list databases, anniversary book databases and any other number of programs or operations relevant to dates.

A person of ordinary skill will understand that any additional libraries, such as Libraries B and C shown in FIG. **3** will have subroutines generally related in function to the subroutines of Library A for implementing the invention with respect to other classes of text. For example, the subroutines of Library B preferably are directed to implementing the invention with respect to EMail addresses in a document and the subroutines of Library C are directed to implementing the invention with respect to telephone and telefax numbers, as shown in FIGS. **1***b*–**1***f*. Other libraries may be added to, for example, operate on URLs, nouns, verbs, names street addresses, etc.

Conventionally, a software program must be entirely recompiled for its functionality to be increased or its operations changed in any manner. The present invention, however, is highly modular and allows libraries to be added at will and additional features to be added to libraries without recompiling. For example, by implementing libraries as "MICROSOFT" Component Object Model Servers or by using equivalent standards known to those skilled in the art, each library is recognized and utilized using the same programming interface, i.e., though the function and results of the subroutines a–e differ from library to library, every library has subroutines a–e. Consequently, an application which is written to recognize and utilize one library can automatically recognize and utilize any other library. Furthermore, an application which is written to recognize and utilize more than one library can automatically recognize and utilize any number of other libraries. Additional libraries can be added at any time, without recompiling, and an application using standards known to those skilled in the art will recognize the addition of one or more libraries at run-time. For example, an application **29** using the invention will use standards known to those skilled in the art to check for the presence of any libraries. Once recognized, the application **29** will be able to use any or all of the libraries.

The invention also allows additional features to be added to a library without recompiling. For example, additional operations can be installed in a library at any time, and recognized by application **29** at run-time. For example, at run-time, application **29** will build the appropriate menus relating to the one or more types of data which are recognizable. At run-time, application **29** will identify, for example, the presence of Library A (steps **41** and **42** of FIG. **4**). Subroutine c of Library A informs application **29** of the name of the Library (step **43**), which of course will be used as one of the names of the menu of operations of appearing in the menu bar. Application **29** subsequently queries sub-

routine d of Library A for the number of operations Library A is capable of performing on the recognized data (step **44**). Subroutine e provides to application **29** the names for each of the number of operations identified, e.g. Schedule, Calendar, To-Do, Anniversary, etc. (step **45**). Application **29** will repeat the process outlined above to build menus relating to every library which is part of the invention (Step **46**). This menu-building procedure at run time allows libraries to be added and to be upgraded at any time, for example to add additional operations performable on a piece of recognized text, without a need for recompiling the program.

Conventional programs that provide any text recognition and operation capabilities, i.e., find and replace, etc. require the text to be embedded in, for example, a document created by the program. This invention does not require that the text be embedded in any document created on or by a particular application program. Any text appearing on a video monitor can be operated on by the invention, whether the text is within an EMail message, World-Wide Web site, created by a word processing or database program, etc. Furthermore, by using parsers as the subroutines for detecting certain types of data, the invention is able to recognize data appearing in a number of formats, rather than a single defined format.

The present invention can be embodied in the form of computer-implemented processes and apparatuses for practicing those processes. The present invention also can be embodied in the form of computer program code embodied in tangible media, such as floppy diskettes, CD-ROMS, hard drives, or any other computer-readable storage medium, wherein, when the computer program code is loaded into and executed by a computer, the computer becomes an apparatus for practicing the invention. The present invention can also be embodied in the form of computer program code, for example, whether stored in a storage medium, loaded into and/or executed by a computer, or transmitted over some transmission medium, such as over electrical wiring or cabling, through fiber optics, or via electromagnetic radiation, wherein, when the computer program code is loaded into and executed by a computer, the computer becomes an apparatus for practicing the invention.

When implemented on a general-purpose microprocessor, the computer program code segments configure the microprocessor to create specific logic circuits.

Furthermore, it should be understood that various changes in the details, materials, and arrangements of the parts which have been described and illustrated in order to explain the nature of this invention may be made by those skilled in the art without departing from the principle and scope of the invention as expressed in the following claims.

What I claim is:

**1**. A computer-implemented method for processing a selected text, comprising the steps of:

(a) recognizing the selected text as belonging to a predetermined class, a computer performing the step of recognizing, including:

(1) comparing a format of the selected text to a predetermined format associated with the predetermined class,

(3) recognizing the selected text as belonging to the predetermined class if the format of the selected text matches the predetermined format; and

(b) performing an operation associated with the predetermined class using the recognized text as a parameter.

**2**. The method of claim **1**, wherein step (a) comprises the step of:

(i) displaying a plurality of menu names, and

(ii) emphasizing, relative to any other displayed menu name, a menu name of a menu identifying one or more operations associated with the predetermined class, which may be performed on the recognized text.

**3**. The method of claim **2**, wherein step (b) comprises the steps of:

(i) displaying the menu; and

(ii) starting one or more operations from the one or more operations identified on the menu.

**4**. The method of claim **2**, wherein:

step (i) comprises the steps of:

(1) checking for a library of the one or more operations; and

(2) obtaining the name of the library; and

step (ii) comprises the steps of:

(1) obtaining a number of the one or more operations in the library; and

(2) obtaining names for each of the number of operations.

**5**. The method of claim **4**, further comprising the step of:

(3) repeating steps (i)1, (i)2, (ii)1 and (ii)2 for each library present.

**6**. The method of claim **1**, wherein the text appears in a body of text, further comprising the step of:

(c) returning to the body of text.

**7**. The method of claim **1**, wherein the text is one of a date, name, telephone number, telefax number, e-mail address, and Uniform Resource Locator.

**8**. The method of claim **7**, wherein the operation is the starting of a writable, computer database application.

**9**. The method of claim **7**, wherein the text is a Uniform Resource Locator and the operation is the starting of a World-Wide Web browser application.

**10**. The method of claim **7**, wherein the text is an EMail address and the operation is the starting of an EMail message preparing and transmitting application.

**11**. The method of claim **7**, wherein the text is a telephone number and the operation is the starting of an application causing a modem to dial the telephone number.

**12**. The method of claim **7**, wherein the text is a telefax number and the operation is the starting of telefax message preparing and transmitting application.

**13**. A method according to claim **1**, wherein step (a) includes:

(1) comparing the format of the selected text to a plurality of predetermined formats associated with the predetermined class, and

(2) recognizing the selected text as belonging to the predetermined class if the format of the selected text matches any one of the plurality of predetermined formats.

**14**. A method according to claim **1**, wherein step (a) is performed a plurality of times, for a plurality of predetermined classes, respectively, and the selected text is recognized as belonging to one of the plurality of predetermined classes, further comprising the step of:

displaying a plurality of menu names, each associated with a respectively different one of the plurality of predetermined classes;

emphasizing the respective menu name associated with the one of the plurality of predetermined classes to which the selected text belongs, relative to menu names

that are not associated with the plurality of predetermined classes.

**15**. An apparatus for text processing, comprising:

(a) means for recognizing a selected text as belonging to a predetermined class, including:

(1) means for comparing a format of the selected text to a predetermined format associated with the predetermined class,

(2) means for recognizing the selected text as belonging to the predetermined class if the format of the selected text matches the predetermined format; and

(b) means for performing an operation associated with the predetermined class using the recognized selected text as a parameter.

**16**. The apparatus of claim **15**, wherein means (a):

(i) displays a plurality of menu names, and

(ii) emphasizes, relative to any other displayed menu name, a menu name of a menu identifying one or more operations associated with the predetermined class, which may be performed on the recognized text.

**17**. The apparatus of claim **16**, wherein means (a):

generates a menu name of a menu of one or more operations by:

(1) checking for a library of the one or more operations; and

(2) obtaining the name of the library; and

generates a menu for listing the one or more operations by:

(1) obtaining a number of the one or more operations in the library; and

(2) obtaining names for each of the number of operations.

**18**. The apparatus of claim **16**, wherein means (b):

(i) displays the menu; and

(ii) starts one or more operations to be run from the one or more operations identified on the menu.

**19**. The apparatus of claim **15**, wherein the text appears in a body of text, further comprising:

(c) means for returning to the body of text.

**20**. The apparatus of claim **15**, wherein the text is one of a date, name, telephone number, telefax number, e-mail address and Uniform Resource Locator.

**21**. The apparatus of claim **20**, wherein the operation is the starting of a writable, computer database application.

**22**. The apparatus of claim **20**, wherein the text is a Uniform Resource Locator and the operation is the starting of a World-Wide Web browser application.

**23**. The apparatus of claim **20**, wherein the text is an EMail address and the operation is the starting of an EMail message preparing and transmitting application.

**24**. The apparatus of claim **20**, wherein the text is a telephone number and the operation is the starting of an application causing a modem to dial the telephone number.

**25**. The apparatus of claim **20**, wherein the text is a telefax number and the operation is the starting of telefax message preparing and transmitting application.

**26**. Apparatus according to claim **15**, wherein the recognizing means includes:

(1) means for comparing the format of the selected text to a plurality of predetermined formats associated with the predetermined class, and

(2) means for recognizing the selected text as belonging to the predetermined class if the format of the selected text matches any one of the plurality of predetermined formats.

**27**. Apparatus according to claim **15**, wherein the recognizing means performs recognition a plurality of times, for a plurality of predetermined classes, respectively, and recognizes the selected text as belonging to one of the plurality of predetermined classes, further comprising:

means for displaying a plurality of menu names, each associated with a respectively different one of the plurality of predetermined classes;

means for emphasizing the respective menu name associated with the one of the plurality of predetermined classes to which the selected text belongs, relative to menu names that are not associated with the plurality of predetermined classes.

**28**. A storage medium encoded with machine-readable computer program code for text processing, comprising:

(a) means for causing a computer to recognize a selected text as belonging to a predetermined class, including:

(1) means for comparing a format of the selected text to a predetermined format associated with the predetermined class,

(2) means for recognizing the selected text as belonging to the predetermined class if the format of the selected text matches the predetermined format; and

(b) means for causing the computer to perform an operation associated with the predetermined class using the recognized selected text as a parameter.

**29**. The storage medium of claim **28**, wherein means (a) causes the computer to:

(i) display a plurality of menu names, and

(ii) emphasize, relative to any other displayed menu name, a menu name of a menu identifying one or more operations associated with the predetermined class, which may be performed on the recognized text.

**30**. The storage medium of claim **29**, wherein means (b) causes the computer to:

(i) display the menu; and

(ii) start one or more operations to be run from the one or more operations identified on the menu.

**31**. The storage medium of claim **28**, wherein the text appears in a body of text, further comprising:

(c) means for causing the computer to return to the body of text.

**32**. The storage medium of claim **28**, wherein the text is one of a date, name, telephone number, telefax number, e-mail address, street address and Uniform Resource Locator.

**33**. The storage medium of claim **32**, wherein the operation is the starting of a writable, computer database application.

**34**. The storage medium of claim **32**, wherein the text is a Uniform Resource Locator and the operation is the starting of a World-Wide Web browser application.

**35**. The storage medium of claim **32**, wherein the text is an EMail address and the operation is the starting of an EMail message preparing and transmitting application.

**36**. The storage medium of claim **32**, wherein the text is a telephone number and the operation is the starting of an application causing a modem to dial the telephone number.

**37**. The storage medium of claim **32**, wherein the text is a telefax number and the operation is the starting of telefax message preparing and transmitting application.

**38**. The storage medium of claim **32**, wherein:

means (a) causes the computer to:

(1) check for a library of the one or more operations; and

(2) obtain the name of the library, to generate a menu name of a menu of the one or more operations; and

(1) obtain a number of the one or more operations in the library; and

(2) obtain names for each of the number of operations, to generate a menu for listing the one or more operations.

**39**. A storage medium according to claim **28**, wherein means (a) includes:

(1) means for causing the computer to compare the format of the selected text to a plurality of predetermined formats associated with the predetermined class, and

(2) means for causing the computer to recognize the selected text as belonging to the predetermined class if the format of the selected text matches any one of the plurality of predetermined formats.

**40**. A storage medium according to claim **28**, wherein means (a) causes the computer to perform recognition a plurality of times, for a plurality of predetermined classes, respectively, and causes the computer to recognize the selected text as belonging to one of the plurality of predetermined classes, further comprising:

means for causes the computer to display a plurality of menu names, each associated with a respectively different one of the plurality of predetermined classes;

means for causes the computer to emphasize the respective menu name associated with the one of the plurality of predetermined classes to which the selected text belongs, relative to menu names that are not associated with the plurality of predetermined classes.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

# CERTIFICATE OF CORRECTION

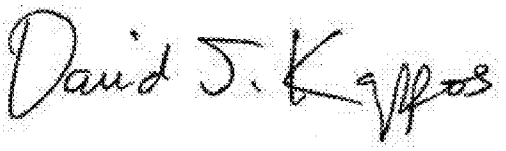


| | |
|---|---|
| PATENT NO. | : 5,859,636 |
| APPLICATION NO. | : 08/579568 |
| DATED | : January 12, 1999 |
| INVENTOR(S) | : Pandit |

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In column 5, at line 63, delete, "(3)" and insert --(2)--.

Signed and Sealed this
Twenty-sixth Day of July, 2011



David J. Kappos
*Director of the United States Patent and Trademark Office*

# Exhibit 28

PTO/SB/26 (07-09)
Approved for use through 07/31/2012. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

| TERMINAL DISCLAIMER TO OBVIATE A DOUBLE PATENTING REJECTION OVER A "PRIOR" PATENT | Docket Number (Optional) 3324/102 |
|---|---|

In re Application of: Atle Hedloy

Application No.: 12/182,048

Filed: July 29, 2008

For: Method, System and Computer Readable Medium for Addressing Handling from a Computer Program

The owner*, Arendi S.A.R.L., of 100 percent interest in the instant application hereby disclaims, except as provided below, the terminal part of the statutory term of any patent granted on the instant application which would extend beyond the expiration date of the full statutory term **prior patent** No. 6,323,853 as the term of said prior patent is defined in 35 U.S.C. 154 and 173, and as the term of said **prior patent** is presently shortened by any terminal disclaimer. The owner hereby agrees that any patent so granted on the instant application shall be enforceable only for and during such period that it and the **prior patent** are commonly owned. This agreement runs with any patent granted on the instant application and is binding upon the grantee, its successors or assigns.

In making the above disclaimer, the owner does not disclaim the terminal part of the term of any patent granted on the instant application that would extend to the expiration date of the full statutory term as defined in 35 U.S.C. 154 and 173 of the **prior patent**, "as the term of said **prior patent** is presently shortened by any terminal disclaimer," in the event that said **prior patent** later:

expires for failure to pay a maintenance fee;
is held unenforceable;
is found invalid by a court of competent jurisdiction;
is statutorily disclaimed in whole or terminally disclaimed under 37 CFR 1.321;
has all claims canceled by a reexamination certificate;
is reissued; or
is in any manner terminated prior to the expiration of its full statutory term as presently shortened by any terminal disclaimer.

Check either box 1 or 2 below, if appropriate.

1. ☐ For submissions on behalf of a business/organization (e.g., corporation, partnership, university, government agency, etc.), the undersigned is empowered to act on behalf of the business/organization.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

2. ☒ The undersigned is an attorney or agent of record. Reg. No. 61,033

/Jakub M. Michna, #61,033/ — Signature
December 8, 2010 — Date

Jakub M. Michna — Typed or printed name

(617) 443-9292 — Telephone Number

☒ Terminal disclaimer fee under 37 CFR 1.20(d) included.

**WARNING: Information on this form may become public. Credit card information should not be included on this form. Provide credit card information and authorization on PTO-2038.**

*Statement under 37 CFR 3.73(b) is required if terminal disclaimer is signed by the assignee (owner).
Form PTO/SB/96 may be used for making this certification. See MPEP § 324.

# Exhibit 29

# THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

# Exhibit 30

# THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

# Exhibit 31

# Intentionally Left Blank

# Exhibit 32

# THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

# Exhibit 33

# TextView

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

Kotlin (/reference/kotlin/android/widget/TextView) | **Java**

```
public class TextView
extends View (/reference/android/view/View) implements ViewTreeObserver.OnPreDrawListener (/reference/android/view/ViewTreeObserver.OnPreDrawListener)
```

java.lang.Object (/reference/java/lang/Object)
↳ android.view.View (/reference/android/view/View)
↳ android.widget.TextView

Known direct subclasses

Button (/reference/android/widget/Button), CheckedTextView (/reference/android/widget/CheckedTextView), Chronometer (/reference/android/widget/Chronometer), DigitalClock (/reference/android/widget/DigitalClock), EditText (/reference/android/widget/EditText), TextClock (/reference/android/widget/TextClock)

| | |
|---|---|
| Button (/reference/android/widget/Button) | A user interface element the user can tap or click to perform an action. |
| CheckedTextView (/reference/android/widget/CheckedTextView) | An extension to **`TextView`** (/reference/android/widget/TextView) that supports the **`Checkable`** (/reference/android/widget/Checkable) interface and displays. |
| Chronometer (/reference/android/widget/Chronometer) | Class that implements a simple timer. |
| DigitalClock (/reference/android/widget/DigitalClock) | *This class was deprecated in API level 17. It is recommended you use* **`TextClock`** (/reference/android/widget/TextClock) *instead.* |
| EditText (/reference/android/widget/EditText) | A user interface element for entering and modifying text. |
| TextClock (/reference/android/widget/TextClock) | **`TextClock`** can display the current date and/or time as a formatted string. |

Known indirect subclasses

AutoCompleteTextView (/reference/android/widget/AutoCompleteTextView), CheckBox (/reference/android/widget/CheckBox), CompoundButton (/reference/android/widget/CompoundButton), ExtractEditText (/reference/android/inputmethodservice/ExtractEditText), MultiAutoCompleteTextView (/reference/android/widget/MultiAutoCompleteTextView), RadioButton (/reference/android/widget/RadioButton), Switch (/reference/android/widget/Switch), ToggleButton (/reference/android/widget/ToggleButton)

| | |
|---|---|
| AutoCompleteTextView (/reference/android/widget/AutoCompleteTextView) | An editable text view that shows completion suggestions automatically while the user is typing. |
| CheckBox (/reference/android/widget/CheckBox) | A checkbox is a specific type of two-states button that can be either checked or unchecked. |
| CompoundButton (/reference/android/widget/CompoundButton) | A button with two states, checked and unchecked. |
| ExtractEditText (/reference/android/inputmethodservice/ExtractEditText) | Specialization of **`EditText`** (/reference/android/widget/EditText) for showing and interacting with the extracted text in a full-screen input method. |
| MultiAutoCompleteTextView (/reference/android/widget/MultiAutoCompleteTextView) | An editable text view, extending **`AutoCompleteTextView`** (/reference/android/widget/AutoCompleteTextView), that can show completion suggestions for the substring of the text where the user is typing instead of necessarily for the entire thing. |
| RadioButton (/reference/android/widget/RadioButton) | A radio button is a two-states button that can be either checked or unchecked. |
| Switch (/reference/android/widget/Switch) | A Switch is a two-state toggle switch widget that can select between two options. |
| ToggleButton (/reference/android/widget/ToggleButton) | Displays checked/unchecked states as a button with a "light" indicator and by default accompanied with the text "ON" or "OFF". |

A user interface element that displays text to the user. To provide user-editable text, see **`EditText`** (/reference/android/widget/EditText).

The following code sample shows a typical use, with an XML layout and code to modify the contents of the text view:

```
<LinearLayout
     xmlns:android="http://schemas.android.com/apk/res/android"
     android:layout_width="match_parent"
     android:layout_height="match_parent">
   <TextView
       android:id="@+id/text_view_id"
       android:layout_height="wrap_content"
       android:layout_width="wrap_content"
       android:text="@string/hello" />
</LinearLayout>
```

This code sample demonstrates how to modify the contents of the text view defined in the previous XML layout:

```
public class MainActivity extends Activity {

   protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        final TextView helloTextView = (TextView) findViewById(R.id.text_view_id);
        helloTextView.setText(R.string.user_greeting);
    }
}
```

To customize the appearance of TextView, see Styles and Themes (https://developer.android.com/guide/topics/ui/themes.html).

### XML attributes

See `TextView Attributes` (/reference/android/R.styleable#TextView), `View Attributes` (/reference/android/R.styleable#View)

## Summary

### Nested classes

| | |
|---|---|
| `enum` | `TextView.BufferType` (/reference/android/widget/TextView.BufferType)<br>Type of the text buffer that defines the characteristics of the text such as static, styleable, or editable. |
| `interface` | `TextView.OnEditorActionListener` (/reference/android/widget/TextView.OnEditorActionListener)<br>Interface definition for a callback to be invoked when an action is performed on the editor. |
| `class` | `TextView.SavedState` (/reference/android/widget/TextView.SavedState)<br>User interface state that is stored by TextView for implementing `View#onSaveInstanceState` (/reference/android/view/View#onSaveInstanceState()). |

### XML attributes

| | |
|---|---|
| `android:allowUndo` (/reference/android/widget/TextView#attr_android:allowUndo) | Whether undo should be allowed for editable text. |
| `android:autoLink` (/reference/android/widget/TextView#attr_android:autoLink) | Controls whether links such as urls and email addresses are automatically found and converted to clickable links. |
| `android:autoSizeMaxTextSize` (/reference/android/widget/TextView#attr_android:autoSizeMaxTextSize) | The maximum text size constraint to be used when auto-sizing text. |

| | |
|---|---|
| **android:autoSizeMinTextSize** (/reference/android/widget/TextView#attr_android:autoSizeMinTextSize) | The minimum text size constraint to be used when auto-sizing text. |
| **android:autoSizePresetSizes** (/reference/android/widget/TextView#attr_android:autoSizePresetSizes) | Resource array of dimensions to be used in conjunction with **autoSizeTextType** set to **uniform**. |
| **android:autoSizeStepGranularity** (/reference/android/widget/TextView#attr_android:autoSizeStepGranularity) | Specify the auto-size step size if **autoSizeTextType** is set to **uniform**. |
| **android:autoSizeTextType** (/reference/android/widget/TextView#attr_android:autoSizeTextType) | Specify the type of auto-size. |
| **android:autoText** (/reference/android/widget/TextView#attr_android:autoText) | If set, specifies that this TextView has a textual input method and automatically corrects some common spelling errors. |
| **android:breakStrategy** (/reference/android/widget/TextView#attr_android:breakStrategy) | Break strategy (control over paragraph layout). |
| **android:bufferType** (/reference/android/widget/TextView#attr_android:bufferType) | Determines the minimum type that getText() will return. |
| **android:capitalize** (/reference/android/widget/TextView#attr_android:capitalize) | If set, specifies that this TextView has a textual input method and should automatically capitalize what the user types. |
| **android:cursorVisible** (/reference/android/widget/TextView#attr_android:cursorVisible) | Makes the cursor visible (the default) or invisible. |
| **android:digits** (/reference/android/widget/TextView#attr_android:digits) | If set, specifies that this TextView has a numeric input method and that these specific characters are the ones that it will accept. |
| **android:drawableBottom** (/reference/android/widget/TextView#attr_android:drawableBottom) | The drawable to be drawn below the text. |
| **android:drawableEnd** (/reference/android/widget/TextView#attr_android:drawableEnd) | The drawable to be drawn to the end of the text. |
| **android:drawableLeft** (/reference/android/widget/TextView#attr_android:drawableLeft) | The drawable to be drawn to the left of the text. |
| **android:drawablePadding** (/reference/android/widget/TextView#attr_android:drawablePadding) | The padding between the drawables and the text. |
| **android:drawableRight** (/reference/android/widget/TextView#attr_android:drawableRight) | The drawable to be drawn to the right of the text. |

| | |
|---|---|
| `android:drawableStart` (/reference/android/widget/TextView#attr_android:drawableStart) | The drawable to be drawn to the start of the text. |
| `android:drawableTint` (/reference/android/widget/TextView#attr_android:drawableTint) | Tint to apply to the compound (left, top, etc.) drawables. |
| `android:drawableTintMode` (/reference/android/widget/TextView#attr_android:drawableTintMode) | Blending mode used to apply the compound (left, top, etc.) drawables tint. |
| `android:drawableTop` (/reference/android/widget/TextView#attr_android:drawableTop) | The drawable to be drawn above the text. |
| `android:editable` (/reference/android/widget/TextView#attr_android:editable) | If set, specifies that this TextView has an input method. |
| `android:editorExtras` (/reference/android/widget/TextView#attr_android:editorExtras) | Reference to an `<input-extras>` (/reference/android/R.styleable#InputExtras) XML resource containing additional data to supply to an input method, which is private to the implementation of the input method. |
| `android:elegantTextHeight` (/reference/android/widget/TextView#attr_android:elegantTextHeight) | Elegant text height, especially for less compacted complex script text. |
| `android:ellipsize` (/reference/android/widget/TextView#attr_android:ellipsize) | If set, causes words that are longer than the view is wide to be ellipsized instead of broken in the middle. |
| `android:ems` (/reference/android/widget/TextView#attr_android:ems) | Makes the TextView be exactly this many ems wide. |
| `android:enabled` (/reference/android/widget/TextView#attr_android:enabled) | Specifies whether the widget is enabled. |
| `android:fallbackLineSpacing` (/reference/android/widget/TextView#attr_android:fallbackLineSpacing) | Whether to respect the ascent and descent of the fallback fonts that are used in displaying the text. |
| `android:firstBaselineToTopHeight` (/reference/android/widget/TextView#attr_android:firstBaselineToTopHeight) | Distance from the top of the TextView to the first text baseline. |
| `android:fontFamily` (/reference/android/widget/TextView#attr_android:fontFamily) | Font family (named by string or as a font resource reference) for the text. |

| | |
|---|---|
| **android:fontFeatureSettings** (/reference/android/widget/TextView#attr_android:fontFeatureSettings) | Font feature settings. |
| **android:fontVariationSettings** (/reference/android/widget/TextView#attr_android:fontVariationSettings) | Font variation settings. |
| **android:freezesText** (/reference/android/widget/TextView#attr_android:freezesText) | If set, the text view will include its current complete text inside of its frozen icicle in addition to meta-data such as the current cursor position. |
| **android:gravity** (/reference/android/widget/TextView#attr_android:gravity) | Specifies how to align the text by the view's x- and/or y-axis when the text is smaller than the view. |
| **android:height** (/reference/android/widget/TextView#attr_android:height) | Makes the TextView be exactly this tall. |
| **android:hint** (/reference/android/widget/TextView#attr_android:hint) | Hint text to display when the text is empty. |
| **android:hyphenationFrequency** (/reference/android/widget/TextView#attr_android:hyphenationFrequency) | Frequency of automatic hyphenation. |
| **android:imeActionId** (/reference/android/widget/TextView#attr_android:imeActionId) | Supply a value for **EditorInfo.actionId** (/reference/android/view/inputmethod/EditorInfo#actionId) used when an input method is connected to the text view. |
| **android:imeActionLabel** (/reference/android/widget/TextView#attr_android:imeActionLabel) | Supply a value for **EditorInfo.actionLabel** (/reference/android/view/inputmethod/EditorInfo#actionLabel) used when an input method is connected to the text view. |
| **android:imeOptions** (/reference/android/widget/TextView#attr_android:imeOptions) | Additional features you can enable in an IME associated with an editor to improve the integration with your application. |
| **android:includeFontPadding** (/reference/android/widget/TextView#attr_android:includeFontPadding) | Leave enough room for ascenders and descenders instead of using the font ascent and descent strictly. |
| **android:inputMethod** (/reference/android/widget/TextView#attr_android:inputMethod) | If set, specifies that this TextView should use the specified input method (specified by fully-qualified class name). |

| | |
|---|---|
| **android:inputType** (/reference/android/widget/TextView#attr_android:inputType) | The type of data being placed in a text field, used to help an input method decide how to let the user enter text. |
| **android:justificationMode** (/reference/android/widget/TextView#attr_android:justificationMode) | Mode for justification. |
| **android:lastBaselineToBottomHeight** (/reference/android/widget/TextView#attr_android:lastBaselineToBottomHeight) | Distance from the bottom of the TextView to the last text baseline. |
| **android:letterSpacing** (/reference/android/widget/TextView#attr_android:letterSpacing) | Text letter-spacing. |
| **android:lineHeight** (/reference/android/widget/TextView#attr_android:lineHeight) | Explicit height between lines of text. |
| **android:lineSpacingExtra** (/reference/android/widget/TextView#attr_android:lineSpacingExtra) | Extra spacing between lines of text. |
| **android:lineSpacingMultiplier** (/reference/android/widget/TextView#attr_android:lineSpacingMultiplier) | Extra spacing between lines of text, as a multiplier. |
| **android:lines** (/reference/android/widget/TextView#attr_android:lines) | Makes the TextView be exactly this many lines tall. |
| **android:linksClickable** (/reference/android/widget/TextView#attr_android:linksClickable) | If set to false, keeps the movement method from being set to the link movement method even if autoLink causes links to be found. |
| **android:marqueeRepeatLimit** (/reference/android/widget/TextView#attr_android:marqueeRepeatLimit) | The number of times to repeat the marquee animation. |
| **android:maxEms** (/reference/android/widget/TextView#attr_android:maxEms) | Makes the TextView be at most this many ems wide. |
| **android:maxHeight** (/reference/android/widget/TextView#attr_android:maxHeight) | Makes the TextView be at most this many pixels tall. |
| **android:maxLength** (/reference/android/widget/TextView#attr_android:maxLength) | Set an input filter to constrain the text length to the specified number. |
| **android:maxLines** (/reference/android/widget/TextView#attr_android:maxLines) | Makes the TextView be at most this many lines tall. |
| **android:maxWidth** (/reference/android/widget/TextView#attr_android:maxWidth) | Makes the TextView be at most this many pixels wide. |
| **android:minEms** (/reference/android/widget/TextView#attr_android:minEms) | Makes the TextView be at least this many ems wide. |

| | |
|---|---|
| **android:minHeight** (/reference/android/widget/TextView#attr_android:minHeight) | Makes the TextView be at least this many pixels tall. |
| **android:minLines** (/reference/android/widget/TextView#attr_android:minLines) | Makes the TextView be at least this many lines tall. |
| **android:minWidth** (/reference/android/widget/TextView#attr_android:minWidth) | Makes the TextView be at least this many pixels wide. |
| **android:numeric** (/reference/android/widget/TextView#attr_android:numeric) | If set, specifies that this TextView has a numeric input method. |
| **android:password** (/reference/android/widget/TextView#attr_android:password) | Whether the characters of the field are displayed as password dots instead of themselves. |
| **android:phoneNumber** (/reference/android/widget/TextView#attr_android:phoneNumber) | If set, specifies that this TextView has a phone number input method. |
| **android:privateImeOptions** (/reference/android/widget/TextView#attr_android:privateImeOptions) | An addition content type description to supply to the input method attached to the text view, which is private to the implementation of the input method. |
| **android:scrollHorizontally** (/reference/android/widget/TextView#attr_android:scrollHorizontally) | Whether the text is allowed to be wider than the view (and therefore can be scrolled horizontally). |
| **android:selectAllOnFocus** (/reference/android/widget/TextView#attr_android:selectAllOnFocus) | If the text is selectable, select it all when the view takes focus. |
| **android:shadowColor** (/reference/android/widget/TextView#attr_android:shadowColor) | Place a blurred shadow of text underneath the text, drawn with the specified color. |
| **android:shadowDx** (/reference/android/widget/TextView#attr_android:shadowDx) | Horizontal offset of the text shadow. |
| **android:shadowDy** (/reference/android/widget/TextView#attr_android:shadowDy) | Vertical offset of the text shadow. |
| **android:shadowRadius** (/reference/android/widget/TextView#attr_android:shadowRadius) | Blur radius of the text shadow. |
| **android:singleLine** (/reference/android/widget/TextView#attr_android:singleLine) | Constrains the text to a single horizontally scrolling line instead of |

| | |
|---|---|
| | letting it wrap onto multiple lines, and advances focus instead of inserting a newline when you press the enter key. |
| `android:text` (/reference/android/widget/TextView#attr_android:text) | Text to display. |
| `android:textAllCaps` (/reference/android/widget/TextView#attr_android:textAllCaps) | Present the text in ALL CAPS. |
| `android:textAppearance` (/reference/android/widget/TextView#attr_android:textAppearance) | Base text color, typeface, size, and style. |
| `android:textColor` (/reference/android/widget/TextView#attr_android:textColor) | Text color. |
| `android:textColorHighlight` (/reference/android/widget/TextView#attr_android:textColorHighlight) | Color of the text selection highlight. |
| `android:textColorHint` (/reference/android/widget/TextView#attr_android:textColorHint) | Color of the hint text. |
| `android:textColorLink` (/reference/android/widget/TextView#attr_android:textColorLink) | Text color for links. |
| `android:textCursorDrawable` (/reference/android/widget/TextView#attr_android:textCursorDrawable) | Reference to a drawable that will be drawn under the insertion cursor. |
| `android:textFontWeight` (/reference/android/widget/TextView#attr_android:textFontWeight) | Weight for the font used in the TextView. |
| `android:textIsSelectable` (/reference/android/widget/TextView#attr_android:textIsSelectable) | Indicates that the content of a non-editable text can be selected. |
| `android:textScaleX` (/reference/android/widget/TextView#attr_android:textScaleX) | Sets the horizontal scaling factor for the text. |
| `android:textSelectHandle` (/reference/android/widget/TextView#attr_android:textSelectHandle) | Reference to a drawable that will be used to display a text selection anchor for positioning the cursor within text. |
| `android:textSelectHandleLeft` (/reference/android/widget/TextView#attr_android:textSelectHandleLeft) | Reference to a drawable that will be used to display a text selection anchor on the left side of a selection region. |
| `android:textSelectHandleRight` (/reference/android/widget/TextView#attr_android:textSelectHandleRight) | Reference to a drawable that will be used to display a text selection anchor on the right side of a selection region. |

| | |
|---|---|
| **android:textSize** (/reference/android/widget/TextView#attr_android:textSize) | Size of the text. |
| **android:textStyle** (/reference/android/widget/TextView#attr_android:textStyle) | Style (normal, bold, italic, bold\|italic) for the text. |
| **android:typeface** (/reference/android/widget/TextView#attr_android:typeface) | Typeface (normal, sans, serif, monospace) for the text. |
| **android:width** (/reference/android/widget/TextView#attr_android:width) | Makes the TextView be exactly this wide. |

## Inherited XML attributes

From class **android.view.View** (/reference/android/view/View)

| | |
|---|---|
| **android:accessibilityHeading** (/reference/android/view/View#attr_android:accessibilityHeading) | Whether or not this view is a heading for accessibility purposes. |
| **android:accessibilityLiveRegion** (/reference/android/view/View#attr_android:accessibilityLiveRegion) | Indicates to accessibility services whether the user should be notified when this view changes. |
| **android:accessibilityPaneTitle** (/reference/android/view/View#attr_android:accessibilityPaneTitle) | The title this view should present to accessibility as a pane title. |
| **android:accessibilityTraversalAfter** (/reference/android/view/View#attr_android:accessibilityTraversalAfter) | Sets the id of a view after which this one is visited in accessibility traversal. |
| **android:accessibilityTraversalBefore** (/reference/android/view/View#attr_android:accessibilityTraversalBefore) | Sets the id of a view before which this one is visited in accessibility traversal. |
| **android:alpha** (/reference/android/view/View#attr_android:alpha) | alpha property of the view, as a value between 0 (completely transparent) and 1 (completely opaque). |
| **android:autofillHints** (/reference/android/view/View#attr_android:autofillHints) | Describes the content of a view so that a autofill service can fill in the appropriate data. |

| | |
|---|---|
| **android:autofilledHighlight** (/reference/android/view/View#attr_android:autofilledHighlight) | Drawable to be drawn over the view to mark it as autofilled<br>May be a reference to another resource, in the form "**@[+][*package*:]*type*/*name***" or a theme attribute in the form "**?[*package*:]*type*/*name***". |
| **android:background** (/reference/android/view/View#attr_android:background) | A drawable to use as the background. |
| **android:backgroundTint** (/reference/android/view/View#attr_android:backgroundTint) | Tint to apply to the background. |
| **android:backgroundTintMode** (/reference/android/view/View#attr_android:backgroundTintMode) | Blending mode used to apply the background tint. |
| **android:clickable** (/reference/android/view/View#attr_android:clickable) | Defines whether this view reacts to click events. |
| **android:contentDescription** (/reference/android/view/View#attr_android:contentDescription) | Defines text that briefly describes content of the view. |
| **android:contextClickable** (/reference/android/view/View#attr_android:contextClickable) | Defines whether this view reacts to context click events. |
| **android:defaultFocusHighlightEnabled** (/reference/android/view/View#attr_android:defaultFocusHighlightEnabled) | Whether this View should use a default focus highlight when it gets focused but doesn't have **R.attr.state_focused** (/reference/android/R.attr#state_focused) defined in its background. |
| **android:drawingCacheQuality** (/reference/android/view/View#attr_android:drawingCacheQuality) | Defines the quality of translucent drawing caches. |
| **android:duplicateParentState** (/reference/android/view/View#attr_android:duplicateParentState) | When this attribute is set to true, the view gets its drawable state (focused, pressed, etc.) from its direct parent rather than from itself. |
| **android:elevation** (/reference/android/view/View#attr_android:elevation) | base z depth of the view. |

| | |
|---|---|
| **android:fadeScrollbars** (/reference/android/view/View#attr_android:fadeScrollbars) | Defines whether to fade out scrollbars when they are not in use. |
| **android:fadingEdgeLength** (/reference/android/view/View#attr_android:fadingEdgeLength) | Defines the length of the fading edges. |
| **android:filterTouchesWhenObscured** (/reference/android/view/View#attr_android:filterTouchesWhenObscured) | Specifies whether to filter touches when the view's window is obscured by another visible window. |
| **android:fitsSystemWindows** (/reference/android/view/View#attr_android:fitsSystemWindows) | Boolean internal attribute to adjust view layout based on system windows such as the status bar. |
| **android:focusable** (/reference/android/view/View#attr_android:focusable) | Controls whether a view can take focus. |
| **android:focusableInTouchMode** (/reference/android/view/View#attr_android:focusableInTouchMode) | Boolean that controls whether a view can take focus while in touch mode. |
| **android:focusedByDefault** (/reference/android/view/View#attr_android:focusedByDefault) | Whether this view is a default-focus view. |
| **android:forceHasOverlappingRendering** (/reference/android/view/View#attr_android:forceHasOverlappingRendering) | Whether this view has elements that may overlap when drawn. |
| **android:foreground** (/reference/android/view/View#attr_android:foreground) | Defines the drawable to draw over the content. |
| **android:foregroundGravity** (/reference/android/view/View#attr_android:foregroundGravity) | Defines the gravity to apply to the foreground drawable. |
| **android:foregroundTint** (/reference/android/view/View#attr_android:foregroundTint) | Tint to apply to the foreground. |
| **android:foregroundTintMode** (/reference/android/view/View#attr_android:foregroundTintMode) | Blending mode used to apply the foreground tint. |
| **android:hapticFeedbackEnabled** (/reference/android/view/View#attr_android:hapticFeedbackEnabled) | Boolean that controls whether a view should have haptic feedback enabled for events such as long presses. |

| | |
|---|---|
| **android:id** (/reference/android/view/View#attr_android:id) | Supply an identifier name for this view, to later retrieve it with **View.findViewById()** (/reference/android/view/View#findViewById(int)) or **Activity.findViewById()** (/reference/android/app/Activity#findViewById(int)). |
| **android:importantForAccessibility** (/reference/android/view/View#attr_android:importantForAccessibility) | Describes whether or not this view is important for accessibility. |
| **android:importantForAutofill** (/reference/android/view/View#attr_android:importantForAutofill) | Hints the Android System whether the view node associated with this View should be included in a view structure used for autofill purposes. |
| **android:importantForContentCapture** (/reference/android/view/View#attr_android:importantForContentCapture) | Hints the Android System whether the view node associated with this View should be use for content capture purposes. |
| **android:isScrollContainer** (/reference/android/view/View#attr_android:isScrollContainer) | Set this if the view will serve as a scrolling container, meaning that it can be resized to shrink its overall window so that there will be space for an input method. |
| **android:keepScreenOn** (/reference/android/view/View#attr_android:keepScreenOn) | Controls whether the view's window should keep the screen on while visible. |
| **android:keyboardNavigationCluster** (/reference/android/view/View#attr_android:keyboardNavigationCluster) | Whether this view is a root of a keyboard navigation cluster. |
| **android:layerType** (/reference/android/view/View#attr_android:layerType) | Specifies the type of layer backing this view. |
| **android:layoutDirection** (/reference/android/view/View#attr_android:layoutDirection) | Defines the direction of layout drawing. |
| **android:longClickable** (/reference/android/view/View#attr_android:longClickable) | Defines whether this view reacts to long click events. |

| | |
|---|---|
| **android:minHeight** (/reference/android/view/View#attr_android:minHeight) | Defines the minimum height of the view. |
| **android:minWidth** (/reference/android/view/View#attr_android:minWidth) | Defines the minimum width of the view. |
| **android:nextClusterForward** (/reference/android/view/View#attr_android:nextClusterForward) | Defines the next keyboard navigation cluster. |
| **android:nextFocusDown** (/reference/android/view/View#attr_android:nextFocusDown) | Defines the next view to give focus to when the next focus is **View.FOCUS_DOWN** (/reference/android/view/View#FOCUS_DOWN) If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, a **RuntimeException** (/reference/java/lang/RuntimeException) will result when the reference is accessed. |
| **android:nextFocusForward** (/reference/android/view/View#attr_android:nextFocusForward) | Defines the next view to give focus to when the next focus is **View.FOCUS_FORWARD** (/reference/android/view/View#FOCUS_FORWARD) If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, a **RuntimeException** (/reference/java/lang/RuntimeException) will result when the reference is accessed. |
| **android:nextFocusLeft** (/reference/android/view/View#attr_android:nextFocusLeft) | Defines the next view to give focus to when the next focus is **View.FOCUS_LEFT** (/reference/android/view/View#FOCUS_LEFT). |
| **android:nextFocusRight** (/reference/android/view/View#attr_android:nextFocusRight) | Defines the next view to give focus to when the next focus is **View.FOCUS_RIGHT** (/reference/android/view/View#FOCUS_RIGHT) If the reference refers to a view that does not exist or is |

| | |
|---|---|
| | part of a hierarchy that is invisible, a **RuntimeException** (/reference/java/lang/RuntimeException) will result when the reference is accessed. |
| **android:nextFocusUp** (/reference/android/view/View#attr_android:nextFocusUp) | Defines the next view to give focus to when the next focus is **View.FOCUS_UP** (/reference/android/view/View#FOCUS_UP) If the reference refers to a view that does not exist or is part of a hierarchy that is invisible, a **RuntimeException** (/reference/java/lang/RuntimeException) will result when the reference is accessed. |
| **android:onClick** (/reference/android/view/View#attr_android:onClick) | Name of the method in this View's context to invoke when the view is clicked. |
| **android:outlineAmbientShadowColor** (/reference/android/view/View#attr_android:outlineAmbientShadowColor) | Sets the color of the ambient shadow that is drawn when the view has a positive Z or elevation value. |
| **android:outlineSpotShadowColor** (/reference/android/view/View#attr_android:outlineSpotShadowColor) | Sets the color of the spot shadow that is drawn when the view has a positive Z or elevation value. |
| **android:padding** (/reference/android/view/View#attr_android:padding) | Sets the padding, in pixels, of all four edges. |
| **android:paddingBottom** (/reference/android/view/View#attr_android:paddingBottom) | Sets the padding, in pixels, of the bottom edge; see **R.attr.padding** (/reference/android/R.attr#padding). |
| **android:paddingEnd** (/reference/android/view/View#attr_android:paddingEnd) | Sets the padding, in pixels, of the end edge; see **R.attr.padding** (/reference/android/R.attr#padding). |

| | |
|---|---|
| **android:paddingHorizontal** (/reference/android/view/View#attr_android:paddingHorizontal) | Sets the padding, in pixels, of the left and right edges; see **R.attr.padding** (/reference/android/R.attr#padding). |
| **android:paddingLeft** (/reference/android/view/View#attr_android:paddingLeft) | Sets the padding, in pixels, of the left edge; see **R.attr.padding** (/reference/android/R.attr#padding). |
| **android:paddingRight** (/reference/android/view/View#attr_android:paddingRight) | Sets the padding, in pixels, of the right edge; see **R.attr.padding** (/reference/android/R.attr#padding). |
| **android:paddingStart** (/reference/android/view/View#attr_android:paddingStart) | Sets the padding, in pixels, of the start edge; see **R.attr.padding** (/reference/android/R.attr#padding). |
| **android:paddingTop** (/reference/android/view/View#attr_android:paddingTop) | Sets the padding, in pixels, of the top edge; see **R.attr.padding** (/reference/android/R.attr#padding). |
| **android:paddingVertical** (/reference/android/view/View#attr_android:paddingVertical) | Sets the padding, in pixels, of the top and bottom edges; see **R.attr.padding** (/reference/android/R.attr#padding). |
| **android:requiresFadingEdge** (/reference/android/view/View#attr_android:requiresFadingEdge) | Defines which edges should be faded on scrolling. |
| **android:rotation** (/reference/android/view/View#attr_android:rotation) | rotation of the view, in degrees. |
| **android:rotationX** (/reference/android/view/View#attr_android:rotationX) | rotation of the view around the x axis, in degrees. |
| **android:rotationY** (/reference/android/view/View#attr_android:rotationY) | rotation of the view around the y axis, in degrees. |
| **android:saveEnabled** (/reference/android/view/View#attr_android:saveEnabled) | If false, no state will be saved for this view when it is |

| | |
|---|---|
| | being frozen. |
| `android:scaleX` (/reference/android/view/View#attr_android:scaleX) | scale of the view in the x direction. |
| `android:scaleY` (/reference/android/view/View#attr_android:scaleY) | scale of the view in the y direction. |
| `android:screenReaderFocusable` (/reference/android/view/View#attr_android:screenReaderFocusable) | Whether this view should be treated as a focusable unit by screen reader accessibility tools. |
| `android:scrollIndicators` (/reference/android/view/View#attr_android:scrollIndicators) | Defines which scroll indicators should be displayed when the view can be scrolled. |
| `android:scrollX` (/reference/android/view/View#attr_android:scrollX) | The initial horizontal scroll offset, in pixels. |
| `android:scrollY` (/reference/android/view/View#attr_android:scrollY) | The initial vertical scroll offset, in pixels. |
| `android:scrollbarAlwaysDrawHorizontalTrack` (/reference/android/view/View#attr_android:scrollbarAlwaysDrawHorizontalTrack) | Defines whether the horizontal scrollbar track should always be drawn. |
| `android:scrollbarAlwaysDrawVerticalTrack` (/reference/android/view/View#attr_android:scrollbarAlwaysDrawVerticalTrack) | Defines whether the vertical scrollbar track should always be drawn. |
| `android:scrollbarDefaultDelayBeforeFade` (/reference/android/view/View#attr_android:scrollbarDefaultDelayBeforeFade) | Defines the delay in milliseconds that a scrollbar waits before fade out. |
| `android:scrollbarFadeDuration` (/reference/android/view/View#attr_android:scrollbarFadeDuration) | Defines the delay in milliseconds that a scrollbar takes to fade out. |
| `android:scrollbarSize` (/reference/android/view/View#attr_android:scrollbarSize) | Sets the width of vertical scrollbars and height of horizontal scrollbars. |
| `android:scrollbarStyle` (/reference/android/view/View#attr_android:scrollbarStyle) | Controls the scrollbar style and position. |
| `android:scrollbarThumbHorizontal` (/reference/android/view/View#attr_android:scrollbarThumbHorizontal) | Defines the horizontal scrollbar thumb drawable. |

| | |
|---|---|
| **android:scrollbarThumbVertical** (/reference/android/view/View#attr_android:scrollbarThumbVertical) | Defines the vertical scrollbar thumb drawable. |
| **android:scrollbarTrackHorizontal** (/reference/android/view/View#attr_android:scrollbarTrackHorizontal) | Defines the horizontal scrollbar track drawable. |
| **android:scrollbarTrackVertical** (/reference/android/view/View#attr_android:scrollbarTrackVertical) | Defines the vertical scrollbar track drawable. |
| **android:scrollbars** (/reference/android/view/View#attr_android:scrollbars) | Defines which scrollbars should be displayed on scrolling or not. |
| **android:soundEffectsEnabled** (/reference/android/view/View#attr_android:soundEffectsEnabled) | Boolean that controls whether a view should have sound effects enabled for events such as clicking and touching. |
| **android:stateListAnimator** (/reference/android/view/View#attr_android:stateListAnimator) | Sets the state-based animator for the View. |
| **android:tag** (/reference/android/view/View#attr_android:tag) | Supply a tag for this view containing a String, to be retrieved later with **View.getTag()** (/reference/android/view/View#getTag()) or searched for with **View.findViewWithTag()** (/reference/android/view/View#findViewWithTag(java.lang.Object)) . |
| **android:textAlignment** (/reference/android/view/View#attr_android:textAlignment) | Defines the alignment of the text. |
| **android:textDirection** (/reference/android/view/View#attr_android:textDirection) | Defines the direction of the text. |
| **android:theme** (/reference/android/view/View#attr_android:theme) | Specifies a theme override for a view. |
| **android:tooltipText** (/reference/android/view/View#attr_android:tooltipText) | Defines text displayed in a small popup window on hover or long press. |
| **android:transformPivotX** (/reference/android/view/View#attr_android:transformPivotX) | x location of the pivot point around which the view |

| | |
|---|---|
| | will rotate and scale. |
| **android:transformPivotY** (/reference/android/view/View#attr_android:transformPivotY) | y location of the pivot point around which the view will rotate and scale. |
| **android:transitionName** (/reference/android/view/View#attr_android:transitionName) | Names a View such that it can be identified for Transitions. |
| **android:translationX** (/reference/android/view/View#attr_android:translationX) | translation in x of the view. |
| **android:translationY** (/reference/android/view/View#attr_android:translationY) | translation in y of the view. |
| **android:translationZ** (/reference/android/view/View#attr_android:translationZ) | translation in z of the view. |
| **android:visibility** (/reference/android/view/View#attr_android:visibility) | Controls the initial visibility of the view. |

## Constants

| | |
|---|---|
| **int** | **AUTO_SIZE_TEXT_TYPE_NONE** (/reference/android/widget/TextView#AUTO_SIZE_TEXT_TYPE_NONE)<br><br>The TextView does not auto-size text (default). |
| **int** | **AUTO_SIZE_TEXT_TYPE_UNIFORM** (/reference/android/widget/TextView#AUTO_SIZE_TEXT_TYPE_UNIFORM)<br><br>The TextView scales text size both horizontally and vertically to fit within the container. |

## Inherited constants

From class **android.view.View** (/reference/android/view/View)

| | |
|---|---|
| **int** | **ACCESSIBILITY_LIVE_REGION_ASSERTIVE** (/reference/android/view/View#ACCESSIBILITY_LIVE_REGION_ASSERTIVE) |

| | |
|---|---|
| | Live region mode specifying that accessibility services should interrupt ongoing speech to immediately announce changes to this view. |
| `int` | **ACCESSIBILITY_LIVE_REGION_NONE** (/reference/android/view/View#ACCESSIBILITY_LIVE_REGION_NONE)<br><br>Live region mode specifying that accessibility services should not automatically announce changes to this view. |
| `int` | **ACCESSIBILITY_LIVE_REGION_POLITE** (/reference/android/view/View#ACCESSIBILITY_LIVE_REGION_POLITE)<br><br>Live region mode specifying that accessibility services should announce changes to this view. |
| `int` | **AUTOFILL_FLAG_INCLUDE_NOT_IMPORTANT_VIEWS** (/reference/android/view/View#AUTOFILL_FLAG_INCLUDE_NOT_IMPORTANT_VIEWS)<br><br>Flag requesting you to add views that are marked as not important for autofill (see **setImportantForAutofill(int)** (/reference/android/view/View#setImportantForAutofill(int))) to a **ViewStructure** (/reference/android/view/ViewStructure). |
| **String** (/reference/java/lang/String) | **AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_DATE** (/reference/android/view/View#AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_DATE)<br><br>Hint indicating that this view can be autofilled with a credit card expiration date. |
| **String** (/reference/java/lang/String) | **AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_DAY** (/reference/android/view/View#AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_DAY)<br><br>Hint indicating that this view can be autofilled with a credit card expiration day. |
| **String** (/reference/java/lang/String) | **AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_MONTH** (/reference/android/view/View#AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_MONTH)<br><br>Hint indicating that this view can be autofilled with a credit card expiration month. |
| **String** (/reference/java/lang/String) | **AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_YEAR** (/reference/android/view/View#AUTOFILL_HINT_CREDIT_CARD_EXPIRATION_YEAR)<br><br>Hint indicating that this view can be autofilled with a credit card expiration year. |
| **String** (/reference/java/lang/String) | **AUTOFILL_HINT_CREDIT_CARD_NUMBER** (/reference/android/view/View#AUTOFILL_HINT_CREDIT_CARD_NUMBER)<br><br>Hint indicating that this view can be autofilled with a credit card number. |

| | |
|---|---|
| String (/reference/java/lang/String) | AUTOFILL_HINT_CREDIT_CARD_SECURITY_CODE (/reference/android/view/View#AUTOFILL_HINT_CREDIT_CARD_SECURITY_CODE)<br>Hint indicating that this view can be autofilled with a credit card security code. |
| String (/reference/java/lang/String) | AUTOFILL_HINT_EMAIL_ADDRESS (/reference/android/view/View#AUTOFILL_HINT_EMAIL_ADDRESS)<br>Hint indicating that this view can be autofilled with an email address. |
| String (/reference/java/lang/String) | AUTOFILL_HINT_NAME (/reference/android/view/View#AUTOFILL_HINT_NAME)<br>Hint indicating that this view can be autofilled with a user's real name. |
| String (/reference/java/lang/String) | AUTOFILL_HINT_PASSWORD (/reference/android/view/View#AUTOFILL_HINT_PASSWORD)<br>Hint indicating that this view can be autofilled with a password. |
| String (/reference/java/lang/String) | AUTOFILL_HINT_PHONE (/reference/android/view/View#AUTOFILL_HINT_PHONE)<br>Hint indicating that this view can be autofilled with a phone number. |
| String (/reference/java/lang/String) | AUTOFILL_HINT_POSTAL_ADDRESS (/reference/android/view/View#AUTOFILL_HINT_POSTAL_ADDRESS)<br>Hint indicating that this view can be autofilled with a postal address. |
| String (/reference/java/lang/String) | AUTOFILL_HINT_POSTAL_CODE (/reference/android/view/View#AUTOFILL_HINT_POSTAL_CODE)<br>Hint indicating that this view can be autofilled with a postal code. |
| String (/reference/java/lang/String) | AUTOFILL_HINT_USERNAME (/reference/android/view/View#AUTOFILL_HINT_USERNAME)<br>Hint indicating that this view can be autofilled with a username. |
| int | AUTOFILL_TYPE_DATE (/reference/android/view/View#AUTOFILL_TYPE_DATE)<br>Autofill type for a field that contains a date, which is represented by a long representing the number of milliseconds since the standard base time known as |

| | |
|---|---|
| | "the epoch", namely January 1, 1970, 00:00:00 GMT (see `Date.getTime()` (/reference/java/util/Date#getTime()). |
| `int` | `AUTOFILL_TYPE_LIST` (/reference/android/view/View#AUTOFILL_TYPE_LIST)<br><br>Autofill type for a selection list field, which is filled by an `int` representing the element index inside the list (starting at `0`). |
| `int` | `AUTOFILL_TYPE_NONE` (/reference/android/view/View#AUTOFILL_TYPE_NONE)<br><br>Autofill type for views that cannot be autofilled. |
| `int` | `AUTOFILL_TYPE_TEXT` (/reference/android/view/View#AUTOFILL_TYPE_TEXT)<br><br>Autofill type for a text field, which is filled by a `CharSequence` (/reference/java/lang/CharSequence). |
| `int` | `AUTOFILL_TYPE_TOGGLE` (/reference/android/view/View#AUTOFILL_TYPE_TOGGLE)<br><br>Autofill type for a togglable field, which is filled by a `boolean`. |
| `int` | `DRAG_FLAG_GLOBAL` (/reference/android/view/View#DRAG_FLAG_GLOBAL)<br><br>Flag indicating that a drag can cross window boundaries. |
| `int` | `DRAG_FLAG_GLOBAL_PERSISTABLE_URI_PERMISSION` (/reference/android/view/View#DRAG_FLAG_GLOBAL_PERSISTABLE_URI_PERMISSION)<br><br>When this flag is used with `DRAG_FLAG_GLOBAL_URI_READ` (/reference/android/view/View#DRAG_FLAG_GLOBAL_URI_READ) and/or `DRAG_FLAG_GLOBAL_URI_WRITE` (/reference/android/view/View#DRAG_FLAG_GLOBAL_URI_WRITE), the URI permission grant can be persisted across device reboots until explicitly revoked with `Context.revokeUriPermission(Uri, int)` (/reference/android/content/Context#revokeUriPermission(android.net.Uri,%20int)) Context.revokeUriPermission}. |
| `int` | `DRAG_FLAG_GLOBAL_PREFIX_URI_PERMISSION` (/reference/android/view/View#DRAG_FLAG_GLOBAL_PREFIX_URI_PERMISSION)<br><br>When this flag is used with `DRAG_FLAG_GLOBAL_URI_READ` (/reference/android/view/View#DRAG_FLAG_GLOBAL_URI_READ) and/or `DRAG_FLAG_GLOBAL_URI_WRITE` (/reference/android/view/View#DRAG_FLAG_GLOBAL_URI_WRITE), the URI permission grant applies to any URI that is a prefix match against the original granted URI. |

| | |
|---|---|
| **int** | **DRAG_FLAG_GLOBAL_URI_READ** (/reference/android/view/View#DRAG_FLAG_GLOBAL_URI_READ)<br><br>When this flag is used with **DRAG_FLAG_GLOBAL** (/reference/android/view/View#DRAG_FLAG_GLOBAL), the drag recipient will be able to request read access to the content URI(s) contained in the **ClipData** (/reference/android/content/ClipData) object. |
| **int** | **DRAG_FLAG_GLOBAL_URI_WRITE** (/reference/android/view/View#DRAG_FLAG_GLOBAL_URI_WRITE)<br><br>When this flag is used with **DRAG_FLAG_GLOBAL** (/reference/android/view/View#DRAG_FLAG_GLOBAL), the drag recipient will be able to request write access to the content URI(s) contained in the **ClipData** (/reference/android/content/ClipData) object. |
| **int** | **DRAG_FLAG_OPAQUE** (/reference/android/view/View#DRAG_FLAG_OPAQUE)<br><br>Flag indicating that the drag shadow will be opaque. |
| **int** | **DRAWING_CACHE_QUALITY_AUTO** (/reference/android/view/View#DRAWING_CACHE_QUALITY_AUTO)<br><br>*This constant was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations,* **setLayerType(int, android.graphics.Paint)** (/reference/android/view/View#setLayerType(int,%20android.graphics.Paint)) *handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a* **Canvas** (/reference/android/graphics/Canvas) *from either a* **Bitmap** (/reference/android/graphics/Bitmap) *or* **Picture** (/reference/android/graphics/Picture) *and call* **draw(android.graphics.Canvas)** (/reference/android/view/View#draw(android.graphics.Canvas)) *on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as* **Config.HARDWARE** (/reference/android/graphics/Bitmap.Config#HARDWARE) *bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the* **PixelCopy** (/reference/android/view/PixelCopy) *API is recommended.* |
| **int** | **DRAWING_CACHE_QUALITY_HIGH** (/reference/android/view/View#DRAWING_CACHE_QUALITY_HIGH)<br><br>*This constant was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations,* **setLayerType(int, android.graphics.Paint)** (/reference/android/view/View#setLayerType(int,%20android.graphics.Paint)) *handles this with hardware rendering. For* |

| | |
|---|---|
| | *software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a* **Canvas** (/reference/android/graphics/Canvas) *from either a* **Bitmap** (/reference/android/graphics/Bitmap) *or* **Picture** (/reference/android/graphics/Picture) *and call* **draw(android.graphics.Canvas)** (/reference/android/view/View#draw(android.graphics.Canvas)) *on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as* **Config.HARDWARE** (/reference/android/graphics/Bitmap.Config#HARDWARE) *bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the* **PixelCopy** (/reference/android/view/PixelCopy) *API is recommended.* |
| **int** | **DRAWING_CACHE_QUALITY_LOW** (/reference/android/view/View#DRAWING_CACHE_QUALITY_LOW)<br><br>*This constant was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations,* **setLayerType(int, android.graphics.Paint)** (/reference/android/view/View#setLayerType(int,%20android.graphics.Paint)) *handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a* **Canvas** (/reference/android/graphics/Canvas) *from either a* **Bitmap** (/reference/android/graphics/Bitmap) *or* **Picture** (/reference/android/graphics/Picture) *and call* **draw(android.graphics.Canvas)** (/reference/android/view/View#draw(android.graphics.Canvas)) *on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as* **Config.HARDWARE** (/reference/android/graphics/Bitmap.Config#HARDWARE) *bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the* **PixelCopy** (/reference/android/view/PixelCopy) *API is recommended.* |
| **int** | **FIND_VIEWS_WITH_CONTENT_DESCRIPTION** (/reference/android/view/View#FIND_VIEWS_WITH_CONTENT_DESCRIPTION)<br><br>Find find views that contain the specified content description. |
| **int** | **FIND_VIEWS_WITH_TEXT** (/reference/android/view/View#FIND_VIEWS_WITH_TEXT)<br><br>Find views that render the specified text. |
| **int** | **FOCUSABLE** (/reference/android/view/View#FOCUSABLE)<br><br>This view wants keystrokes. |
| **int** | **FOCUSABLES_ALL** (/reference/android/view/View#FOCUSABLES_ALL) |

| | |
|---|---|
| | View flag indicating whether **addFocusables(java.util.ArrayList, int, int)** (/reference/android/view/View#addFocusables(java.util.ArrayList<android.view.View>,%20int,%20int)) should add all focusable Views regardless if they are focusable in touch mode. |
| **int** | **FOCUSABLES_TOUCH_MODE** (/reference/android/view/View#FOCUSABLES_TOUCH_MODE)<br><br>View flag indicating whether **addFocusables(java.util.ArrayList, int, int)** (/reference/android/view/View#addFocusables(java.util.ArrayList<android.view.View>,%20int,%20int)) should add only Views focusable in touch mode. |
| **int** | **FOCUSABLE_AUTO** (/reference/android/view/View#FOCUSABLE_AUTO)<br><br>This view determines focusability automatically. |
| **int** | **FOCUS_BACKWARD** (/reference/android/view/View#FOCUS_BACKWARD)<br><br>Use with **focusSearch(int)** (/reference/android/view/View#focusSearch(int)). |
| **int** | **FOCUS_DOWN** (/reference/android/view/View#FOCUS_DOWN)<br><br>Use with **focusSearch(int)** (/reference/android/view/View#focusSearch(int)). |
| **int** | **FOCUS_FORWARD** (/reference/android/view/View#FOCUS_FORWARD)<br><br>Use with **focusSearch(int)** (/reference/android/view/View#focusSearch(int)). |
| **int** | **FOCUS_LEFT** (/reference/android/view/View#FOCUS_LEFT)<br><br>Use with **focusSearch(int)** (/reference/android/view/View#focusSearch(int)). |
| **int** | **FOCUS_RIGHT** (/reference/android/view/View#FOCUS_RIGHT)<br><br>Use with **focusSearch(int)** (/reference/android/view/View#focusSearch(int)). |
| **int** | **FOCUS_UP** (/reference/android/view/View#FOCUS_UP) |

| | |
|---|---|
| | Use with `focusSearch(int)` (/reference/android/view/View#focusSearch(int)). |
| **int** | `GONE` (/reference/android/view/View#GONE)<br><br>This view is invisible, and it doesn't take any space for layout purposes. |
| **int** | `HAPTIC_FEEDBACK_ENABLED` (/reference/android/view/View#HAPTIC_FEEDBACK_ENABLED)<br><br>View flag indicating whether this view should have haptic feedback enabled for events such as long presses. |
| **int** | `IMPORTANT_FOR_ACCESSIBILITY_AUTO` (/reference/android/view/View#IMPORTANT_FOR_ACCESSIBILITY_AUTO)<br><br>Automatically determine whether a view is important for accessibility. |
| **int** | `IMPORTANT_FOR_ACCESSIBILITY_NO` (/reference/android/view/View#IMPORTANT_FOR_ACCESSIBILITY_NO)<br><br>The view is not important for accessibility. |
| **int** | `IMPORTANT_FOR_ACCESSIBILITY_NO_HIDE_DESCENDANTS` (/reference/android/view/View#IMPORTANT_FOR_ACCESSIBILITY_NO_HIDE_DESCENDANTS)<br><br>The view is not important for accessibility, nor are any of its descendant views. |
| **int** | `IMPORTANT_FOR_ACCESSIBILITY_YES` (/reference/android/view/View#IMPORTANT_FOR_ACCESSIBILITY_YES)<br><br>The view is important for accessibility. |
| **int** | `IMPORTANT_FOR_AUTOFILL_AUTO` (/reference/android/view/View#IMPORTANT_FOR_AUTOFILL_AUTO)<br><br>Automatically determine whether a view is important for autofill. |
| **int** | `IMPORTANT_FOR_AUTOFILL_NO` (/reference/android/view/View#IMPORTANT_FOR_AUTOFILL_NO)<br><br>The view is not important for autofill, but its children (if any) will be traversed. |

| | |
|---|---|
| int | IMPORTANT_FOR_AUTOFILL_NO_EXCLUDE_DESCENDANTS (/reference/android/view/View#IMPORTANT_FOR_AUTOFILL_NO_EXCLUDE_DESCENDANTS)<br><br>The view is not important for autofill, and its children (if any) will not be traversed. |
| int | IMPORTANT_FOR_AUTOFILL_YES (/reference/android/view/View#IMPORTANT_FOR_AUTOFILL_YES)<br><br>The view is important for autofill, and its children (if any) will be traversed. |
| int | IMPORTANT_FOR_AUTOFILL_YES_EXCLUDE_DESCENDANTS (/reference/android/view/View#IMPORTANT_FOR_AUTOFILL_YES_EXCLUDE_DESCENDANTS)<br><br>The view is important for autofill, but its children (if any) will not be traversed. |
| int | IMPORTANT_FOR_CONTENT_CAPTURE_AUTO (/reference/android/view/View#IMPORTANT_FOR_CONTENT_CAPTURE_AUTO)<br><br>Automatically determine whether a view is important for content capture. |
| int | IMPORTANT_FOR_CONTENT_CAPTURE_NO (/reference/android/view/View#IMPORTANT_FOR_CONTENT_CAPTURE_NO)<br><br>The view is not important for content capture, but its children (if any) will be traversed. |
| int | IMPORTANT_FOR_CONTENT_CAPTURE_NO_EXCLUDE_DESCENDANTS (/reference/android/view/View#IMPORTANT_FOR_CONTENT_CAPTURE_NO_EXCLUDE_DESCENDANTS)<br><br>The view is not important for content capture, and its children (if any) will not be traversed. |
| int | IMPORTANT_FOR_CONTENT_CAPTURE_YES (/reference/android/view/View#IMPORTANT_FOR_CONTENT_CAPTURE_YES)<br><br>The view is important for content capture, and its children (if any) will be traversed. |
| int | IMPORTANT_FOR_CONTENT_CAPTURE_YES_EXCLUDE_DESCENDANTS (/reference/android/view/View#IMPORTANT_FOR_CONTENT_CAPTURE_YES_EXCLUDE_DESCENDANTS)<br><br>The view is important for content capture, but its children (if any) will not be traversed. |

| | |
|---|---|
| **int** | **INVISIBLE** (/reference/android/view/View#INVISIBLE)<br><br>This view is invisible, but it still takes up space for layout purposes. |
| **int** | **KEEP_SCREEN_ON** (/reference/android/view/View#KEEP_SCREEN_ON)<br><br>View flag indicating that the screen should remain on while the window containing this view is visible to the user. |
| **int** | **LAYER_TYPE_HARDWARE** (/reference/android/view/View#LAYER_TYPE_HARDWARE)<br><br>Indicates that the view has a hardware layer. |
| **int** | **LAYER_TYPE_NONE** (/reference/android/view/View#LAYER_TYPE_NONE)<br><br>Indicates that the view does not have a layer. |
| **int** | **LAYER_TYPE_SOFTWARE** (/reference/android/view/View#LAYER_TYPE_SOFTWARE)<br><br>Indicates that the view has a software layer. |
| **int** | **LAYOUT_DIRECTION_INHERIT** (/reference/android/view/View#LAYOUT_DIRECTION_INHERIT)<br><br>Horizontal layout direction of this view is inherited from its parent. |
| **int** | **LAYOUT_DIRECTION_LOCALE** (/reference/android/view/View#LAYOUT_DIRECTION_LOCALE)<br><br>Horizontal layout direction of this view is from deduced from the default language script for the locale. |
| **int** | **LAYOUT_DIRECTION_LTR** (/reference/android/view/View#LAYOUT_DIRECTION_LTR)<br><br>Horizontal layout direction of this view is from Left to Right. |
| **int** | **LAYOUT_DIRECTION_RTL** (/reference/android/view/View#LAYOUT_DIRECTION_RTL) |

| | |
|---|---|
| | Horizontal layout direction of this view is from Right to Left. |
| **int** | **MEASURED_HEIGHT_STATE_SHIFT** (/reference/android/view/View#MEASURED_HEIGHT_STATE_SHIFT)<br><br>Bit shift of **MEASURED_STATE_MASK** (/reference/android/view/View#MEASURED_STATE_MASK) to get to the height bits for functions that combine both width and height into a single int, such as **getMeasuredState()** (/reference/android/view/View#getMeasuredState()) and the childState argument of **resolveSizeAndState(int, int, int)** (/reference/android/view/View#resolveSizeAndState(int,%20int,%20int)). |
| **int** | **MEASURED_SIZE_MASK** (/reference/android/view/View#MEASURED_SIZE_MASK)<br><br>Bits of **getMeasuredWidthAndState()** (/reference/android/view/View#getMeasuredWidthAndState()) and **getMeasuredWidthAndState()** (/reference/android/view/View#getMeasuredWidthAndState()) that provide the actual measured size. |
| **int** | **MEASURED_STATE_MASK** (/reference/android/view/View#MEASURED_STATE_MASK)<br><br>Bits of **getMeasuredWidthAndState()** (/reference/android/view/View#getMeasuredWidthAndState()) and **getMeasuredWidthAndState()** (/reference/android/view/View#getMeasuredWidthAndState()) that provide the additional state bits. |
| **int** | **MEASURED_STATE_TOO_SMALL** (/reference/android/view/View#MEASURED_STATE_TOO_SMALL)<br><br>Bit of **getMeasuredWidthAndState()** (/reference/android/view/View#getMeasuredWidthAndState()) and **getMeasuredWidthAndState()** (/reference/android/view/View#getMeasuredWidthAndState()) that indicates the measured size is smaller that the space the view would like to have. |
| **int** | **NOT_FOCUSABLE** (/reference/android/view/View#NOT_FOCUSABLE)<br><br>This view does not want keystrokes. |
| **int** | **NO_ID** (/reference/android/view/View#NO_ID)<br><br>Used to mark a View that has no ID. |
| **int** | **OVER_SCROLL_ALWAYS** (/reference/android/view/View#OVER_SCROLL_ALWAYS)<br><br>Always allow a user to over-scroll this view, provided it is a view that can scroll. |

| | |
|---|---|
| int | OVER_SCROLL_IF_CONTENT_SCROLLS (/reference/android/view/View#OVER_SCROLL_IF_CONTENT_SCROLLS)<br>Allow a user to over-scroll this view only if the content is large enough to meaningfully scroll, provided it is a view that can scroll. |
| int | OVER_SCROLL_NEVER (/reference/android/view/View#OVER_SCROLL_NEVER)<br>Never allow a user to over-scroll this view. |
| int | SCREEN_STATE_OFF (/reference/android/view/View#SCREEN_STATE_OFF)<br>Indicates that the screen has changed state and is now off. |
| int | SCREEN_STATE_ON (/reference/android/view/View#SCREEN_STATE_ON)<br>Indicates that the screen has changed state and is now on. |
| int | SCROLLBARS_INSIDE_INSET (/reference/android/view/View#SCROLLBARS_INSIDE_INSET)<br>The scrollbar style to display the scrollbars inside the padded area, increasing the padding of the view. |
| int | SCROLLBARS_INSIDE_OVERLAY (/reference/android/view/View#SCROLLBARS_INSIDE_OVERLAY)<br>The scrollbar style to display the scrollbars inside the content area, without increasing the padding. |
| int | SCROLLBARS_OUTSIDE_INSET (/reference/android/view/View#SCROLLBARS_OUTSIDE_INSET)<br>The scrollbar style to display the scrollbars at the edge of the view, increasing the padding of the view. |
| int | SCROLLBARS_OUTSIDE_OVERLAY (/reference/android/view/View#SCROLLBARS_OUTSIDE_OVERLAY)<br>The scrollbar style to display the scrollbars at the edge of the view, without increasing the padding. |
| int | SCROLLBAR_POSITION_DEFAULT (/reference/android/view/View#SCROLLBAR_POSITION_DEFAULT) |

| | |
|---|---|
| | Position the scroll bar at the default position as determined by the system. |
| int | SCROLLBAR_POSITION_LEFT (/reference/android/view/View#SCROLLBAR_POSITION_LEFT)<br>Position the scroll bar along the left edge. |
| int | SCROLLBAR_POSITION_RIGHT (/reference/android/view/View#SCROLLBAR_POSITION_RIGHT)<br>Position the scroll bar along the right edge. |
| int | SCROLL_AXIS_HORIZONTAL (/reference/android/view/View#SCROLL_AXIS_HORIZONTAL)<br>Indicates scrolling along the horizontal axis. |
| int | SCROLL_AXIS_NONE (/reference/android/view/View#SCROLL_AXIS_NONE)<br>Indicates no axis of view scrolling. |
| int | SCROLL_AXIS_VERTICAL (/reference/android/view/View#SCROLL_AXIS_VERTICAL)<br>Indicates scrolling along the vertical axis. |
| int | SCROLL_INDICATOR_BOTTOM (/reference/android/view/View#SCROLL_INDICATOR_BOTTOM)<br>Scroll indicator direction for the bottom edge of the view. |
| int | SCROLL_INDICATOR_END (/reference/android/view/View#SCROLL_INDICATOR_END)<br>Scroll indicator direction for the ending edge of the view. |
| int | SCROLL_INDICATOR_LEFT (/reference/android/view/View#SCROLL_INDICATOR_LEFT)<br>Scroll indicator direction for the left edge of the view. |
| int | SCROLL_INDICATOR_RIGHT (/reference/android/view/View#SCROLL_INDICATOR_RIGHT) |

| | |
|---|---|
| | Scroll indicator direction for the right edge of the view. |
| **int** | **SCROLL_INDICATOR_START** (/reference/android/view/View#SCROLL_INDICATOR_START)<br><br>Scroll indicator direction for the starting edge of the view. |
| **int** | **SCROLL_INDICATOR_TOP** (/reference/android/view/View#SCROLL_INDICATOR_TOP)<br><br>Scroll indicator direction for the top edge of the view. |
| **int** | **SOUND_EFFECTS_ENABLED** (/reference/android/view/View#SOUND_EFFECTS_ENABLED)<br><br>View flag indicating whether this view should have sound effects enabled for events such as clicking and touching. |
| **int** | **STATUS_BAR_HIDDEN** (/reference/android/view/View#STATUS_BAR_HIDDEN)<br><br>*This constant was deprecated in API level 15. Use* **SYSTEM_UI_FLAG_LOW_PROFILE** (/reference/android/view/View#SYSTEM_UI_FLAG_LOW_PROFILE) *instead.* |
| **int** | **STATUS_BAR_VISIBLE** (/reference/android/view/View#STATUS_BAR_VISIBLE)<br><br>*This constant was deprecated in API level 15. Use* **SYSTEM_UI_FLAG_VISIBLE** (/reference/android/view/View#SYSTEM_UI_FLAG_VISIBLE) *instead.* |
| **int** | **SYSTEM_UI_FLAG_FULLSCREEN** (/reference/android/view/View#SYSTEM_UI_FLAG_FULLSCREEN)<br><br>Flag for **setSystemUiVisibility(int)** (/reference/android/view/View#setSystemUiVisibility(int)): View has requested to go into the normal fullscreen mode so that its content can take over the screen while still allowing the user to interact with the application. |
| **int** | **SYSTEM_UI_FLAG_HIDE_NAVIGATION** (/reference/android/view/View#SYSTEM_UI_FLAG_HIDE_NAVIGATION)<br><br>Flag for **setSystemUiVisibility(int)** (/reference/android/view/View#setSystemUiVisibility(int)): View has requested that the system navigation be temporarily hidden. |

| | |
|---|---|
| **int** | **SYSTEM_UI_FLAG_IMMERSIVE** (/reference/android/view/View#SYSTEM_UI_FLAG_IMMERSIVE)<br><br>Flag for **setSystemUiVisibility(int)** (/reference/android/view/View#setSystemUiVisibility(int)): View would like to remain interactive when hiding the navigation bar with **SYSTEM_UI_FLAG_HIDE_NAVIGATION** (/reference/android/view/View#SYSTEM_UI_FLAG_HIDE_NAVIGATION). |
| **int** | **SYSTEM_UI_FLAG_IMMERSIVE_STICKY** (/reference/android/view/View#SYSTEM_UI_FLAG_IMMERSIVE_STICKY)<br><br>Flag for **setSystemUiVisibility(int)** (/reference/android/view/View#setSystemUiVisibility(int)): View would like to remain interactive when hiding the status bar with **SYSTEM_UI_FLAG_FULLSCREEN** (/reference/android/view/View#SYSTEM_UI_FLAG_FULLSCREEN) and/or hiding the navigation bar with **SYSTEM_UI_FLAG_HIDE_NAVIGATION** (/reference/android/view/View#SYSTEM_UI_FLAG_HIDE_NAVIGATION). |
| **int** | **SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN** (/reference/android/view/View#SYSTEM_UI_FLAG_LAYOUT_FULLSCREEN)<br><br>Flag for **setSystemUiVisibility(int)** (/reference/android/view/View#setSystemUiVisibility(int)): View would like its window to be laid out as if it has requested **SYSTEM_UI_FLAG_FULLSCREEN** (/reference/android/view/View#SYSTEM_UI_FLAG_FULLSCREEN), even if it currently hasn't. |
| **int** | **SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION** (/reference/android/view/View#SYSTEM_UI_FLAG_LAYOUT_HIDE_NAVIGATION)<br><br>Flag for **setSystemUiVisibility(int)** (/reference/android/view/View#setSystemUiVisibility(int)): View would like its window to be laid out as if it has requested **SYSTEM_UI_FLAG_HIDE_NAVIGATION** (/reference/android/view/View#SYSTEM_UI_FLAG_HIDE_NAVIGATION), even if it currently hasn't. |
| **int** | **SYSTEM_UI_FLAG_LAYOUT_STABLE** (/reference/android/view/View#SYSTEM_UI_FLAG_LAYOUT_STABLE)<br><br>Flag for **setSystemUiVisibility(int)** (/reference/android/view/View#setSystemUiVisibility(int)): When using other layout flags, we would like a stable view of the content insets given to **fitSystemWindows(android.graphics.Rect)** (/reference/android/view/View#fitSystemWindows(android.graphics.Rect)). |
| **int** | **SYSTEM_UI_FLAG_LIGHT_NAVIGATION_BAR** (/reference/android/view/View#SYSTEM_UI_FLAG_LIGHT_NAVIGATION_BAR)<br><br>Flag for **setSystemUiVisibility(int)** (/reference/android/view/View#setSystemUiVisibility(int)): Requests the navigation bar to draw in a mode that is compatible with light navigation bar backgrounds. |
| **int** | **SYSTEM_UI_FLAG_LIGHT_STATUS_BAR** (/reference/android/view/View#SYSTEM_UI_FLAG_LIGHT_STATUS_BAR) |

| | |
|---|---|
| | Flag for `setSystemUiVisibility(int)` (/reference/android/view/View#setSystemUiVisibility(int)): Requests the status bar to draw in a mode that is compatible with light status bar backgrounds. |
| `int` | `SYSTEM_UI_FLAG_LOW_PROFILE` (/reference/android/view/View#SYSTEM_UI_FLAG_LOW_PROFILE)<br><br>Flag for `setSystemUiVisibility(int)` (/reference/android/view/View#setSystemUiVisibility(int)): View has requested the system UI to enter an unobtrusive "low profile" mode. |
| `int` | `SYSTEM_UI_FLAG_VISIBLE` (/reference/android/view/View#SYSTEM_UI_FLAG_VISIBLE)<br><br>Special constant for `setSystemUiVisibility(int)` (/reference/android/view/View#setSystemUiVisibility(int)): View has requested the system UI (status bar) to be visible (the default). |
| `int` | `SYSTEM_UI_LAYOUT_FLAGS` (/reference/android/view/View#SYSTEM_UI_LAYOUT_FLAGS)<br><br>Flags that can impact the layout in relation to system UI. |
| `int` | `TEXT_ALIGNMENT_CENTER` (/reference/android/view/View#TEXT_ALIGNMENT_CENTER)<br><br>Center the paragraph, e.g. ALIGN_CENTER. |
| `int` | `TEXT_ALIGNMENT_GRAVITY` (/reference/android/view/View#TEXT_ALIGNMENT_GRAVITY)<br><br>Default for the root view. |
| `int` | `TEXT_ALIGNMENT_INHERIT` (/reference/android/view/View#TEXT_ALIGNMENT_INHERIT)<br><br>Default text alignment. |
| `int` | `TEXT_ALIGNMENT_TEXT_END` (/reference/android/view/View#TEXT_ALIGNMENT_TEXT_END)<br><br>Align to the end of the paragraph, e.g. ALIGN_OPPOSITE. |
| `int` | `TEXT_ALIGNMENT_TEXT_START` (/reference/android/view/View#TEXT_ALIGNMENT_TEXT_START) |

| | |
|---|---|
| | Align to the start of the paragraph, e.g. ALIGN_NORMAL. |
| **int** | **TEXT_ALIGNMENT_VIEW_END** (/reference/android/view/View#TEXT_ALIGNMENT_VIEW_END)<br>Align to the end of the view, which is ALIGN_RIGHT if the view's resolved layoutDirection is LTR, and ALIGN_LEFT otherwise. |
| **int** | **TEXT_ALIGNMENT_VIEW_START** (/reference/android/view/View#TEXT_ALIGNMENT_VIEW_START)<br>Align to the start of the view, which is ALIGN_LEFT if the view's resolved layoutDirection is LTR, and ALIGN_RIGHT otherwise. |
| **int** | **TEXT_DIRECTION_ANY_RTL** (/reference/android/view/View#TEXT_DIRECTION_ANY_RTL)<br>Text direction is using "any-RTL" algorithm. |
| **int** | **TEXT_DIRECTION_FIRST_STRONG** (/reference/android/view/View#TEXT_DIRECTION_FIRST_STRONG)<br>Text direction is using "first strong algorithm". |
| **int** | **TEXT_DIRECTION_FIRST_STRONG_LTR** (/reference/android/view/View#TEXT_DIRECTION_FIRST_STRONG_LTR)<br>Text direction is using "first strong algorithm". |
| **int** | **TEXT_DIRECTION_FIRST_STRONG_RTL** (/reference/android/view/View#TEXT_DIRECTION_FIRST_STRONG_RTL)<br>Text direction is using "first strong algorithm". |
| **int** | **TEXT_DIRECTION_INHERIT** (/reference/android/view/View#TEXT_DIRECTION_INHERIT)<br>Text direction is inherited through **ViewGroup** (/reference/android/view/ViewGroup) |
| **int** | **TEXT_DIRECTION_LOCALE** (/reference/android/view/View#TEXT_DIRECTION_LOCALE)<br>Text direction is coming from the system Locale. |

| | |
|---|---|
| `int` | **TEXT_DIRECTION_LTR** (/reference/android/view/View#TEXT_DIRECTION_LTR)<br><br>Text direction is forced to LTR. |
| `int` | **TEXT_DIRECTION_RTL** (/reference/android/view/View#TEXT_DIRECTION_RTL)<br><br>Text direction is forced to RTL. |
| **String** (/reference/java/lang/String) | **VIEW_LOG_TAG** (/reference/android/view/View#VIEW_LOG_TAG)<br><br>The logging tag used by this class with android.util.Log. |
| `int` | **VISIBLE** (/reference/android/view/View#VISIBLE)<br><br>This view is visible. |

## Inherited fields

From class **android.view.View** (/reference/android/view/View)

| | |
|---|---|
| `public static final` **Property** (/reference/android/util/Property)`<`**View** (/reference/android/view/View)`,` **Float** (/reference/java/lang/Float)`>` | **ALPHA** (/reference/android/view/View#ALPHA)<br><br>A Property wrapper around the **alpha** functionality handled by the **View#setAlpha(float)** (/reference/android/view/View#setAlpha(float)) and **View#getAlpha()** (/reference/android/view/View#getAlpha()) methods. |
| `protected static final int[]` | **EMPTY_STATE_SET** (/reference/android/view/View#EMPTY_STATE_SET)<br><br>Indicates the view has no states set. |
| `protected static final int[]` | **ENABLED_FOCUSED_SELECTED_STATE_SET** (/reference/android/view/View#ENABLED_FOCUSED_SELECTED_STATE_SET) |

| | |
|---|---|
| | Indicates the view is enabled, focused and selected. |
| `protected static final int[]` | **ENABLED_FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET** (/reference/android/view/View#ENABLED_FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET)<br><br>Indicates the view is enabled, focused, selected and its window has the focus. |
| `protected static final int[]` | **ENABLED_FOCUSED_STATE_SET** (/reference/android/view/View#ENABLED_FOCUSED_STATE_SET)<br><br>Indicates the view is enabled and has the focus. |
| `protected static final int[]` | **ENABLED_FOCUSED_WINDOW_FOCUSED_STATE_SET** (/reference/android/view/View#ENABLED_FOCUSED_WINDOW_FOCUSED_STATE_SET)<br><br>Indicates the view is enabled, focused and its window has the focus. |
| `protected static final int[]` | **ENABLED_SELECTED_STATE_SET** (/reference/android/view/View#ENABLED_SELECTED_STATE_SET)<br><br>Indicates the view is enabled and selected. |
| `protected static final int[]` | **ENABLED_SELECTED_WINDOW_FOCUSED_STATE_SET** (/reference/android/view/View#ENABLED_SELECTED_WINDOW_FOCUSED_STATE_SET)<br><br>Indicates the view is enabled, selected and its window has the focus. |
| `protected static final int[]` | **ENABLED_STATE_SET** (/reference/android/view/View#ENABLED_STATE_SET)<br><br>Indicates the view is enabled. |

| | |
|---|---|
| `protected static final int[]` | **ENABLED_WINDOW_FOCUSED_STATE_SET** (/reference/android/view/View#ENABLED_WINDOW_FOCUSED_STATE_SET)<br><br>Indicates the view is enabled and that its window has focus. |
| `protected static final int[]` | **FOCUSED_SELECTED_STATE_SET** (/reference/android/view/View#FOCUSED_SELECTED_STATE_SET)<br><br>Indicates the view is focused and selected. |
| `protected static final int[]` | **FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET** (/reference/android/view/View#FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET)<br><br>Indicates the view is focused, selected and its window has the focus. |
| `protected static final int[]` | **FOCUSED_STATE_SET** (/reference/android/view/View#FOCUSED_STATE_SET)<br><br>Indicates the view is focused. |
| `protected static final int[]` | **FOCUSED_WINDOW_FOCUSED_STATE_SET** (/reference/android/view/View#FOCUSED_WINDOW_FOCUSED_STATE_SET)<br><br>Indicates the view has the focus and that its window has the focus. |
| `protected static final int[]` | **PRESSED_ENABLED_FOCUSED_SELECTED_STATE_SET** (/reference/android/view/View#PRESSED_ENABLED_FOCUSED_SELECTED_STATE_SET)<br><br>Indicates the view is pressed, enabled, focused and selected. |
| `protected static final int[]` | **PRESSED_ENABLED_FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET** (/reference/android/view/View#PRESSED_ENABLED_FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET) |

| | |
|---|---|
| | Indicates the view is pressed, enabled, focused, selected and its window has the focus. |
| `protected static final int[]` | **PRESSED_ENABLED_FOCUSED_STATE_SET** (/reference/android/view/View#PRESSED_ENABLED_FOCUSED_STATE_SET)<br><br>Indicates the view is pressed, enabled and focused. |
| `protected static final int[]` | **PRESSED_ENABLED_FOCUSED_WINDOW_FOCUSED_STATE_SET** (/reference/android/view/View#PRESSED_ENABLED_FOCUSED_WINDOW_FOCUSED_STATE_SET)<br><br>Indicates the view is pressed, enabled, focused and its window has the focus. |
| `protected static final int[]` | **PRESSED_ENABLED_SELECTED_STATE_SET** (/reference/android/view/View#PRESSED_ENABLED_SELECTED_STATE_SET)<br><br>Indicates the view is pressed, enabled and selected. |
| `protected static final int[]` | **PRESSED_ENABLED_SELECTED_WINDOW_FOCUSED_STATE_SET** (/reference/android/view/View#PRESSED_ENABLED_SELECTED_WINDOW_FOCUSED_STATE_SET)<br><br>Indicates the view is pressed, enabled, selected and its window has the focus. |
| `protected static final int[]` | **PRESSED_ENABLED_STATE_SET** (/reference/android/view/View#PRESSED_ENABLED_STATE_SET)<br><br>Indicates the view is pressed and enabled. |
| `protected static final int[]` | **PRESSED_ENABLED_WINDOW_FOCUSED_STATE_SET** |

| | |
|---|---|
| | (/reference/android/view/View#PRESSED_ENABLED_WINDOW_FOCUSED_STATE_SET)<br><br>Indicates the view is pressed, enabled and its window has the focus. |
| `protected static final int[]` | **PRESSED_FOCUSED_SELECTED_STATE_SET**<br>(/reference/android/view/View#PRESSED_FOCUSED_SELECTED_STATE_SET)<br><br>Indicates the view is pressed, focused and selected. |
| `protected static final int[]` | **PRESSED_FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET**<br>(/reference/android/view/View#PRESSED_FOCUSED_SELECTED_WINDOW_FOCUSED_STATE_SET)<br><br>Indicates the view is pressed, focused, selected and its window has the focus. |
| `protected static final int[]` | **PRESSED_FOCUSED_STATE_SET**<br>(/reference/android/view/View#PRESSED_FOCUSED_STATE_SET)<br><br>Indicates the view is pressed and focused. |
| `protected static final int[]` | **PRESSED_FOCUSED_WINDOW_FOCUSED_STATE_SET**<br>(/reference/android/view/View#PRESSED_FOCUSED_WINDOW_FOCUSED_STATE_SET)<br><br>Indicates the view is pressed, focused and its window has the focus. |
| `protected static final int[]` | **PRESSED_SELECTED_STATE_SET**<br>(/reference/android/view/View#PRESSED_SELECTED_STATE_SET)<br><br>Indicates the view is pressed and selected. |
| `protected static final int[]` | **PRESSED_SELECTED_WINDOW_FOCUSED_STATE_SET** |

| | |
|---|---|
| | (/reference/android/view/View#PRESSED_SELECTED_WINDOW_FOCUSED_STATE_SET)<br><br>Indicates the view is pressed, selected and its window has the focus. |
| `protected static final int[]` | **PRESSED_STATE_SET** (/reference/android/view/View#PRESSED_STATE_SET)<br><br>Indicates the view is pressed. |
| `protected static final int[]` | **PRESSED_WINDOW_FOCUSED_STATE_SET** (/reference/android/view/View#PRESSED_WINDOW_FOCUSED_STATE_SET)<br><br>Indicates the view is pressed and its window has the focus. |
| `public static final` **Property** (/reference/android/util/Property)`<`**View** (/reference/android/view/View)`,` **Float** (/reference/java/lang/Float)`>` | **ROTATION** (/reference/android/view/View#ROTATION)<br><br>A Property wrapper around the **rotation** functionality handled by the **View#setRotation(float)** (/reference/android/view/View#setRotation(float)) and **View#getRotation()** (/reference/android/view/View#getRotation()) methods. |
| `public static final` **Property** (/reference/android/util/Property)`<`**View** (/reference/android/view/View)`,` **Float** (/reference/java/lang/Float)`>` | **ROTATION_X** (/reference/android/view/View#ROTATION_X)<br><br>A Property wrapper around the **rotationX** functionality handled by the **View#setRotationX(float)** (/reference/android/view/View#setRotationX(float)) and **View#getRotationX()** (/reference/android/view/View#getRotationX()) methods. |
| `public static final` **Property** (/reference/android/util/Property)`<`**View** (/reference/android/view/View)`,` **Float** (/reference/java/lang/Float)`>` | **ROTATION_Y** (/reference/android/view/View#ROTATION_Y)<br><br>A Property wrapper around the **rotationY** functionality handled by the **View#setRotationY(float)** |

| | |
|---|---|
| | (/reference/android/view/View#setRotationY(float)) and **View#getRotationY()** (/reference/android/view/View#getRotationY()) methods. |
| **public static final Property** (/reference/android/util/Property)**<View** (/reference/android/view/View)**, Float** (/reference/java/lang/Float)**>** | **SCALE_X** (/reference/android/view/View#SCALE_X)<br><br>A Property wrapper around the **scaleX** functionality handled by the **View#setScaleX(float)** (/reference/android/view/View#setScaleX(float)) and **View#getScaleX()** (/reference/android/view/View#getScaleX()) methods. |
| **public static final Property** (/reference/android/util/Property)**<View** (/reference/android/view/View)**, Float** (/reference/java/lang/Float)**>** | **SCALE_Y** (/reference/android/view/View#SCALE_Y)<br><br>A Property wrapper around the **scaleY** functionality handled by the **View#setScaleY(float)** (/reference/android/view/View#setScaleY(float)) and **View#getScaleY()** (/reference/android/view/View#getScaleY()) methods. |
| **protected static final int[]** | **SELECTED_STATE_SET** (/reference/android/view/View#SELECTED_STATE_SET)<br><br>Indicates the view is selected. |
| **protected static final int[]** | **SELECTED_WINDOW_FOCUSED_STATE_SET** (/reference/android/view/View#SELECTED_WINDOW_FOCUSED_STATE_SET)<br><br>Indicates the view is selected and that its window has the focus. |
| **public static final Property** (/reference/android/util/Property)**<View** (/reference/android/view/View)**, Float** (/reference/java/lang/Float)**>** | **TRANSLATION_X** (/reference/android/view/View#TRANSLATION_X)<br><br>A Property wrapper around the **translationX** functionality handled by the **View#setTranslationX(float)** (/reference/android/view/View#setTranslationX(float)) and |

| | |
|---|---|
| | **View#getTranslationX()** (/reference/android/view/View#getTranslationX()) methods. |
| **public static final Property** (/reference/android/util/Property)**<View** (/reference/android/view/View)**, Float** (/reference/java/lang/Float)**>** | **TRANSLATION_Y** (/reference/android/view/View#TRANSLATION_Y)<br><br>A Property wrapper around the **translationY** functionality handled by the **View#setTranslationY(float)** (/reference/android/view/View#setTranslationY(float)) and **View#getTranslationY()** (/reference/android/view/View#getTranslationY()) methods. |
| **public static final Property** (/reference/android/util/Property)**<View** (/reference/android/view/View)**, Float** (/reference/java/lang/Float)**>** | **TRANSLATION_Z** (/reference/android/view/View#TRANSLATION_Z)<br><br>A Property wrapper around the **translationZ** functionality handled by the **View#setTranslationZ(float)** (/reference/android/view/View#setTranslationZ(float)) and **View#getTranslationZ()** (/reference/android/view/View#getTranslationZ()) methods. |
| **protected static final int[]** | **WINDOW_FOCUSED_STATE_SET** (/reference/android/view/View#WINDOW_FOCUSED_STATE_SET)<br><br>Indicates the view's window has focus. |
| **public static final Property** (/reference/android/util/Property)**<View** (/reference/android/view/View)**, Float** (/reference/java/lang/Float)**>** | **X** (/reference/android/view/View#X)<br><br>A Property wrapper around the **x** functionality handled by the **View#setX(float)** (/reference/android/view/View#setX(float)) and **View#getX()** (/reference/android/view/View#getX()) methods. |
| **public static final Property** (/reference/android/util/Property)**<View** (/reference/android/view/View)**, Float** (/reference/java/lang/Float)**>** | **Y** (/reference/android/view/View#Y)<br><br>A Property wrapper around the **y** functionality handled by the |

View#setY(float) (/reference/android/view/View#setY(float)) and View#getY() (/reference/android/view/View#getY()) methods.

---

`public static final` Property (/reference/android/util/Property)`<`View (/reference/android/view/View)`,` Z (/reference/android/view/View#Z) Float (/reference/java/lang/Float)`>`

A Property wrapper around the **z** functionality handled by the View#setZ(float) (/reference/android/view/View#setZ(float)) and View#getZ() (/reference/android/view/View#getZ()) methods.

## Public constructors

---

TextView (/reference/android/widget/TextView#TextView(android.content.Context))`(`Context (/reference/android/content/Context) `context)`

---

TextView (/reference/android/widget/TextView#TextView(android.content.Context,%20android.util.AttributeSet))`(`Context (/reference/android/content/Context) `context,` AttributeSet (/reference/android/util/AttributeSet) `attrs)`

---

TextView (/reference/android/widget/TextView#TextView(android.content.Context,%20android.util.AttributeSet,%20int))`(`Context (/reference/android/content/Context) `context,` AttributeSet (/reference/android/util/AttributeSet) `attrs, int defStyleAttr)`

---

TextView (/reference/android/widget/TextView#TextView(android.content.Context,%20android.util.AttributeSet,%20int,%20int))`(`Context (/reference/android/content/Context) `context,` AttributeSet (/reference/android/util/AttributeSet) `attrs, int defStyleAttr, int defStyleRes)`

## Public methods

---

`void` addExtraDataToAccessibilityNodeInfo (/reference/android/widget/TextView#addExtraDataToAccessibilityNodeInfo(android.view.accessibility.AccessibilityNodeInfo,%20java.lang.String,%20android.os.Bundle))
`(`AccessibilityNodeInfo (/reference/android/view/accessibility/AccessibilityNodeInfo) `info,` String (/reference/java/lang/String) `extraDataKey,` Bundle (/reference/android/os/Bundle) `arguments)`

| | |
|---|---|
| | Adds extra data to an **AccessibilityNodeInfo** (/reference/android/view/accessibility/AccessibilityNodeInfo) based on an explicit request for the additional data. |
| **void** | **addTextChangedListener** (/reference/android/widget/TextView#addTextChangedListener(android.text.TextWatcher))**(TextWatcher** (/reference/android/text/TextWatcher) **watcher)**<br><br>Adds a TextWatcher to the list of those whose methods are called whenever this TextView's text changes. |
| **final void** | **append** (/reference/android/widget/TextView#append(java.lang.CharSequence))**(CharSequence** (/reference/java/lang/CharSequence) **text)**<br><br>Convenience method to append the specified text to the TextView's display buffer, upgrading it to **TextView.BufferType.EDITABLE** (/reference/android/widget/TextView.BufferType#EDITABLE) if it was not already editable. |
| **void** | **append** (/reference/android/widget/TextView#append(java.lang.CharSequence,%20int,%20int))**(CharSequence** (/reference/java/lang/CharSequence) **text, int start, int end)**<br><br>Convenience method to append the specified text slice to the TextView's display buffer, upgrading it to **TextView.BufferType.EDITABLE** (/reference/android/widget/TextView.BufferType#EDITABLE) if it was not already editable. |
| **void** | **autofill** (/reference/android/widget/TextView#autofill(android.view.autofill.AutofillValue))**(AutofillValue** (/reference/android/view/autofill/AutofillValue) **value)**<br><br>Automatically fills the content of this view with the **value**. |
| **void** | **beginBatchEdit** (/reference/android/widget/TextView#beginBatchEdit())**()** |
| **boolean** | **bringPointIntoView** (/reference/android/widget/TextView#bringPointIntoView(int))**(int offset)**<br><br>Move the point, specified by the offset, into the view if it is needed. |
| **void** | **cancelLongPress** (/reference/android/widget/TextView#cancelLongPress())**()**<br><br>Cancels a pending long press. |

| | |
|---|---|
| **void** | **clearComposingText** (/reference/android/widget/TextView#clearComposingText())() <br><br> Use **BaseInputConnection#removeComposingSpans** (/reference/android/view/inputmethod/BaseInputConnection#removeComposingSpans(android.text.Spannable)) to remove any IME composing state from this text view. |
| **void** | **computeScroll** (/reference/android/widget/TextView#computeScroll())() <br><br> Called by a parent to request that a child update its values for mScrollX and mScrollY if necessary. |
| **void** | **debug** (/reference/android/widget/TextView#debug(int))(**int depth**) |
| **boolean** | **didTouchFocusSelect** (/reference/android/widget/TextView#didTouchFocusSelect())() <br><br> Returns true, only while processing a touch gesture, if the initial touch down event caused focus to move to the text view and as a result its selection changed. |
| **void** | **drawableHotspotChanged** (/reference/android/widget/TextView#drawableHotspotChanged(float,%20float))(**float x, float y**) <br><br> This function is called whenever the view hotspot changes and needs to be propagated to drawables or child views managed by the view. |
| **void** | **endBatchEdit** (/reference/android/widget/TextView#endBatchEdit())() |
| **boolean** | **extractText** (/reference/android/widget/TextView#extractText(android.view.inputmethod.ExtractedTextRequest,%20android.view.inputmethod.ExtractedText))(**ExtractedTextRequest** (/reference/android/view/inputmethod/ExtractedTextRequest) **request, ExtractedText** (/reference/android/view/inputmethod/ExtractedText) **outText**) <br><br> If this TextView contains editable content, extract a portion of it based on the information in ***request*** in to ***outText***. |
| **void** | **findViewsWithText** (/reference/android/widget/TextView#findViewsWithText(java.util.ArrayList<android.view.View>,%20java.lang.CharSequence,%20int))(**ArrayList** (/reference/java/util/ArrayList)**<View** (/reference/android/view/View)**> outViews, CharSequence** (/reference/java/lang/CharSequence) **searched, int flags**) <br><br> Finds the Views that contain given text. |

| | |
|---|---|
| CharSequence (/reference/java/lang/CharSequence) | **getAccessibilityClassName** (/reference/android/widget/TextView#getAccessibilityClassName())() Return the class name of this object to be used for accessibility purposes. |
| final int | **getAutoLinkMask** (/reference/android/widget/TextView#getAutoLinkMask())() Gets the autolink mask of the text. |
| int | **getAutoSizeMaxTextSize** (/reference/android/widget/TextView#getAutoSizeMaxTextSize())() |
| int | **getAutoSizeMinTextSize** (/reference/android/widget/TextView#getAutoSizeMinTextSize())() |
| int | **getAutoSizeStepGranularity** (/reference/android/widget/TextView#getAutoSizeStepGranularity())() |
| int[] | **getAutoSizeTextAvailableSizes** (/reference/android/widget/TextView#getAutoSizeTextAvailableSizes())() |
| int | **getAutoSizeTextType** (/reference/android/widget/TextView#getAutoSizeTextType())() Returns the type of auto-size set for this widget. |
| int | **getAutofillType** (/reference/android/widget/TextView#getAutofillType())() Describes the autofill type of this view, so an **AutofillService** (/reference/android/service/autofill/AutofillService) can create the proper **AutofillValue** (/reference/android/view/autofill/AutofillValue) when autofilling the view. |
| AutofillValue (/reference/android/view/autofill/AutofillValue) | **getAutofillValue** (/reference/android/widget/TextView#getAutofillValue())() Gets the **TextView** (/reference/android/widget/TextView)'s current text for AutoFill. |
| int | **getBaseline** (/reference/android/widget/TextView#getBaseline())() Return the offset of the widget's text baseline from the widget's top boundary. |

| | |
|---|---|
| int | **getBreakStrategy** (/reference/android/widget/TextView#getBreakStrategy())() <br><br> Gets the current strategy for breaking paragraphs into lines. |
| int | **getCompoundDrawablePadding** (/reference/android/widget/TextView#getCompoundDrawablePadding())() <br><br> Returns the padding between the compound drawables and the text. |
| BlendMode (/reference/android/graphics/BlendMode) | **getCompoundDrawableTintBlendMode** (/reference/android/widget/TextView#getCompoundDrawableTintBlendMode())() <br> Returns the blending mode used to apply the tint to the compound drawables, if specified. |
| ColorStateList (/reference/android/content/res/ColorStateList) | **getCompoundDrawableTintList** (/reference/android/widget/TextView#getCompoundDrawableTintList())() |
| PorterDuff.Mode (/reference/android/graphics/PorterDuff.Mode) | **getCompoundDrawableTintMode** (/reference/android/widget/TextView#getCompoundDrawableTintMode())() <br> Returns the blending mode used to apply the tint to the compound drawables, if specified. |
| Drawable[] (/reference/android/graphics/drawable/Drawable) | **getCompoundDrawables** (/reference/android/widget/TextView#getCompoundDrawables())() <br> Returns drawables for the left, top, right, and bottom borders. |
| Drawable[] (/reference/android/graphics/drawable/Drawable) | **getCompoundDrawablesRelative** (/reference/android/widget/TextView#getCompoundDrawablesRelative())() <br> Returns drawables for the start, top, end, and bottom borders. |
| int | **getCompoundPaddingBottom** (/reference/android/widget/TextView#getCompoundPaddingBottom())() <br><br> Returns the bottom padding of the view, plus space for the bottom Drawable if any. |
| int | **getCompoundPaddingEnd** (/reference/android/widget/TextView#getCompoundPaddingEnd())() |

| | |
|---|---|
| | Returns the end padding of the view, plus space for the end Drawable if any. |
| `int` | **getCompoundPaddingLeft** (/reference/android/widget/TextView#getCompoundPaddingLeft())() Returns the left padding of the view, plus space for the left Drawable if any. |
| `int` | **getCompoundPaddingRight** (/reference/android/widget/TextView#getCompoundPaddingRight())() Returns the right padding of the view, plus space for the right Drawable if any. |
| `int` | **getCompoundPaddingStart** (/reference/android/widget/TextView#getCompoundPaddingStart())() Returns the start padding of the view, plus space for the start Drawable if any. |
| `int` | **getCompoundPaddingTop** (/reference/android/widget/TextView#getCompoundPaddingTop())() Returns the top padding of the view, plus space for the top Drawable if any. |
| `final int` | **getCurrentHintTextColor** (/reference/android/widget/TextView#getCurrentHintTextColor())() Return the current color selected to paint the hint text. |
| `final int` | **getCurrentTextColor** (/reference/android/widget/TextView#getCurrentTextColor())() Return the current color selected for normal text. |
| ActionMode.Callback (/reference/android/view/ActionMode.Callback) | **getCustomInsertionActionModeCallback** (/reference/android/widget/TextView#getCustomInsertionActionModeCallback())() Retrieves the value set in **setCustomInsertionActionModeCallback(ActionMode.Callback)** (/reference/android/widget/TextView#setCustomInsertionActionModeCallback(android.view.ActionMode.Callback)). |
| ActionMode.Callback (/reference/android/view/ActionMode.Callback) | **getCustomSelectionActionModeCallback** (/reference/android/widget/TextView#getCustomSelectionActionModeCallback())() Retrieves the value set in **setCustomSelectionActionModeCallback(ActionMode.Callback)** |

(/reference/android/widget/TextView#setCustomSelectionActionModeCallback(android.view.ActionMode.Callback)).

| | |
|---|---|
| Editable (/reference/android/text/Editable) | getEditableText (/reference/android/widget/TextView#getEditableText())() Return the text that TextView is displaying as an Editable object. |
| TextUtils.TruncateAt (/reference/android/text/TextUtils.TruncateAt) | getEllipsize (/reference/android/widget/TextView#getEllipsize())() Returns where, if anywhere, words that are longer than the view is wide should be ellipsized. |
| CharSequence (/reference/java/lang/CharSequence) | getError (/reference/android/widget/TextView#getError())() Returns the error message that was set to be displayed with setError(CharSequence) (/reference/android/widget/TextView#setError(java.lang.CharSequence)), or null if no error was set or if it the error was cleared by the widget after user input. |
| int | getExtendedPaddingBottom (/reference/android/widget/TextView#getExtendedPaddingBottom())() Returns the extended bottom padding of the view, including both the bottom Drawable if any and any extra space to keep more than maxLines of text from showing. |
| int | getExtendedPaddingTop (/reference/android/widget/TextView#getExtendedPaddingTop())() Returns the extended top padding of the view, including both the top Drawable if any and any extra space to keep more than maxLines of text from showing. |
| InputFilter[] (/reference/android/text/InputFilter) | getFilters (/reference/android/widget/TextView#getFilters())() Returns the current list of input filters. |
| int | getFirstBaselineToTopHeight (/reference/android/widget/TextView#getFirstBaselineToTopHeight())() Returns the distance between the first text baseline and the top of this TextView. |
| void | getFocusedRect (/reference/android/widget/TextView#getFocusedRect(android.graphics.Rect))(Rect (/reference/android/graphics/Rect) r) |

| | |
|---|---|
| | When a view has focus and the user navigates away from it, the next view is searched for starting from the rectangle filled in by this method. |
| String (/reference/java/lang/String) | getFontFeatureSettings (/reference/android/widget/TextView#getFontFeatureSettings())() Returns the font feature settings. |
| String (/reference/java/lang/String) | getFontVariationSettings (/reference/android/widget/TextView#getFontVariationSettings())() Returns the font variation settings. |
| boolean | getFreezesText (/reference/android/widget/TextView#getFreezesText())() Return whether this text view is including its entire text contents in frozen icicles. |
| int | getGravity (/reference/android/widget/TextView#getGravity())() Returns the horizontal and vertical alignment of this TextView. |
| int | getHighlightColor (/reference/android/widget/TextView#getHighlightColor())() |
| CharSequence (/reference/java/lang/CharSequence) | getHint (/reference/android/widget/TextView#getHint())() Returns the hint that is displayed when the text of the TextView is empty. |
| final ColorStateList (/reference/android/content/res/ColorStateList) | getHintTextColors (/reference/android/widget/TextView#getHintTextColors())() |
| int | getHyphenationFrequency (/reference/android/widget/TextView#getHyphenationFrequency())() Gets the current frequency of automatic hyphenation to be used when determining word breaks. |
| int | getImeActionId (/reference/android/widget/TextView#getImeActionId())() |

| | |
|---|---|
| | Get the IME action ID previous set with **setImeActionLabel(CharSequence, int)** (/reference/android/widget/TextView#setImeActionLabel(java.lang.CharSequence,%20int)). |
| **CharSequence** (/reference/java/lang/CharSequence) | **getImeActionLabel** (/reference/android/widget/TextView#getImeActionLabel())() Get the IME action label previous set with **setImeActionLabel(CharSequence, int)** (/reference/android/widget/TextView#setImeActionLabel(java.lang.CharSequence,%20int)). |
| **LocaleList** (/reference/android/os/LocaleList) | **getImeHintLocales** (/reference/android/widget/TextView#getImeHintLocales())() |
| **int** | **getImeOptions** (/reference/android/widget/TextView#getImeOptions())() Get the type of the Input Method Editor (IME). |
| **boolean** | **getIncludeFontPadding** (/reference/android/widget/TextView#getIncludeFontPadding())() Gets whether the TextView includes extra top and bottom padding to make room for accents that go above the normal ascent and descent. |
| **Bundle** (/reference/android/os/Bundle) | **getInputExtras** (/reference/android/widget/TextView#getInputExtras(boolean))(**boolean create**) Retrieve the input extras currently associated with the text view, which can be viewed as well as modified. |
| **int** | **getInputType** (/reference/android/widget/TextView#getInputType())() Get the type of the editable content. |
| **int** | **getJustificationMode** (/reference/android/widget/TextView#getJustificationMode())() |
| **final KeyListener** (/reference/android/text/method/KeyListener) | **getKeyListener** (/reference/android/widget/TextView#getKeyListener())() Gets the current **KeyListener** (/reference/android/text/method/KeyListener) for the TextView. |

| | |
|---|---|
| `int` | **`getLastBaselineToBottomHeight`** (/reference/android/widget/TextView#getLastBaselineToBottomHeight())**`()`**<br><br>Returns the distance between the last text baseline and the bottom of this TextView. |
| `final` **`Layout`** (/reference/android/text/Layout) | **`getLayout`** (/reference/android/widget/TextView#getLayout())**`()`**<br><br>Gets the **`Layout`** (/reference/android/text/Layout) that is currently being used to display the text. |
| `float` | **`getLetterSpacing`** (/reference/android/widget/TextView#getLetterSpacing())**`()`**<br><br>Gets the text letter-space value, which determines the spacing between characters. |
| `int` | **`getLineBounds`** (/reference/android/widget/TextView#getLineBounds(int,%20android.graphics.Rect))**`(int line,`** **`Rect`** (/reference/android/graphics/Rect) **`bounds)`**<br><br>Return the baseline for the specified line (0...getLineCount() - 1) If bounds is not null, return the top, left, right, bottom extents of the specified line in it. |
| `int` | **`getLineCount`** (/reference/android/widget/TextView#getLineCount())**`()`**<br><br>Return the number of lines of text, or 0 if the internal Layout has not been built. |
| `int` | **`getLineHeight`** (/reference/android/widget/TextView#getLineHeight())**`()`**<br><br>Gets the vertical distance between lines of text, in pixels. |
| `float` | **`getLineSpacingExtra`** (/reference/android/widget/TextView#getLineSpacingExtra())**`()`**<br><br>Gets the line spacing extra space |
| `float` | **`getLineSpacingMultiplier`** (/reference/android/widget/TextView#getLineSpacingMultiplier())**`()`**<br><br>Gets the line spacing multiplier |
| `final` **`ColorStateList`** | **`getLinkTextColors`** (/reference/android/widget/TextView#getLinkTextColors())**`()`** |

(/reference/android/content/res/ColorStateList)

| | |
|---|---|
| `final boolean` | **`getLinksClickable`** (/reference/android/widget/TextView#getLinksClickable())**( )**<br><br>Returns whether the movement method will automatically be set to **`LinkMovementMethod`** (/reference/android/text/method/LinkMovementMethod) if **`setAutoLinkMask(int)`** (/reference/android/widget/TextView#setAutoLinkMask(int)) has been set to nonzero and links are detected in **`setText(char[], int, int)`** (/reference/android/widget/TextView#setText(char[],%20int,%20int)). |
| `int` | **`getMarqueeRepeatLimit`** (/reference/android/widget/TextView#getMarqueeRepeatLimit())**( )**<br><br>Gets the number of times the marquee animation is repeated. |
| `int` | **`getMaxEms`** (/reference/android/widget/TextView#getMaxEms())**( )**<br><br>Returns the maximum width of TextView in terms of ems or -1 if the maximum width was set using **`setMaxWidth(int)`** (/reference/android/widget/TextView#setMaxWidth(int)) or **`setWidth(int)`** (/reference/android/widget/TextView#setWidth(int)). |
| `int` | **`getMaxHeight`** (/reference/android/widget/TextView#getMaxHeight())**( )**<br><br>Returns the maximum height of TextView in terms of pixels or -1 if the maximum height was set using **`setMaxLines(int)`** (/reference/android/widget/TextView#setMaxLines(int)) or **`setLines(int)`** (/reference/android/widget/TextView#setLines(int)). |
| `int` | **`getMaxLines`** (/reference/android/widget/TextView#getMaxLines())**( )**<br><br>Returns the maximum height of TextView in terms of number of lines or -1 if the maximum height was set using **`setMaxHeight(int)`** (/reference/android/widget/TextView#setMaxHeight(int)) or **`setHeight(int)`** (/reference/android/widget/TextView#setHeight(int)). |
| `int` | **`getMaxWidth`** (/reference/android/widget/TextView#getMaxWidth())**( )**<br><br>Returns the maximum width of TextView in terms of pixels or -1 if the maximum width was set using **`setMaxEms(int)`** (/reference/android/widget/TextView#setMaxEms(int)) or **`setEms(int)`** (/reference/android/widget/TextView#setEms(int)). |
| `int` | **`getMinEms`** (/reference/android/widget/TextView#getMinEms())**( )** |

| | |
|---|---|
| | Returns the minimum width of TextView in terms of ems or -1 if the minimum width was set using **setMinWidth(int)** (/reference/android/widget/TextView#setMinWidth(int)) or **setWidth(int)** (/reference/android/widget/TextView#setWidth(int)). |
| **int** | **getMinHeight** (/reference/android/widget/TextView#getMinHeight())() <br><br> Returns the minimum height of TextView in terms of pixels or -1 if the minimum height was set using **setMinLines(int)** (/reference/android/widget/TextView#setMinLines(int)) or **setLines(int)** (/reference/android/widget/TextView#setLines(int)). |
| **int** | **getMinLines** (/reference/android/widget/TextView#getMinLines())() <br><br> Returns the minimum height of TextView in terms of number of lines or -1 if the minimum height was set using **setMinHeight(int)** (/reference/android/widget/TextView#setMinHeight(int)) or **setHeight(int)** (/reference/android/widget/TextView#setHeight(int)). |
| **int** | **getMinWidth** (/reference/android/widget/TextView#getMinWidth())() <br><br> Returns the minimum width of TextView in terms of pixels or -1 if the minimum width was set using **setMinEms(int)** (/reference/android/widget/TextView#setMinEms(int)) or **setEms(int)** (/reference/android/widget/TextView#setEms(int)). |
| **final MovementMethod** (/reference/android/text/method/MovementMethod) | **getMovementMethod** (/reference/android/widget/TextView#getMovementMethod())() <br><br> Gets the **MovementMethod** (/reference/android/text/method/MovementMethod) being used for this TextView, which provides positioning, scrolling, and text selection functionality. |
| **int** | **getOffsetForPosition** (/reference/android/widget/TextView#getOffsetForPosition(float,%20float))(**float x, float y**) <br><br> Get the character offset closest to the specified absolute position. |
| **TextPaint** (/reference/android/text/TextPaint) | **getPaint** (/reference/android/widget/TextView#getPaint())() <br><br> Gets the **TextPaint** (/reference/android/text/TextPaint) used for the text. |
| **int** | **getPaintFlags** (/reference/android/widget/TextView#getPaintFlags())() |

| | |
|---|---|
| | Gets the flags on the Paint being used to display the text. |
| String (/reference/java/lang/String) | getPrivateImeOptions (/reference/android/widget/TextView#getPrivateImeOptions())()<br><br>Get the private type of the content. |
| int | getSelectionEnd (/reference/android/widget/TextView#getSelectionEnd())()<br><br>Convenience for Selection#getSelectionEnd (/reference/android/text/Selection#getSelectionEnd(java.lang.CharSequence)). |
| int | getSelectionStart (/reference/android/widget/TextView#getSelectionStart())()<br><br>Convenience for Selection#getSelectionStart (/reference/android/text/Selection#getSelectionStart(java.lang.CharSequence)). |
| int | getShadowColor (/reference/android/widget/TextView#getShadowColor())()<br><br>Gets the color of the shadow layer. |
| float | getShadowDx (/reference/android/widget/TextView#getShadowDx())() |
| float | getShadowDy (/reference/android/widget/TextView#getShadowDy())()<br><br>Gets the vertical offset of the shadow layer. |
| float | getShadowRadius (/reference/android/widget/TextView#getShadowRadius())()<br><br>Gets the radius of the shadow layer. |
| final boolean | getShowSoftInputOnFocus (/reference/android/widget/TextView#getShowSoftInputOnFocus())()<br><br>Returns whether the soft input method will be made visible when this TextView gets focused. |
| CharSequence | getText (/reference/android/widget/TextView#getText())() |

| | |
|---|---|
| (/reference/java/lang/CharSequence) | Return the text that TextView is displaying. |
| **TextClassifier** (/reference/android/view/textclassifier/TextClassifier) | **getTextClassifier** (/reference/android/widget/TextView#getTextClassifier())() Returns the **TextClassifier** (/reference/android/view/textclassifier/TextClassifier) used by this TextView. |
| **final ColorStateList** (/reference/android/content/res/ColorStateList) | **getTextColors** (/reference/android/widget/TextView#getTextColors())() Gets the text colors for the different states (normal, selected, focused) of the TextView. |
| **Drawable** (/reference/android/graphics/drawable/Drawable) | **getTextCursorDrawable** (/reference/android/widget/TextView#getTextCursorDrawable())() Returns the Drawable corresponding to the text cursor. |
| **TextDirectionHeuristic** (/reference/android/text/TextDirectionHeuristic) | **getTextDirectionHeuristic** (/reference/android/widget/TextView#getTextDirectionHeuristic())() Returns resolved **TextDirectionHeuristic** (/reference/android/text/TextDirectionHeuristic) that will be used for text layout. |
| **Locale** (/reference/java/util/Locale) | **getTextLocale** (/reference/android/widget/TextView#getTextLocale())() Get the default primary **Locale** (/reference/java/util/Locale) of the text in this TextView. |
| **LocaleList** (/reference/android/os/LocaleList) | **getTextLocales** (/reference/android/widget/TextView#getTextLocales())() Get the default **LocaleList** (/reference/android/os/LocaleList) of the text in this TextView. |
| **PrecomputedText.Params** (/reference/android/text/PrecomputedText.Params) | **getTextMetricsParams** (/reference/android/widget/TextView#getTextMetricsParams())() Gets the parameters for text layout precomputation, for use with **PrecomputedText** (/reference/android/text/PrecomputedText). |

| | |
|---|---|
| `float` | **getTextScaleX** (/reference/android/widget/TextView#getTextScaleX())() <br><br> Gets the extent by which text should be stretched horizontally. |
| Drawable (/reference/android/graphics/drawable/Drawable) | **getTextSelectHandle** (/reference/android/widget/TextView#getTextSelectHandle())() <br><br> Returns the Drawable corresponding to the selection handle used for positioning the cursor within text. |
| Drawable (/reference/android/graphics/drawable/Drawable) | **getTextSelectHandleLeft** (/reference/android/widget/TextView#getTextSelectHandleLeft())() <br><br> Returns the Drawable corresponding to the left handle used for selecting text. |
| Drawable (/reference/android/graphics/drawable/Drawable) | **getTextSelectHandleRight** (/reference/android/widget/TextView#getTextSelectHandleRight())() <br><br> Returns the Drawable corresponding to the right handle used for selecting text. |
| `float` | **getTextSize** (/reference/android/widget/TextView#getTextSize())() |
| `int` | **getTotalPaddingBottom** (/reference/android/widget/TextView#getTotalPaddingBottom())() <br><br> Returns the total bottom padding of the view, including the bottom Drawable if any, the extra space to keep more than maxLines from showing, and the vertical offset for gravity, if any. |
| `int` | **getTotalPaddingEnd** (/reference/android/widget/TextView#getTotalPaddingEnd())() <br><br> Returns the total end padding of the view, including the end Drawable if any. |
| `int` | **getTotalPaddingLeft** (/reference/android/widget/TextView#getTotalPaddingLeft())() <br><br> Returns the total left padding of the view, including the left Drawable if any. |
| `int` | **getTotalPaddingRight** (/reference/android/widget/TextView#getTotalPaddingRight())() <br><br> Returns the total right padding of the view, including the right Drawable if any. |

| | |
|---|---|
| `int` | **`getTotalPaddingStart`** (/reference/android/widget/TextView#getTotalPaddingStart())**`()`**<br><br>Returns the total start padding of the view, including the start Drawable if any. |
| `int` | **`getTotalPaddingTop`** (/reference/android/widget/TextView#getTotalPaddingTop())**`()`**<br><br>Returns the total top padding of the view, including the top Drawable if any, the extra space to keep more than maxLines from showing, and the vertical offset for gravity, if any. |
| `final` **`TransformationMethod`** (/reference/android/text/method/TransformationMethod) | **`getTransformationMethod`** (/reference/android/widget/TextView#getTransformationMethod())**`()`**<br><br>Gets the current **`TransformationMethod`** (/reference/android/text/method/TransformationMethod) for the TextView. |
| **`Typeface`** (/reference/android/graphics/Typeface) | **`getTypeface`** (/reference/android/widget/TextView#getTypeface())**`()`**<br><br>Gets the current **`Typeface`** (/reference/android/graphics/Typeface) that is used to style the text. |
| **`URLSpan[]`** (/reference/android/text/style/URLSpan) | **`getUrls`** (/reference/android/widget/TextView#getUrls())**`()`**<br><br>Returns the list of **`URLSpans`** (/reference/android/text/style/URLSpan) attached to the text (by **`Linkify`** (/reference/android/text/util/Linkify) or otherwise) if any. |
| `boolean` | **`hasOverlappingRendering`** (/reference/android/widget/TextView#hasOverlappingRendering())**`()`**<br><br>Returns whether this View has content which overlaps. |
| `boolean` | **`hasSelection`** (/reference/android/widget/TextView#hasSelection())**`()`**<br><br>Return true iff there is a selection of nonzero length inside this text view. |
| `void` | **`invalidateDrawable`** (/reference/android/widget/TextView#invalidateDrawable(android.graphics.drawable.Drawable))**`(`**`Drawable`** (/reference/android/graphics/drawable/Drawable) **`drawable)`** |

| | |
|---|---|
| | Invalidates the specified Drawable. |
| **boolean** | **isAllCaps** (/reference/android/widget/TextView#isAllCaps())() <br> Checks whether the transformation method applied to this TextView is set to ALL CAPS. |
| **boolean** | **isCursorVisible** (/reference/android/widget/TextView#isCursorVisible())() |
| **boolean** | **isElegantTextHeight** (/reference/android/widget/TextView#isElegantTextHeight())() <br> Get the value of the TextView's elegant height metrics flag. |
| **boolean** | **isFallbackLineSpacing** (/reference/android/widget/TextView#isFallbackLineSpacing())() |
| **final boolean** | **isHorizontallyScrollable** (/reference/android/widget/TextView#isHorizontallyScrollable())() <br> Returns whether the text is allowed to be wider than the View. |
| **boolean** | **isInputMethodTarget** (/reference/android/widget/TextView#isInputMethodTarget())() <br> Returns whether this text view is a current input method target. |
| **boolean** | **isSingleLine** (/reference/android/widget/TextView#isSingleLine())() <br> Returns if the text is constrained to a single horizontally scrolling line ignoring new line characters instead of letting it wrap onto multiple lines. |
| **boolean** | **isSuggestionsEnabled** (/reference/android/widget/TextView#isSuggestionsEnabled())() <br> Return whether or not suggestions are enabled on this TextView. |
| **boolean** | **isTextSelectable** (/reference/android/widget/TextView#isTextSelectable())() <br> Returns the state of the **textIsSelectable** flag (See **setTextIsSelectable()** (/reference/android/widget/TextView#setTextIsSelectable(boolean))). |

| | |
|---|---|
| `void` | **jumpDrawablesToCurrentState** (/reference/android/widget/TextView#jumpDrawablesToCurrentState())()<br><br>Call **Drawable#jumpToCurrentState()** (/reference/android/graphics/drawable/Drawable#jumpToCurrentState()) on all Drawable objects associated with this view. |
| `int` | **length** (/reference/android/widget/TextView#length())()<br><br>Returns the length, in characters, of the text managed by this TextView |
| `boolean` | **moveCursorToVisibleOffset** (/reference/android/widget/TextView#moveCursorToVisibleOffset())()<br><br>Move the cursor, if needed, so that it is at an offset that is visible to the user. |
| `void` | **onBeginBatchEdit** (/reference/android/widget/TextView#onBeginBatchEdit())()<br><br>Called by the framework in response to a request to begin a batch of edit operations through a call to link **beginBatchEdit()** (/reference/android/widget/TextView#beginBatchEdit()). |
| `boolean` | **onCheckIsTextEditor** (/reference/android/widget/TextView#onCheckIsTextEditor())()<br><br>Check whether the called view is a text editor, in which case it would make sense to automatically display a soft input window for it. |
| `void` | **onCommitCompletion** (/reference/android/widget/TextView#onCommitCompletion(android.view.inputmethod.CompletionInfo))(**CompletionInfo** (/reference/android/view/inputmethod/CompletionInfo) `text`)<br><br>Called by the framework in response to a text completion from the current input method, provided by it calling **InputConnection#commitCompletion** (/reference/android/view/inputmethod/InputConnection#commitCompletion(android.view.inputmethod.CompletionInfo)). |
| `void` | **onCommitCorrection** (/reference/android/widget/TextView#onCommitCorrection(android.view.inputmethod.CorrectionInfo))(**CorrectionInfo** (/reference/android/view/inputmethod/CorrectionInfo) `info`)<br><br>Called by the framework in response to a text auto-correction (such as fixing a typo using a dictionary) from the current input method, provided by it calling **InputConnection#commitCorrection(CorrectionInfo)** (/reference/android/view/inputmethod/InputConnection#commitCorrection(android.view.inputmethod.CorrectionInfo)). |

| | |
|---|---|
| InputConnection (/reference/android/view/inputmethod/InputConnection) | onCreateInputConnection (/reference/android/widget/TextView#onCreateInputConnection(android.view.inputmethod.EditorInfo))(EditorInfo (/reference/android/view/inputmethod/EditorInfo) outAttrs)<br><br>Create a new InputConnection for an InputMethod to interact with the view. |
| boolean | onDragEvent (/reference/android/widget/TextView#onDragEvent(android.view.DragEvent))(DragEvent (/reference/android/view/DragEvent) event)<br><br>Handles drag events sent by the system following a call to startDragAndDrop() (/reference/android/view/View#startDragAndDrop(android.content.ClipData,%20android.view.View.DragShadowBuilder,%20java.lang.Object,%20int)). |
| void | onEditorAction (/reference/android/widget/TextView#onEditorAction(int))(int actionCode)<br><br>Called when an attached input method calls InputConnection#performEditorAction(int) (/reference/android/view/inputmethod/InputConnection#performEditorAction(int)) for this text view. |
| void | onEndBatchEdit (/reference/android/widget/TextView#onEndBatchEdit())()<br><br>Called by the framework in response to a request to end a batch of edit operations through a call to link endBatchEdit() (/reference/android/widget/TextView#endBatchEdit()). |
| boolean | onGenericMotionEvent (/reference/android/widget/TextView#onGenericMotionEvent(android.view.MotionEvent))(MotionEvent (/reference/android/view/MotionEvent) event)<br><br>Implement this method to handle generic motion events. |
| boolean | onKeyDown (/reference/android/widget/TextView#onKeyDown(int,%20android.view.KeyEvent))(int keyCode, KeyEvent (/reference/android/view/KeyEvent) event)<br><br>Default implementation of KeyEvent.Callback#onKeyDown(int, KeyEvent) (/reference/android/view/KeyEvent.Callback#onKeyDown(int,%20android.view.KeyEvent)): perform press of the view when KeyEvent#KEYCODE_DPAD_CENTER (/reference/android/view/KeyEvent#KEYCODE_DPAD_CENTER) or KeyEvent#KEYCODE_ENTER (/reference/android/view/KeyEvent#KEYCODE_ENTER) is released, if the view is enabled and clickable. |

| | |
|---|---|
| **boolean** | **onKeyMultiple** (/reference/android/widget/TextView#onKeyMultiple(int,%20int,%20android.view.KeyEvent))**(int keyCode, int repeatCount, KeyEvent** (/reference/android/view/KeyEvent) **event)**<br><br>Default implementation of **KeyEvent.Callback#onKeyMultiple(int, int, KeyEvent)** (/reference/android/view/KeyEvent.Callback#onKeyMultiple(int,%20int,%20android.view.KeyEvent)): always returns false (doesn't handle the event). |
| **boolean** | **onKeyPreIme** (/reference/android/widget/TextView#onKeyPreIme(int,%20android.view.KeyEvent))**(int keyCode, KeyEvent** (/reference/android/view/KeyEvent) **event)**<br><br>Handle a key event before it is processed by any input method associated with the view hierarchy. |
| **boolean** | **onKeyShortcut** (/reference/android/widget/TextView#onKeyShortcut(int,%20android.view.KeyEvent))**(int keyCode, KeyEvent** (/reference/android/view/KeyEvent) **event)**<br><br>Called on the focused view when a key shortcut event is not handled. |
| **boolean** | **onKeyUp** (/reference/android/widget/TextView#onKeyUp(int,%20android.view.KeyEvent))**(int keyCode, KeyEvent** (/reference/android/view/KeyEvent) **event)**<br><br>Default implementation of **KeyEvent.Callback#onKeyUp(int, KeyEvent)** (/reference/android/view/KeyEvent.Callback#onKeyUp(int,%20android.view.KeyEvent)): perform clicking of the view when **KeyEvent#KEYCODE_DPAD_CENTER** (/reference/android/view/KeyEvent#KEYCODE_DPAD_CENTER), **KeyEvent#KEYCODE_ENTER** (/reference/android/view/KeyEvent#KEYCODE_ENTER) or **KeyEvent#KEYCODE_SPACE** (/reference/android/view/KeyEvent#KEYCODE_SPACE) is released. |
| **boolean** | **onPreDraw** (/reference/android/widget/TextView#onPreDraw())**()**<br><br>Callback method to be invoked when the view tree is about to be drawn. |
| **boolean** | **onPrivateIMECommand** (/reference/android/widget/TextView#onPrivateIMECommand(java.lang.String,%20android.os.Bundle))**(String** (/reference/java/lang/String) **action, Bundle** (/reference/android/os/Bundle) **data)**<br><br>Called by the framework in response to a private command from the current method, provided by it calling **InputConnection#performPrivateCommand** (/reference/android/view/inputmethod/InputConnection#performPrivateCommand(java.lang.String,%20android.os.Bundle)). |

| | |
|---|---|
| PointerIcon (/reference/android/view/PointerIcon) | onResolvePointerIcon (/reference/android/widget/TextView#onResolvePointerIcon(android.view.MotionEvent,%20int))(MotionEvent (/reference/android/view/MotionEvent) event, int pointerIndex)<br><br>Returns the pointer icon for the motion event, or null if it doesn't specify the icon. |
| void | onRestoreInstanceState (/reference/android/widget/TextView#onRestoreInstanceState(android.os.Parcelable))(Parcelable (/reference/android/os/Parcelable) state)<br><br>Hook allowing a view to re-apply a representation of its internal state that had previously been generated by onSaveInstanceState() (/reference/android/view/View#onSaveInstanceState()). |
| void | onRtlPropertiesChanged (/reference/android/widget/TextView#onRtlPropertiesChanged(int))(int layoutDirection)<br><br>Called when any RTL property (layout direction or text direction or text alignment) has been changed. |
| Parcelable (/reference/android/os/Parcelable) | onSaveInstanceState (/reference/android/widget/TextView#onSaveInstanceState())()<br><br>Hook allowing a view to generate a representation of its internal state that can later be used to create a new instance with that same state. |
| void | onScreenStateChanged (/reference/android/widget/TextView#onScreenStateChanged(int))(int screenState)<br><br>This method is called whenever the state of the screen this view is attached to changes. |
| boolean | onTextContextMenuItem (/reference/android/widget/TextView#onTextContextMenuItem(int))(int id)<br><br>Called when a context menu option for the text view is selected. |
| boolean | onTouchEvent (/reference/android/widget/TextView#onTouchEvent(android.view.MotionEvent))(MotionEvent (/reference/android/view/MotionEvent) event)<br><br>Implement this method to handle touch screen motion events. |
| boolean | onTrackballEvent (/reference/android/widget/TextView#onTrackballEvent(android.view.MotionEvent))(MotionEvent (/reference/android/view/MotionEvent) event) |

| | |
|---|---|
| | Implement this method to handle trackball motion events. |
| **void** | **onWindowFocusChanged** (/reference/android/widget/TextView#onWindowFocusChanged(boolean))**(boolean hasWindowFocus)**<br><br>Called when the window containing this view gains or loses focus. |
| **boolean** | **performLongClick** (/reference/android/widget/TextView#performLongClick())**()**<br><br>Calls this view's OnLongClickListener, if it is defined. |
| **void** | **removeTextChangedListener** (/reference/android/widget/TextView#removeTextChangedListener(android.text.TextWatcher))**(TextWatcher** (/reference/android/text/TextWatcher) **watcher)**<br><br>Removes the specified TextWatcher from the list of those whose methods are called whenever this TextView's text changes. |
| **void** | **sendAccessibilityEventUnchecked** (/reference/android/widget/TextView#sendAccessibilityEventUnchecked(android.view.accessibility.AccessibilityEvent))**(AccessibilityEvent** (/reference/android/view/accessibility/AccessibilityEvent) **event)**<br><br>This method behaves exactly as **sendAccessibilityEvent(int)** (/reference/android/view/View#sendAccessibilityEvent(int)) but takes as an argument an empty **AccessibilityEvent** (/reference/android/view/accessibility/AccessibilityEvent) and does not perform a check whether accessibility is enabled. |
| **void** | **setAllCaps** (/reference/android/widget/TextView#setAllCaps(boolean))**(boolean allCaps)**<br><br>Sets the properties of this field to transform input to ALL CAPS display. |
| **final void** | **setAutoLinkMask** (/reference/android/widget/TextView#setAutoLinkMask(int))**(int mask)**<br><br>Sets the autolink mask of the text. |
| **void** | **setAutoSizeTextTypeUniformWithConfiguration** (/reference/android/widget/TextView#setAutoSizeTextTypeUniformWithConfiguration(int,%20int,%20int,%20int))**(int autoSizeMinTextSize, int autoSizeMaxTextSize, int autoSizeStepGranularity, int unit)** |

| | |
|---|---|
| | Specify whether this widget should automatically scale the text to try to perfectly fit within the layout bounds. |
| **void** | **setAutoSizeTextTypeUniformWithPresetSizes** (/reference/android/widget/TextView#setAutoSizeTextTypeUniformWithPresetSizes(int[],%20int))**(int[] presetSizes, int unit)**<br><br>Specify whether this widget should automatically scale the text to try to perfectly fit within the layout bounds. |
| **void** | **setAutoSizeTextTypeWithDefaults** (/reference/android/widget/TextView#setAutoSizeTextTypeWithDefaults(int))**(int autoSizeTextType)**<br><br>Specify whether this widget should automatically scale the text to try to perfectly fit within the layout bounds by using the default auto-size configuration. |
| **void** | **setBreakStrategy** (/reference/android/widget/TextView#setBreakStrategy(int))**(int breakStrategy)**<br><br>Sets the break strategy for breaking paragraphs into lines. |
| **void** | **setCompoundDrawablePadding** (/reference/android/widget/TextView#setCompoundDrawablePadding(int))**(int pad)**<br><br>Sets the size of the padding between the compound drawables and the text. |
| **void** | **setCompoundDrawableTintBlendMode** (/reference/android/widget/TextView#setCompoundDrawableTintBlendMode(android.graphics.BlendMode))**(BlendMode** (/reference/android/graphics/BlendMode) **blendMode)**<br><br>Specifies the blending mode used to apply the tint specified by **setCompoundDrawableTintList(android.content.res.ColorStateList)** (/reference/android/widget/TextView#setCompoundDrawableTintList(android.content.res.ColorStateList)) to the compound drawables. |
| **void** | **setCompoundDrawableTintList** (/reference/android/widget/TextView#setCompoundDrawableTintList(android.content.res.ColorStateList))**(ColorStateList** (/reference/android/content/res/ColorStateList) **tint)**<br><br>Applies a tint to the compound drawables. |
| **void** | **setCompoundDrawableTintMode** (/reference/android/widget/TextView#setCompoundDrawableTintMode(android.graphics.PorterDuff.Mode))**(PorterDuff.Mode** (/reference/android/graphics/PorterDuff.Mode) **tintMode)**<br><br>Specifies the blending mode used to apply the tint specified by **setCompoundDrawableTintList(android.content.res.ColorStateList)** |

(/reference/android/widget/TextView#setCompoundDrawableTintList(android.content.res.ColorStateList)) to the compound drawables.

| | |
|---|---|
| **void** | **setCompoundDrawables**<br>(/reference/android/widget/TextView#setCompoundDrawables(android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable))<br>**(Drawable** (/reference/android/graphics/drawable/Drawable) **left, Drawable** (/reference/android/graphics/drawable/Drawable) **top, Drawable** (/reference/android/graphics/drawable/Drawable) **right, Drawable** (/reference/android/graphics/drawable/Drawable) **bottom)**<br><br>Sets the Drawables (if any) to appear to the left of, above, to the right of, and below the text. |
| **void** | **setCompoundDrawablesRelative**<br>(/reference/android/widget/TextView#setCompoundDrawablesRelative(android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable))<br>**(Drawable** (/reference/android/graphics/drawable/Drawable) **start, Drawable** (/reference/android/graphics/drawable/Drawable) **top, Drawable** (/reference/android/graphics/drawable/Drawable) **end, Drawable** (/reference/android/graphics/drawable/Drawable) **bottom)**<br><br>Sets the Drawables (if any) to appear to the start of, above, to the end of, and below the text. |
| **void** | **setCompoundDrawablesRelativeWithIntrinsicBounds**<br>(/reference/android/widget/TextView#setCompoundDrawablesRelativeWithIntrinsicBounds(android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable))<br>**(Drawable** (/reference/android/graphics/drawable/Drawable) **start, Drawable** (/reference/android/graphics/drawable/Drawable) **top, Drawable** (/reference/android/graphics/drawable/Drawable) **end, Drawable** (/reference/android/graphics/drawable/Drawable) **bottom)**<br><br>Sets the Drawables (if any) to appear to the start of, above, to the end of, and below the text. |
| **void** | **setCompoundDrawablesRelativeWithIntrinsicBounds**<br>(/reference/android/widget/TextView#setCompoundDrawablesRelativeWithIntrinsicBounds(int,%20int,%20int,%20int))**(int start, int top, int end, int bottom)**<br><br>Sets the Drawables (if any) to appear to the start of, above, to the end of, and below the text. |
| **void** | **setCompoundDrawablesWithIntrinsicBounds** |

| | |
|---|---|
| | (/reference/android/widget/TextView#setCompoundDrawablesWithIntrinsicBounds(android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable))**(Drawable** (/reference/android/graphics/drawable/Drawable) **left, Drawable** (/reference/android/graphics/drawable/Drawable) **top, Drawable** (/reference/android/graphics/drawable/Drawable) **right, Drawable** (/reference/android/graphics/drawable/Drawable) **bottom)**<br><br>Sets the Drawables (if any) to appear to the left of, above, to the right of, and below the text. |
| **void** | **setCompoundDrawablesWithIntrinsicBounds** (/reference/android/widget/TextView#setCompoundDrawablesWithIntrinsicBounds(int,%20int,%20int,%20int))**(int left, int top, int right, int bottom)**<br><br>Sets the Drawables (if any) to appear to the left of, above, to the right of, and below the text. |
| **void** | **setCursorVisible** (/reference/android/widget/TextView#setCursorVisible(boolean))**(boolean visible)**<br><br>Set whether the cursor is visible. |
| **void** | **setCustomInsertionActionModeCallback** (/reference/android/widget/TextView#setCustomInsertionActionModeCallback(android.view.ActionMode.Callback))**(ActionMode.Callback** (/reference/android/view/ActionMode.Callback) **actionModeCallback)**<br><br>If provided, this ActionMode.Callback will be used to create the ActionMode when text insertion is initiated in this View. |
| **void** | **setCustomSelectionActionModeCallback** (/reference/android/widget/TextView#setCustomSelectionActionModeCallback(android.view.ActionMode.Callback))**(ActionMode.Callback** (/reference/android/view/ActionMode.Callback) **actionModeCallback)**<br><br>If provided, this ActionMode.Callback will be used to create the ActionMode when text selection is initiated in this View. |
| **final void** | **setEditableFactory** (/reference/android/widget/TextView#setEditableFactory(android.text.Editable.Factory))**(Editable.Factory** (/reference/android/text/Editable.Factory) **factory)**<br><br>Sets the Factory used to create new **Editable** (/reference/android/text/Editable). |

| | |
|---|---|
| **void** | **setElegantTextHeight** (/reference/android/widget/TextView#setElegantTextHeight(boolean))**(boolean elegant)**<br><br>Set the TextView's elegant height metrics flag. |
| **void** | **setEllipsize** (/reference/android/widget/TextView#setEllipsize(android.text.TextUtils.TruncateAt))**(TextUtils.TruncateAt** (/reference/android/text/TextUtils.TruncateAt) **where)**<br><br>Causes words in the text that are longer than the view's width to be ellipsized instead of broken in the middle. |
| **void** | **setEms** (/reference/android/widget/TextView#setEms(int))**(int ems)**<br><br>Sets the width of the TextView to be exactly **ems** wide. |
| **void** | **setEnabled** (/reference/android/widget/TextView#setEnabled(boolean))**(boolean enabled)**<br><br>Set the enabled state of this view. |
| **void** | **setError** (/reference/android/widget/TextView#setError(java.lang.CharSequence))**(CharSequence** (/reference/java/lang/CharSequence) **error)**<br><br>Sets the right-hand compound drawable of the TextView to the "error" icon and sets an error message that will be displayed in a popup when the TextView has focus. |
| **void** | **setError** (/reference/android/widget/TextView#setError(java.lang.CharSequence,%20android.graphics.drawable.Drawable))**(CharSequence** (/reference/java/lang/CharSequence) **error, Drawable** (/reference/android/graphics/drawable/Drawable) **icon)**<br><br>Sets the right-hand compound drawable of the TextView to the specified icon and sets an error message that will be displayed in a popup when the TextView has focus. |
| **void** | **setExtractedText** (/reference/android/widget/TextView#setExtractedText(android.view.inputmethod.ExtractedText))**(ExtractedText** (/reference/android/view/inputmethod/ExtractedText) **text)**<br><br>Apply to this text view the given extracted text, as previously returned by **extractText(android.view.inputmethod.ExtractedTextRequest, android.view.inputmethod.ExtractedText)** |

(/reference/android/widget/TextView#extractText(android.view.inputmethod.ExtractedTextRequest,%20android.view.inputmethod.ExtractedText)).

| | |
|---|---|
| **void** | **setFallbackLineSpacing** (/reference/android/widget/TextView#setFallbackLineSpacing(boolean))**(boolean enabled)**<br><br>Set whether to respect the ascent and descent of the fallback fonts that are used in displaying the text (which is needed to avoid text from consecutive lines running into each other). |
| **void** | **setFilters** (/reference/android/widget/TextView#setFilters(android.text.InputFilter[]))**(InputFilter[]** (/reference/android/text/InputFilter) **filters)**<br><br>Sets the list of input filters that will be used if the buffer is Editable. |
| **void** | **setFirstBaselineToTopHeight** (/reference/android/widget/TextView#setFirstBaselineToTopHeight(int))**(int firstBaselineToTopHeight)**<br><br>Updates the top padding of the TextView so that **firstBaselineToTopHeight** is the distance between the top of the TextView and first line's baseline. |
| **void** | **setFontFeatureSettings** (/reference/android/widget/TextView#setFontFeatureSettings(java.lang.String))**(String** (/reference/java/lang/String) **fontFeatureSettings)**<br><br>Sets font feature settings. |
| **boolean** | **setFontVariationSettings** (/reference/android/widget/TextView#setFontVariationSettings(java.lang.String))**(String** (/reference/java/lang/String) **fontVariationSettings)**<br><br>Sets TrueType or OpenType font variation settings. |
| **void** | **setFreezesText** (/reference/android/widget/TextView#setFreezesText(boolean))**(boolean freezesText)**<br><br>Control whether this text view saves its entire text contents when freezing to an icicle, in addition to dynamic state such as cursor position. |
| **void** | **setGravity** (/reference/android/widget/TextView#setGravity(int))**(int gravity)**<br><br>Sets the horizontal alignment of the text and the vertical gravity that will be used when there is extra space in the TextView beyond what is required for the text itself. |

| | |
|---|---|
| **void** | **setHeight** (/reference/android/widget/TextView#setHeight(int))**(int pixels)**<br><br>Sets the height of the TextView to be exactly **pixels** tall. |
| **void** | **setHighlightColor** (/reference/android/widget/TextView#setHighlightColor(int))**(int color)**<br><br>Sets the color used to display the selection highlight. |
| **final void** | **setHint** (/reference/android/widget/TextView#setHint(java.lang.CharSequence))**(CharSequence** (/reference/java/lang/CharSequence) **hint)**<br><br>Sets the text to be displayed when the text of the TextView is empty. |
| **final void** | **setHint** (/reference/android/widget/TextView#setHint(int))**(int resid)**<br><br>Sets the text to be displayed when the text of the TextView is empty, from a resource. |
| **final void** | **setHintTextColor** (/reference/android/widget/TextView#setHintTextColor(android.content.res.ColorStateList))**(ColorStateList** (/reference/android/content/res/ColorStateList) **colors)**<br><br>Sets the color of the hint text. |
| **final void** | **setHintTextColor** (/reference/android/widget/TextView#setHintTextColor(int))**(int color)**<br><br>Sets the color of the hint text for all the states (disabled, focussed, selected...) of this TextView. |
| **void** | **setHorizontallyScrolling** (/reference/android/widget/TextView#setHorizontallyScrolling(boolean))**(boolean whether)**<br><br>Sets whether the text should be allowed to be wider than the View is. |
| **void** | **setHyphenationFrequency** (/reference/android/widget/TextView#setHyphenationFrequency(int))**(int hyphenationFrequency)**<br><br>Sets the frequency of automatic hyphenation to use when determining word breaks. |
| **void** | **setImeActionLabel** (/reference/android/widget/TextView#setImeActionLabel(java.lang.CharSequence,%20int))**(CharSequence** |

| | |
|---|---|
| | (/reference/java/lang/CharSequence) **label, int actionId)**<br><br>Change the custom IME action associated with the text view, which will be reported to an IME with **EditorInfo#actionLabel** (/reference/android/view/inputmethod/EditorInfo#actionLabel) and **EditorInfo#actionId** (/reference/android/view/inputmethod/EditorInfo#actionId) when it has focus. |
| **void** | **setImeHintLocales** (/reference/android/widget/TextView#setImeHintLocales(android.os.LocaleList))**(LocaleList** (/reference/android/os/LocaleList) **hintLocales)**<br><br>Change "hint" locales associated with the text view, which will be reported to an IME with **EditorInfo#hintLocales** (/reference/android/view/inputmethod/EditorInfo#hintLocales) when it has focus. |
| **void** | **setImeOptions** (/reference/android/widget/TextView#setImeOptions(int))**(int imeOptions)**<br><br>Change the editor type integer associated with the text view, which is reported to an Input Method Editor (IME) with **EditorInfo#imeOptions** (/reference/android/view/inputmethod/EditorInfo#imeOptions) when it has focus. |
| **void** | **setIncludeFontPadding** (/reference/android/widget/TextView#setIncludeFontPadding(boolean))**(boolean includepad)**<br><br>Set whether the TextView includes extra top and bottom padding to make room for accents that go above the normal ascent and descent. |
| **void** | **setInputExtras** (/reference/android/widget/TextView#setInputExtras(int))**(int xmlResId)**<br><br>Set the extra input data of the text, which is the **EditorInfo#extras** (/reference/android/view/inputmethod/EditorInfo#extras) Bundle that will be filled in when creating an input connection. |
| **void** | **setInputType** (/reference/android/widget/TextView#setInputType(int))**(int type)**<br><br>Set the type of the content with a constant as defined for **EditorInfo#inputType** (/reference/android/view/inputmethod/EditorInfo#inputType). |
| **void** | **setJustificationMode** (/reference/android/widget/TextView#setJustificationMode(int))**(int justificationMode)**<br><br>Set justification mode. |

| | |
|---|---|
| `void` | **`setKeyListener`** (/reference/android/widget/TextView#setKeyListener(android.text.method.KeyListener))**`(`**`KeyListener`** (/reference/android/text/method/KeyListener) **`input)`**<br><br>Sets the key listener to be used with this TextView. |
| `void` | **`setLastBaselineToBottomHeight`** (/reference/android/widget/TextView#setLastBaselineToBottomHeight(int))**`(int lastBaselineToBottomHeight)`**<br><br>Updates the bottom padding of the TextView so that **`lastBaselineToBottomHeight`** is the distance between the bottom of the TextView and the last line's baseline. |
| `void` | **`setLetterSpacing`** (/reference/android/widget/TextView#setLetterSpacing(float))**`(float letterSpacing)`**<br><br>Sets text letter-spacing in em units. |
| `void` | **`setLineHeight`** (/reference/android/widget/TextView#setLineHeight(int))**`(int lineHeight)`**<br><br>Sets an explicit line height for this TextView. |
| `void` | **`setLineSpacing`** (/reference/android/widget/TextView#setLineSpacing(float,%20float))**`(float add, float mult)`**<br><br>Sets line spacing for this TextView. |
| `void` | **`setLines`** (/reference/android/widget/TextView#setLines(int))**`(int lines)`**<br><br>Sets the height of the TextView to be exactly **`lines`** tall. |
| `final void` | **`setLinkTextColor`** (/reference/android/widget/TextView#setLinkTextColor(android.content.res.ColorStateList))**`(`**`ColorStateList`** (/reference/android/content/res/ColorStateList) **`colors)`**<br><br>Sets the color of links in the text. |
| `final void` | **`setLinkTextColor`** (/reference/android/widget/TextView#setLinkTextColor(int))**`(int color)`**<br><br>Sets the color of links in the text. |

| | |
|---|---|
| `final void` | **`setLinksClickable`** (/reference/android/widget/TextView#setLinksClickable(boolean))**`(boolean whether)`**<br><br>Sets whether the movement method will automatically be set to **`LinkMovementMethod`** (/reference/android/text/method/LinkMovementMethod) if **`setAutoLinkMask(int)`** (/reference/android/widget/TextView#setAutoLinkMask(int)) has been set to nonzero and links are detected in **`setText(char[], int, int)`** (/reference/android/widget/TextView#setText(char[],%20int,%20int)). |
| `void` | **`setMarqueeRepeatLimit`** (/reference/android/widget/TextView#setMarqueeRepeatLimit(int))**`(int marqueeLimit)`**<br><br>Sets how many times to repeat the marquee animation. |
| `void` | **`setMaxEms`** (/reference/android/widget/TextView#setMaxEms(int))**`(int maxEms)`**<br><br>Sets the width of the TextView to be at most **`maxEms`** wide. |
| `void` | **`setMaxHeight`** (/reference/android/widget/TextView#setMaxHeight(int))**`(int maxPixels)`**<br><br>Sets the height of the TextView to be at most **`maxPixels`** tall. |
| `void` | **`setMaxLines`** (/reference/android/widget/TextView#setMaxLines(int))**`(int maxLines)`**<br><br>Sets the height of the TextView to be at most **`maxLines`** tall. |
| `void` | **`setMaxWidth`** (/reference/android/widget/TextView#setMaxWidth(int))**`(int maxPixels)`**<br><br>Sets the width of the TextView to be at most **`maxPixels`** wide. |
| `void` | **`setMinEms`** (/reference/android/widget/TextView#setMinEms(int))**`(int minEms)`**<br><br>Sets the width of the TextView to be at least **`minEms`** wide. |
| `void` | **`setMinHeight`** (/reference/android/widget/TextView#setMinHeight(int))**`(int minPixels)`**<br><br>Sets the height of the TextView to be at least **`minPixels`** tall. |

| | |
|---|---|
| **void** | **setMinLines** (/reference/android/widget/TextView#setMinLines(int))**(int minLines)**<br><br>Sets the height of the TextView to be at least **minLines** tall. |
| **void** | **setMinWidth** (/reference/android/widget/TextView#setMinWidth(int))**(int minPixels)**<br><br>Sets the width of the TextView to be at least **minPixels** wide. |
| **final void** | **setMovementMethod** (/reference/android/widget/TextView#setMovementMethod(android.text.method.MovementMethod))**(MovementMethod** (/reference/android/text/method/MovementMethod) **movement)**<br><br>Sets the **MovementMethod** (/reference/android/text/method/MovementMethod) for handling arrow key movement for this TextView. |
| **void** | **setOnEditorActionListener** (/reference/android/widget/TextView#setOnEditorActionListener(android.widget.TextView.OnEditorActionListener))**(TextView.OnEditorActionListener** (/reference/android/widget/TextView.OnEditorActionListener) **l)**<br><br>Set a special listener to be called when an action is performed on the text view. |
| **void** | **setPadding** (/reference/android/widget/TextView#setPadding(int,%20int,%20int,%20int))**(int left, int top, int right, int bottom)**<br><br>Sets the padding. |
| **void** | **setPaddingRelative** (/reference/android/widget/TextView#setPaddingRelative(int,%20int,%20int,%20int))**(int start, int top, int end, int bottom)**<br><br>Sets the relative padding. |
| **void** | **setPaintFlags** (/reference/android/widget/TextView#setPaintFlags(int))**(int flags)**<br><br>Sets flags on the Paint being used to display the text and reflows the text if they are different from the old flags. |
| **void** | **setPrivateImeOptions** (/reference/android/widget/TextView#setPrivateImeOptions(java.lang.String))**(String** (/reference/java/lang/String) **type)**<br><br>Set the private content type of the text, which is the **EditorInfo#privateImeOptions** (/reference/android/view/inputmethod/EditorInfo#privateImeOptions) |

| | |
|---|---|
| | field that will be filled in when creating an input connection. |
| `void` | **setRawInputType** (/reference/android/widget/TextView#setRawInputType(int))**(int type)**<br><br>Directly change the content type integer of the text view, without modifying any other state. |
| `void` | **setScroller** (/reference/android/widget/TextView#setScroller(android.widget.Scroller))**(**Scroller (/reference/android/widget/Scroller) **s)**<br><br>Sets the Scroller used for producing a scrolling animation |
| `void` | **setSelectAllOnFocus** (/reference/android/widget/TextView#setSelectAllOnFocus(boolean))**(boolean selectAllOnFocus)**<br><br>Set the TextView so that when it takes focus, all the text is selected. |
| `void` | **setSelected** (/reference/android/widget/TextView#setSelected(boolean))**(boolean selected)**<br><br>Changes the selection state of this view. |
| `void` | **setShadowLayer** (/reference/android/widget/TextView#setShadowLayer(float,%20float,%20float,%20int))**(float radius, float dx, float dy, int color)**<br><br>Gives the text a shadow of the specified blur radius and color, the specified distance from its drawn position. |
| `final void` | **setShowSoftInputOnFocus** (/reference/android/widget/TextView#setShowSoftInputOnFocus(boolean))**(boolean show)**<br><br>Sets whether the soft input method will be made visible when this TextView gets focused. |
| `void` | **setSingleLine** (/reference/android/widget/TextView#setSingleLine(boolean))**(boolean singleLine)**<br><br>If true, sets the properties of this field (number of lines, horizontally scrolling, transformation method) to be for a single-line input; if false, restores these to the default conditions. |
| `void` | **setSingleLine** (/reference/android/widget/TextView#setSingleLine())**()** |

| | |
|---|---|
| | Sets the properties of this field (lines, horizontally scrolling, transformation method) to be for a single-line input. |
| `final void` | **setSpannableFactory** (/reference/android/widget/TextView#setSpannableFactory(android.text.Spannable.Factory))**(Spannable.Factory** (/reference/android/text/Spannable.Factory) **factory)**<br><br>Sets the Factory used to create new **Spannable** (/reference/android/text/Spannable). |
| `final void` | **setText** (/reference/android/widget/TextView#setText(int))**(int resid)**<br><br>Sets the text to be displayed using a string resource identifier. |
| `final void` | **setText** (/reference/android/widget/TextView#setText(java.lang.CharSequence))**(CharSequence** (/reference/java/lang/CharSequence) **text)**<br><br>Sets the text to be displayed. |
| `void` | **setText** (/reference/android/widget/TextView#setText(java.lang.CharSequence,%20android.widget.TextView.BufferType))**(CharSequence** (/reference/java/lang/CharSequence) **text, TextView.BufferType** (/reference/android/widget/TextView.BufferType) **type)**<br><br>Sets the text to be displayed and the **TextView.BufferType** (/reference/android/widget/TextView.BufferType). |
| `final void` | **setText** (/reference/android/widget/TextView#setText(int,%20android.widget.TextView.BufferType))**(int resid, TextView.BufferType** (/reference/android/widget/TextView.BufferType) **type)**<br><br>Sets the text to be displayed using a string resource identifier and the **TextView.BufferType** (/reference/android/widget/TextView.BufferType). |
| `final void` | **setText** (/reference/android/widget/TextView#setText(char[],%20int,%20int))**(char[] text, int start, int len)**<br><br>Sets the TextView to display the specified slice of the specified char array. |
| `void` | **setTextAppearance** (/reference/android/widget/TextView#setTextAppearance(android.content.Context,%20int))**(Context** (/reference/android/content/Context) **context, int resId)**<br><br>*This method was deprecated in API level 23. Use* **setTextAppearance(int)** (/reference/android/widget/TextView#setTextAppearance(int)) *instead.* |

| | |
|---|---|
| **void** | **setTextAppearance** (/reference/android/widget/TextView#setTextAppearance(int))**(int resId)**<br><br>Sets the text appearance from the specified style resource. |
| **void** | **setTextClassifier** (/reference/android/widget/TextView#setTextClassifier(android.view.textclassifier.TextClassifier))**(**TextClassifier (/reference/android/view/textclassifier/TextClassifier) **textClassifier)**<br><br>Sets the **TextClassifier** (/reference/android/view/textclassifier/TextClassifier) for this TextView. |
| **void** | **setTextColor** (/reference/android/widget/TextView#setTextColor(int))**(int color)**<br><br>Sets the text color for all the states (normal, selected, focused) to be this color. |
| **void** | **setTextColor** (/reference/android/widget/TextView#setTextColor(android.content.res.ColorStateList))**(**ColorStateList (/reference/android/content/res/ColorStateList) **colors)**<br><br>Sets the text color. |
| **void** | **setTextCursorDrawable** (/reference/android/widget/TextView#setTextCursorDrawable(android.graphics.drawable.Drawable))**(**Drawable (/reference/android/graphics/drawable/Drawable) **textCursorDrawable)**<br><br>Sets the Drawable corresponding to the text cursor. |
| **void** | **setTextCursorDrawable** (/reference/android/widget/TextView#setTextCursorDrawable(int))**(int textCursorDrawable)**<br><br>Sets the Drawable corresponding to the text cursor. |
| **void** | **setTextIsSelectable** (/reference/android/widget/TextView#setTextIsSelectable(boolean))**(boolean selectable)**<br><br>Sets whether the content of this view is selectable by the user. |
| **final void** | **setTextKeepState** (/reference/android/widget/TextView#setTextKeepState(java.lang.CharSequence))**(**CharSequence (/reference/java/lang/CharSequence) **text)** |

| | |
|---|---|
| | Sets the text to be displayed but retains the cursor position. |
| `final void` | **setTextKeepState** (/reference/android/widget/TextView#setTextKeepState(java.lang.CharSequence,%20android.widget.TextView.BufferType))**(CharSequence** (/reference/java/lang/CharSequence) **text, TextView.BufferType** (/reference/android/widget/TextView.BufferType) **type)**<br><br>Sets the text to be displayed and the **TextView.BufferType** (/reference/android/widget/TextView.BufferType) but retains the cursor position. |
| `void` | **setTextLocale** (/reference/android/widget/TextView#setTextLocale(java.util.Locale))**(Locale** (/reference/java/util/Locale) **locale)**<br><br>Set the default **Locale** (/reference/java/util/Locale) of the text in this TextView to a one-member **LocaleList** (/reference/android/os/LocaleList) containing just the given Locale. |
| `void` | **setTextLocales** (/reference/android/widget/TextView#setTextLocales(android.os.LocaleList))**(LocaleList** (/reference/android/os/LocaleList) **locales)**<br><br>Set the default **LocaleList** (/reference/android/os/LocaleList) of the text in this TextView to the given value. |
| `void` | **setTextMetricsParams** (/reference/android/widget/TextView#setTextMetricsParams(android.text.PrecomputedText.Params))**(PrecomputedText.Params** (/reference/android/text/PrecomputedText.Params) **params)**<br><br>Apply the text layout parameter. |
| `void` | **setTextScaleX** (/reference/android/widget/TextView#setTextScaleX(float))**(float size)**<br><br>Sets the horizontal scale factor for text. |
| `void` | **setTextSelectHandle** (/reference/android/widget/TextView#setTextSelectHandle(int))**(int textSelectHandle)**<br><br>Sets the Drawable corresponding to the selection handle used for positioning the cursor within text. |
| `void` | **setTextSelectHandle** (/reference/android/widget/TextView#setTextSelectHandle(android.graphics.drawable.Drawable))**(Drawable** (/reference/android/graphics/drawable/Drawable) **textSelectHandle)**<br><br>Sets the Drawable corresponding to the selection handle used for positioning the cursor within text. |

| | |
|---|---|
| **void** | **setTextSelectHandleLeft** (/reference/android/widget/TextView#setTextSelectHandleLeft(int))**(int textSelectHandleLeft)**<br><br>Sets the Drawable corresponding to the left handle used for selecting text. |
| **void** | **setTextSelectHandleLeft** (/reference/android/widget/TextView#setTextSelectHandleLeft(android.graphics.drawable.Drawable))**(Drawable** (/reference/android/graphics/drawable/Drawable) **textSelectHandleLeft)**<br><br>Sets the Drawable corresponding to the left handle used for selecting text. |
| **void** | **setTextSelectHandleRight** (/reference/android/widget/TextView#setTextSelectHandleRight(android.graphics.drawable.Drawable))**(Drawable** (/reference/android/graphics/drawable/Drawable) **textSelectHandleRight)**<br><br>Sets the Drawable corresponding to the right handle used for selecting text. |
| **void** | **setTextSelectHandleRight** (/reference/android/widget/TextView#setTextSelectHandleRight(int))**(int textSelectHandleRight)**<br><br>Sets the Drawable corresponding to the right handle used for selecting text. |
| **void** | **setTextSize** (/reference/android/widget/TextView#setTextSize(int,%20float))**(int unit, float size)**<br><br>Set the default text size to a given unit and value. |
| **void** | **setTextSize** (/reference/android/widget/TextView#setTextSize(float))**(float size)**<br><br>Set the default text size to the given value, interpreted as "scaled pixel" units. |
| **final void** | **setTransformationMethod** (/reference/android/widget/TextView#setTransformationMethod(android.text.method.TransformationMethod))<br>**(TransformationMethod** (/reference/android/text/method/TransformationMethod) **method)**<br><br>Sets the transformation that is applied to the text that this TextView is displaying. |
| **void** | **setTypeface** (/reference/android/widget/TextView#setTypeface(android.graphics.Typeface))**(Typeface** (/reference/android/graphics/Typeface) **tf)**<br><br>Sets the typeface and style in which the text should be displayed. |

| | |
|---|---|
| void | **setTypeface** (/reference/android/widget/TextView#setTypeface(android.graphics.Typeface,%20int))(**Typeface** (/reference/android/graphics/Typeface) **tf, int style**)<br><br>Sets the typeface and style in which the text should be displayed, and turns on the fake bold and italic bits in the Paint if the Typeface that you provided does not have all the bits in the style that you specified. |
| void | **setWidth** (/reference/android/widget/TextView#setWidth(int))(**int pixels**)<br><br>Sets the width of the TextView to be exactly **pixels** wide. |
| boolean | **showContextMenu** (/reference/android/widget/TextView#showContextMenu())()<br><br>Shows the context menu for this view. |
| boolean | **showContextMenu** (/reference/android/widget/TextView#showContextMenu(float,%20float))(**float x, float y**)<br><br>Shows the context menu for this view anchored to the specified view-relative coordinate. |

## Protected methods

| | |
|---|---|
| int | **computeHorizontalScrollRange** (/reference/android/widget/TextView#computeHorizontalScrollRange())()<br><br>Compute the horizontal range that the horizontal scrollbar represents. |
| int | **computeVerticalScrollExtent** (/reference/android/widget/TextView#computeVerticalScrollExtent())()<br><br>Compute the vertical extent of the vertical scrollbar's thumb within the vertical range. |
| int | **computeVerticalScrollRange** (/reference/android/widget/TextView#computeVerticalScrollRange())()<br><br>Compute the vertical range that the vertical scrollbar represents. |

| | |
|---|---|
| `void` | `drawableStateChanged` (/reference/android/widget/TextView#drawableStateChanged())() <br> This function is called whenever the state of the view changes in such a way that it impacts the state of drawables being sh |
| `int` | `getBottomPaddingOffset` (/reference/android/widget/TextView#getBottomPaddingOffset())() <br> Amount by which to extend the bottom fading region. |
| `boolean` | `getDefaultEditable` (/reference/android/widget/TextView#getDefaultEditable())() <br> Subclasses override this to specify that they have a KeyListener by default even if not specifically called for in the XML optio |
| `MovementMethod` (/reference/android/text/method/MovementMethod) | `getDefaultMovementMethod` (/reference/android/widget/TextView#getDefaultMovementMethod())() <br> Subclasses override this to specify a default movement method. |
| `float` | `getLeftFadingEdgeStrength` (/reference/android/widget/TextView#getLeftFadingEdgeStrength())() <br> Returns the strength, or intensity, of the left faded edge. |
| `int` | `getLeftPaddingOffset` (/reference/android/widget/TextView#getLeftPaddingOffset())() <br> Amount by which to extend the left fading region. |
| `float` | `getRightFadingEdgeStrength` (/reference/android/widget/TextView#getRightFadingEdgeStrength())() <br> Returns the strength, or intensity, of the right faded edge. |
| `int` | `getRightPaddingOffset` (/reference/android/widget/TextView#getRightPaddingOffset())() <br> Amount by which to extend the right fading region. |
| `int` | `getTopPaddingOffset` (/reference/android/widget/TextView#getTopPaddingOffset())() <br> Amount by which to extend the top fading region. |

| | |
|---|---|
| `boolean` | **`isPaddingOffsetRequired`** (/reference/android/widget/TextView#isPaddingOffsetRequired())**`()`**<br><br>If the View draws content inside its padding and enables fading edges, it needs to support padding offsets. |
| `void` | **`onAttachedToWindow`** (/reference/android/widget/TextView#onAttachedToWindow())**`()`**<br><br>This is called when the view is attached to a window. |
| `void` | **`onConfigurationChanged`** (/reference/android/widget/TextView#onConfigurationChanged(android.content.res.Configur**`(Configuration`** (/reference/android/content/res/Configuration) **`newConfig)`**<br><br>Called when the current configuration of the resources being used by the application have changed. |
| `void` | **`onCreateContextMenu`** (/reference/android/widget/TextView#onCreateContextMenu(android.view.ContextMenu))**`(ContextMenu`** (/reference/android/view/ContextMenu) **`menu)`**<br><br>Views should implement this if the view itself is going to add items to the context menu. |
| `int[]` | **`onCreateDrawableState`** (/reference/android/widget/TextView#onCreateDrawableState(int))**`(int extraSpace)`**<br><br>Generate the new **`Drawable`** (/reference/android/graphics/drawable/Drawable) state for this view. |
| `void` | **`onDraw`** (/reference/android/widget/TextView#onDraw(android.graphics.Canvas))**`(Canvas`** (/reference/android/graphics/Canvas) **`canvas)`**<br><br>Implement this to do your drawing. |
| `void` | **`onFocusChanged`** (/reference/android/widget/TextView#onFocusChanged(boolean,%20int,%20android.graphics.Rect))**`(boolean focused, int direction,`** **`Rect`** (/reference/android/graphics/Rect) **`previouslyFocusedRect)`**<br><br>Called by the view system when the focus state of this view changes. |
| `void` | **`onLayout`** (/reference/android/widget/TextView#onLayout(boolean,%20int,%20int,%20int,%20int))**`(boolean changed,`** |

| | |
|---|---|
| | **left, int top, int right, int bottom)**<br><br>Called from layout when this view should assign a size and position to each of its children. |
| **void** | **onMeasure** (/reference/android/widget/TextView#onMeasure(int,%20int))**(int widthMeasureSpec, int heightMeasureSpec)**<br><br>Measure the view and its content to determine the measured width and the measured height. |
| **void** | **onScrollChanged** (/reference/android/widget/TextView#onScrollChanged(int,%20int,%20int,%20int))**(int horiz, int vert, int oldHoriz, int oldVert)**<br><br>This is called in response to an internal scroll in this view (i.e., the view scrolled its own contents). |
| **void** | **onSelectionChanged** (/reference/android/widget/TextView#onSelectionChanged(int,%20int))**(int selStart, int selEnd)**<br><br>This method is called when the selection has changed, in case any subclasses would like to know. |
| **void** | **onTextChanged** (/reference/android/widget/TextView#onTextChanged(java.lang.CharSequence,%20int,%20int,%20int))**(CharSequence** (/reference/java/lang/CharSequence) **text, int start, int lengthBefore, int lengthAfte**<br><br>This method is called when the text is changed, in case any subclasses would like to know. |
| **void** | **onVisibilityChanged** (/reference/android/widget/TextView#onVisibilityChanged(android.view.View,%20int))**(View** (/reference/android/view/View) **changedView, int visibility)**<br><br>Called when the visibility of the view or an ancestor of the view has changed. |
| **boolean** | **setFrame** (/reference/android/widget/TextView#setFrame(int,%20int,%20int,%20int))**(int l, int t, int r, int b)** |
| **boolean** | **verifyDrawable** (/reference/android/widget/TextView#verifyDrawable(android.graphics.drawable.Drawable))**(Drawable** (/reference/android/graphics/drawable/Drawable) **who)** |

If your view subclass is displaying its own Drawable objects, it should override this function and return true for any Drawable displaying.

## Inherited methods

From class **android.view.View** (/reference/android/view/View)

| | |
|---|---|
| `void` | **addChildrenForAccessibility** (/reference/android/view/View#addChildrenForAccessibility(java.util.ArrayList<android.view.View>)) (**ArrayList** (/reference/java/util/ArrayList)<**View** (/reference/android/view/View)> **outChildren**)<br><br>Adds the children of this View relevant for accessibility to the given list as output. |
| `void` | **addExtraDataToAccessibilityNodeInfo** (/reference/android/view/View#addExtraDataToAccessibilityNodeInfo(android.view.accessibility.AccessibilityNodeInfo,%20java.lang.String,%20android.os.Bundle)) (**AccessibilityNodeInfo** (/reference/android/view/accessibility/AccessibilityNodeInfo) **info**, **String** (/reference/java/lang/String) **extraDataKey**, **Bundle** (/reference/android/os/Bundle) **arguments**)<br><br>Adds extra data to an **AccessibilityNodeInfo** (/reference/android/view/accessibility/AccessibilityNodeInfo) based on an explicit request for the additional data. |
| `void` | **addFocusables** (/reference/android/view/View#addFocusables(java.util.ArrayList<android.view.View>,%20int)) (**ArrayList** (/reference/java/util/ArrayList)<**View** (/reference/android/view/View)> |

| | |
|---|---|
| | **views, int direction)**<br><br>Add any focusable views that are descendants of this view (possibly including this view if it is focusable itself) to views. |
| **void** | **addFocusables** (/reference/android/view/View#addFocusables(java.util.ArrayList<android.view.View>,%20int,%20int))<br>**(ArrayList** (/reference/java/util/ArrayList)**<View** (/reference/android/view/View)**> views, int direction, int focusableMode)**<br><br>Adds any focusable views that are descendants of this view (possibly including this view if it is focusable itself) to views. |
| **void** | **addKeyboardNavigationClusters** (/reference/android/view/View#addKeyboardNavigationClusters(java.util.Collection<android.view.View>,%20int))<br>**(Collection** (/reference/java/util/Collection)**<View** (/reference/android/view/View)**> views, int direction)**<br><br>Adds any keyboard navigation cluster roots that are descendants of this view (possibly including this view if it is a cluster root itself) to views. |
| **void** | **addOnAttachStateChangeListener** (/reference/android/view/View#addOnAttachStateChangeListener(android.view.View.OnAttachStateChangeListener))<br>**(View.OnAttachStateChangeListener** (/reference/android/view/View.OnAttachStateChangeListener) **listener)**<br><br>Add a listener for attach state changes. |
| **void** | **addOnLayoutChangeListener** |

| | |
|---|---|
| | (/reference/android/view/View#addOnLayoutChangeListener(android.view.View.OnLayoutChangeListener))<br>**(View.OnLayoutChangeListener** (/reference/android/view/View.OnLayoutChangeListener) **listener)**<br><br>Add a listener that will be called when the bounds of the view change due to layout processing. |
| **void** | **addOnUnhandledKeyEventListener** (/reference/android/view/View#addOnUnhandledKeyEventListener(android.view.View.OnUnhandledKeyEventListener))<br>**(View.OnUnhandledKeyEventListener** (/reference/android/view/View.OnUnhandledKeyEventListener) **listener)**<br><br>Adds a listener which will receive unhandled **KeyEvent** (/reference/android/view/KeyEvent)s. |
| **void** | **addTouchables** (/reference/android/view/View#addTouchables(java.util.ArrayList<android.view.View>))<br>**(ArrayList** (/reference/java/util/ArrayList)**<View** (/reference/android/view/View)**> views)**<br><br>Add any touchable views that are descendants of this view (possibly including this view if it is touchable itself) to views. |
| **ViewPropertyAnimator** (/reference/android/view/ViewPropertyAnimator) | **animate** (/reference/android/view/View#animate())**()**<br><br>This method returns a ViewPropertyAnimator object, which can be used to animate specific properties on this View. |
| **void** | **announceForAccessibility** (/reference/android/view/View#announceForAccessibility(java.lang.CharSequence)) |

| | |
|---|---|
| | **(CharSequence** (/reference/java/lang/CharSequence) **text)**<br><br>Convenience method for sending a **AccessibilityEvent#TYPE_ANNOUNCEMENT** (/reference/android/view/accessibility/AccessibilityEvent#TYPE_ANNOUNCEMENT) **AccessibilityEvent** (/reference/android/view/accessibility/AccessibilityEvent) to suggest that an accessibility service announce the specified text to its users. |
| **void** | **autofill** (/reference/android/view/View#autofill(android.view.autofill.AutofillValue)) **(AutofillValue** (/reference/android/view/autofill/AutofillValue) **value)**<br><br>Automatically fills the content of this view with the **value**. |
| **void** | **autofill** (/reference/android/view/View#autofill(android.util.SparseArray<android.view.autofill.AutofillValue>)) **(SparseArray** (/reference/android/util/SparseArray)**<AutofillValue** (/reference/android/view/autofill/AutofillValue)**> values)**<br><br>Automatically fills the content of the virtual children within this view. |
| **boolean** | **awakenScrollBars** (/reference/android/view/View#awakenScrollBars(int,%20boolean)) **(int startDelay, boolean invalidate)**<br><br>Trigger the scrollbars to draw. |
| **boolean** | **awakenScrollBars** (/reference/android/view/View#awakenScrollBars(int))**(int startDelay)**<br><br>Trigger the scrollbars to draw. |
| **boolean** | **awakenScrollBars** (/reference/android/view/View#awakenScrollBars())**()**<br><br>Trigger the scrollbars to draw. |

| | |
|---|---|
| **void** | **bringToFront** (/reference/android/view/View#bringToFront())**( )**<br><br>Change the view's z order in the tree, so it's on top of other sibling views. |
| **void** | **buildDrawingCache** (/reference/android/view/View#buildDrawingCache(boolean))**(boolean autoScale)**<br><br>*This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations,* **setLayerType(int, android.graphics.Paint)** (/reference/android/view/View#setLayerType(int,%20android.graphics.Paint)) *handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a* **Canvas** (/reference/android/graphics/Canvas) *from either a* **Bitmap** (/reference/android/graphics/Bitmap) *or* **Picture** (/reference/android/graphics/Picture) *and call* **draw(android.graphics.Canvas)** (/reference/android/view/View#draw(android.graphics.Canvas)) *on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as* **Config.HARDWARE** (/reference/android/graphics/Bitmap.Config#HARDWARE) *bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the* **PixelCopy** (/reference/android/view/PixelCopy) *API is recommended.* |
| **void** | **buildDrawingCache** (/reference/android/view/View#buildDrawingCache())**( )**<br><br>*This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases* |

| | |
|---|---|
| | *where caching layers are useful, such as for alpha animations,* **setLayerType(int, android.graphics.Paint)** (/reference/android/view/View#setLayerType(int,%20android.graphics.Paint)) *handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a* **Canvas** (/reference/android/graphics/Canvas) *from either a* **Bitmap** (/reference/android/graphics/Bitmap) *or* **Picture** (/reference/android/graphics/Picture) *and call* **draw(android.graphics.Canvas)** (/reference/android/view/View#draw(android.graphics.Canvas)) *on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as* **Config.HARDWARE** (/reference/android/graphics/Bitmap.Config#HARDWARE) *bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the* **PixelCopy** (/reference/android/view/PixelCopy) *API is recommended.* |
| **void** | **buildLayer** (/reference/android/view/View#buildLayer())**()**<br><br>Forces this view's layer to be created and this view to be rendered into its layer. |
| **boolean** | **callOnClick** (/reference/android/view/View#callOnClick())**()**<br><br>Directly call any attached OnClickListener. |
| **boolean** | **canResolveLayoutDirection** (/reference/android/view/View#canResolveLayoutDirection())**()**<br><br>Check if layout direction resolution can be done. |
| **boolean** | **canResolveTextAlignment** (/reference/android/view/View#canResolveTextAlignment())**()**<br><br>Check if text alignment resolution can be done. |

| | |
|---|---|
| `boolean` | **canResolveTextDirection** (/reference/android/view/View#canResolveTextDirection())() <br><br> Check if text direction resolution can be done. |
| `boolean` | **canScrollHorizontally** (/reference/android/view/View#canScrollHorizontally(int))(int direction) <br><br> Check if this view can be scrolled horizontally in a certain direction. |
| `boolean` | **canScrollVertically** (/reference/android/view/View#canScrollVertically(int))(int direction) <br><br> Check if this view can be scrolled vertically in a certain direction. |
| `final void` | **cancelDragAndDrop** (/reference/android/view/View#cancelDragAndDrop())() <br><br> Cancels an ongoing drag and drop operation. |
| `void` | **cancelLongPress** (/reference/android/view/View#cancelLongPress())() <br><br> Cancels a pending long press. |
| `final void` | **cancelPendingInputEvents** (/reference/android/view/View#cancelPendingInputEvents())() <br><br> Cancel any deferred high-level input events that were previously posted to the event queue. |
| `boolean` | **checkInputConnectionProxy** (/reference/android/view/View#checkInputConnectionProxy(android.view.View))(**View** (/reference/android/view/View) view) <br><br> Called by the **InputMethodManager** |

| | |
|---|---|
| | (/reference/android/view/inputmethod/InputMethodManager) when a view who is not the current input connection target is trying to make a call on the manager. |
| `void` | **clearAnimation** (/reference/android/view/View#clearAnimation())() <br><br> Cancels any animations for this view. |
| `void` | **clearFocus** (/reference/android/view/View#clearFocus())() <br><br> Called when this view wants to give up focus. |
| `static int` | **combineMeasuredStates** (/reference/android/view/View#combineMeasuredStates(int,%20int))(`int curState, int newState`) <br><br> Merge two states as returned by **getMeasuredState()** (/reference/android/view/View#getMeasuredState()). |
| `int` | **computeHorizontalScrollExtent** (/reference/android/view/View#computeHorizontalScrollExtent())() <br><br> Compute the horizontal extent of the horizontal scrollbar's thumb within the horizontal range. |
| `int` | **computeHorizontalScrollOffset** (/reference/android/view/View#computeHorizontalScrollOffset())() <br><br> Compute the horizontal offset of the horizontal scrollbar's thumb within the horizontal range. |
| `int` | **computeHorizontalScrollRange** (/reference/android/view/View#computeHorizontalScrollRange())() <br><br> Compute the horizontal range that the horizontal scrollbar represents. |

| | |
|---|---|
| `void` | **computeScroll** (/reference/android/view/View#computeScroll())() <br><br> Called by a parent to request that a child update its values for mScrollX and mScrollY if necessary. |
| **WindowInsets** (/reference/android/view/WindowInsets) | **computeSystemWindowInsets** (/reference/android/view/View#computeSystemWindowInsets(android.view.WindowInsets,%20android.graphics.Rect)) (**WindowInsets** (/reference/android/view/WindowInsets) `in`, **Rect** (/reference/android/graphics/Rect) `outLocalInsets`) <br><br> Compute insets that should be consumed by this view and the ones that should propagate to those under it. |
| `int` | **computeVerticalScrollExtent** (/reference/android/view/View#computeVerticalScrollExtent())() <br><br> Compute the vertical extent of the vertical scrollbar's thumb within the vertical range. |
| `int` | **computeVerticalScrollOffset** (/reference/android/view/View#computeVerticalScrollOffset())() <br><br> Compute the vertical offset of the vertical scrollbar's thumb within the horizontal range. |
| `int` | **computeVerticalScrollRange** (/reference/android/view/View#computeVerticalScrollRange())() <br><br> Compute the vertical range that the vertical scrollbar represents. |
| **AccessibilityNodeInfo** (/reference/android/view/accessibility/AccessibilityNodeInfo) | **createAccessibilityNodeInfo** (/reference/android/view/View#createAccessibilityNodeInfo())() |

| | |
|---|---|
| | Returns an **AccessibilityNodeInfo** (/reference/android/view/accessibility/AccessibilityNodeInfo) representing this view from the point of view of an **AccessibilityService** (/reference/android/accessibilityservice/AccessibilityService). |
| **void** | **createContextMenu** (/reference/android/view/View#createContextMenu(android.view.ContextMenu))**(ContextMenu** (/reference/android/view/ContextMenu) **menu)**<br><br>Show the context menu for this view. |
| **void** | **destroyDrawingCache** (/reference/android/view/View#destroyDrawingCache())**( )**<br><br>*This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations,* **setLayerType(int, android.graphics.Paint)** (/reference/android/view/View#setLayerType(int,%20android.graphics.Paint)) *handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a* **Canvas** (/reference/android/graphics/Canvas) *from either a* **Bitmap** (/reference/android/graphics/Bitmap) *or* **Picture** (/reference/android/graphics/Picture) *and call* **draw(android.graphics.Canvas)** (/reference/android/view/View#draw(android.graphics.Canvas)) *on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as* **Config.HARDWARE** (/reference/android/graphics/Bitmap.Config#HARDWARE) *bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the* **PixelCopy** (/reference/android/view/PixelCopy) *API is recommended.* |

| | |
|---|---|
| **WindowInsets** (/reference/android/view/WindowInsets) | **dispatchApplyWindowInsets** (/reference/android/view/View#dispatchApplyWindowInsets(android.view.WindowInsets)) **(WindowInsets** (/reference/android/view/WindowInsets) **insets)**<br><br>Request to apply the given window insets to this view or another view in its subtree. |
| **boolean** | **dispatchCapturedPointerEvent** (/reference/android/view/View#dispatchCapturedPointerEvent(android.view.MotionEvent)) **(MotionEvent** (/reference/android/view/MotionEvent) **event)**<br><br>Pass a captured pointer event down to the focused view. |
| **void** | **dispatchConfigurationChanged** (/reference/android/view/View#dispatchConfigurationChanged(android.content.res.Configuration)) **(Configuration** (/reference/android/content/res/Configuration) **newConfig)**<br><br>Dispatch a notification about a resource configuration change down the view hierarchy. |
| **void** | **dispatchDisplayHint** (/reference/android/view/View#dispatchDisplayHint(int))**(int hint)**<br><br>Dispatch a hint about whether this view is displayed. |
| **boolean** | **dispatchDragEvent** (/reference/android/view/View#dispatchDragEvent(android.view.DragEvent)) **(DragEvent** (/reference/android/view/DragEvent) **event)**<br><br>Detects if this View is enabled and has a drag event listener. |

| | |
|---|---|
| `void` | **dispatchDraw** (/reference/android/view/View#dispatchDraw(android.graphics.Canvas))**(Canvas** (/reference/android/graphics/Canvas) **canvas)**<br><br>Called by draw to draw the child views. |
| `void` | **dispatchDrawableHotspotChanged** (/reference/android/view/View#dispatchDrawableHotspotChanged(float,%20float))<br>**(float x, float y)**<br><br>Dispatches drawableHotspotChanged to all of this View's children. |
| `void` | **dispatchFinishTemporaryDetach** (/reference/android/view/View#dispatchFinishTemporaryDetach())**()**<br><br>Dispatch **onFinishTemporaryDetach()** (/reference/android/view/View#onFinishTemporaryDetach()) to this View and its direct children if this is a container View. |
| `boolean` | **dispatchGenericFocusedEvent** (/reference/android/view/View#dispatchGenericFocusedEvent(android.view.MotionEvent))<br>**(MotionEvent** (/reference/android/view/MotionEvent) **event)**<br><br>Dispatch a generic motion event to the currently focused view. |
| `boolean` | **dispatchGenericMotionEvent** (/reference/android/view/View#dispatchGenericMotionEvent(android.view.MotionEvent))<br>**(MotionEvent** (/reference/android/view/MotionEvent) **event)**<br><br>Dispatch a generic motion event. |

| | |
|---|---|
| **boolean** | **dispatchGenericPointerEvent** (/reference/android/view/View#dispatchGenericPointerEvent(android.view.MotionEvent)) (**MotionEvent** (/reference/android/view/MotionEvent) **event**)<br><br>Dispatch a generic motion event to the view under the first pointer. |
| **boolean** | **dispatchHoverEvent** (/reference/android/view/View#dispatchHoverEvent(android.view.MotionEvent)) (**MotionEvent** (/reference/android/view/MotionEvent) **event**)<br><br>Dispatch a hover event. |
| **boolean** | **dispatchKeyEvent** (/reference/android/view/View#dispatchKeyEvent(android.view.KeyEvent))(**KeyEvent** (/reference/android/view/KeyEvent) **event**)<br><br>Dispatch a key event to the next view on the focus path. |
| **boolean** | **dispatchKeyEventPreIme** (/reference/android/view/View#dispatchKeyEventPreIme(android.view.KeyEvent)) (**KeyEvent** (/reference/android/view/KeyEvent) **event**)<br><br>Dispatch a key event before it is processed by any input method associated with the view hierarchy. |
| **boolean** | **dispatchKeyShortcutEvent** (/reference/android/view/View#dispatchKeyShortcutEvent(android.view.KeyEvent)) (**KeyEvent** (/reference/android/view/KeyEvent) **event**)<br><br>Dispatches a key shortcut event. |
| **boolean** | **dispatchNestedFling** (/reference/android/view/View#dispatchNestedFling(float,%20float,%20boolean)) |

| | |
|---|---|
| | **(float velocityX, float velocityY, boolean consumed)**<br><br>Dispatch a fling to a nested scrolling parent. |
| **boolean** | **dispatchNestedPreFling** (/reference/android/view/View#dispatchNestedPreFling(float,%20float))**(float velocityX, float velocityY)**<br><br>Dispatch a fling to a nested scrolling parent before it is processed by this view. |
| **boolean** | **dispatchNestedPrePerformAccessibilityAction** (/reference/android/view/View#dispatchNestedPrePerformAccessibilityAction(int,%20android.os.Bundle))<br>**(int action, Bundle** (/reference/android/os/Bundle) **arguments)**<br><br>Report an accessibility action to this view's parents for delegated processing. |
| **boolean** | **dispatchNestedPreScroll** (/reference/android/view/View#dispatchNestedPreScroll(int,%20int,%20int[],%20int[]))<br>**(int dx, int dy, int[] consumed, int[] offsetInWindow)**<br><br>Dispatch one step of a nested scroll in progress before this view consumes any portion of it. |
| **boolean** | **dispatchNestedScroll** (/reference/android/view/View#dispatchNestedScroll(int,%20int,%20int,%20int,%20int[]))<br>**(int dxConsumed, int dyConsumed, int dxUnconsumed, int dyUnconsumed, int[] offsetInWindow)**<br><br>Dispatch one step of a nested scroll in progress. |
| **void** | **dispatchPointerCaptureChanged** (/reference/android/view/View#dispatchPointerCaptureChanged(boolean))**(boolean** |

| | |
|---|---|
| | **hasCapture)** |
| **boolean** | **dispatchPopulateAccessibilityEvent** (/reference/android/view/View#dispatchPopulateAccessibilityEvent(android.view.accessibility.AccessibilityEvent)) **(AccessibilityEvent** (/reference/android/view/accessibility/AccessibilityEvent) **event)**<br><br>Dispatches an **AccessibilityEvent** (/reference/android/view/accessibility/AccessibilityEvent) to the **View** (/reference/android/view/View) first and then to its children for adding their text content to the event. |
| **void** | **dispatchProvideAutofillStructure** (/reference/android/view/View#dispatchProvideAutofillStructure(android.view.ViewStructure,%20int)) **(ViewStructure** (/reference/android/view/ViewStructure) **structure, int flags)**<br><br>Dispatches creation of a **ViewStructure** (/reference/android/view/ViewStructure)s for autofill purposes down the hierarchy, when an Assist structure is being created as part of an autofill request. |
| **void** | **dispatchProvideStructure** (/reference/android/view/View#dispatchProvideStructure(android.view.ViewStructure)) **(ViewStructure** (/reference/android/view/ViewStructure) **structure)**<br><br>Dispatch creation of **ViewStructure** (/reference/android/view/ViewStructure) down the hierarchy. |
| **void** | **dispatchRestoreInstanceState** |

| | |
|---|---|
| | (/reference/android/view/View#dispatchRestoreInstanceState(android.util.SparseArray<android.os.Parcelable>))<br>**(SparseArray (/reference/android/util/SparseArray)<Parcelable (/reference/android/os/Parcelable)> container)**<br><br>Called by **restoreHierarchyState(android.util.SparseArray)** (/reference/android/view/View#restoreHierarchyState(android.util.SparseArray<android.os.Parcelable>))<br>to retrieve the state for this view and its children. |
| **void** | **dispatchSaveInstanceState** (/reference/android/view/View#dispatchSaveInstanceState(android.util.SparseArray<android.os.Parcelable>))<br>**(SparseArray (/reference/android/util/SparseArray)<Parcelable (/reference/android/os/Parcelable)> container)**<br><br>Called by **saveHierarchyState(android.util.SparseArray)** (/reference/android/view/View#saveHierarchyState(android.util.SparseArray<android.os.Parcelable>))<br>to store the state for this view and its children. |
| **void** | **dispatchSetActivated** (/reference/android/view/View#dispatchSetActivated(boolean))**(boolean activated)**<br><br>Dispatch setActivated to all of this View's children. |
| **void** | **dispatchSetPressed** (/reference/android/view/View#dispatchSetPressed(boolean))<br>**(boolean pressed)**<br><br>Dispatch setPressed to all of this View's children. |
| **void** | **dispatchSetSelected** (/reference/android/view/View#dispatchSetSelected(boolean)) |

| | |
|---|---|
| | **(boolean selected)**<br><br>Dispatch setSelected to all of this View's children. |
| **void** | **dispatchStartTemporaryDetach** (/reference/android/view/View#dispatchStartTemporaryDetach())**()**<br><br>Dispatch **onStartTemporaryDetach()** (/reference/android/view/View#onStartTemporaryDetach()) to this View and its direct children if this is a container View. |
| **void** | **dispatchSystemUiVisibilityChanged** (/reference/android/view/View#dispatchSystemUiVisibilityChanged(int))**(int visibility)**<br><br>Dispatch callbacks to **setOnSystemUiVisibilityChangeListener(View.OnSystemUiVisibilityChangeListener)** (/reference/android/view/View#setOnSystemUiVisibilityChangeListener(android.view.View.OnSystemUiVisibilityChangeListener))<br>down the view hierarchy. |
| **boolean** | **dispatchTouchEvent** (/reference/android/view/View#dispatchTouchEvent(android.view.MotionEvent))<br>**(MotionEvent** (/reference/android/view/MotionEvent) **event)**<br><br>Pass the touch screen motion event down to the target view, or this view if it is the target. |
| **boolean** | **dispatchTrackballEvent** (/reference/android/view/View#dispatchTrackballEvent(android.view.MotionEvent))<br>**(MotionEvent** (/reference/android/view/MotionEvent) **event)** |

| | |
|---|---|
| | Pass a trackball motion event down to the focused view. |
| **boolean** | **dispatchUnhandledMove** (/reference/android/view/View#dispatchUnhandledMove(android.view.View,%20int)) **(View** (/reference/android/view/View) **focused, int direction)**<br><br>This method is the last chance for the focused view and its ancestors to respond to an arrow key. |
| **void** | **dispatchVisibilityChanged** (/reference/android/view/View#dispatchVisibilityChanged(android.view.View,%20int)) **(View** (/reference/android/view/View) **changedView, int visibility)**<br><br>Dispatch a view visibility change down the view hierarchy. |
| **void** | **dispatchWindowFocusChanged** (/reference/android/view/View#dispatchWindowFocusChanged(boolean))**(boolean hasFocus)**<br><br>Called when the window containing this view gains or loses window focus. |
| void | dispatchWindowInsetsAnimationFinish (/reference/android/view/View#dispatchWindowInsetsAnimationFinish(android.view.WindowInsetsAnimationCallback.InsetsAnimation)) (WindowInsetsAnimationCallback.InsetsAnimation (/reference/android/view/WindowInsetsAnimationCallback.InsetsAnimation) animation)<br><br>Dispatches **WindowInsetsAnimationCallback#onFinish(InsetsAnimation)** (/reference/android/view/WindowInsetsAnimationCallback#onFinish(android.view.WindowInsetsAnimationCallback.InsetsAnimation)) when Window Insets animation finishes. |

| | |
|---|---|
| void | dispatchWindowInsetsAnimationPrepare (/reference/android/view/View#dispatchWindowInsetsAnimationPrepare(android.view.WindowInsetsAnimationCallback.InsetsAnimation)) (WindowInsetsAnimationCallback.InsetsAnimation (/reference/android/view/WindowInsetsAnimationCallback.InsetsAnimation) animation)<br><br>Dispatches **WindowInsetsAnimationCallback#onPrepare(InsetsAnimation)** (/reference/android/view/WindowInsetsAnimationCallback#onPrepare(android.view.WindowInsetsAnimationCallback.InsetsAnimation)) when Window Insets animation is being prepared. |
| WindowInsets (/reference/android/view/WindowInsets) | dispatchWindowInsetsAnimationProgress (/reference/android/view/View#dispatchWindowInsetsAnimationProgress(android.view.WindowInsets)) (WindowInsets (/reference/android/view/WindowInsets) insets)<br><br>Dispatches **WindowInsetsAnimationCallback#onProgress(WindowInsets)** (/reference/android/view/WindowInsetsAnimationCallback#onProgress(android.view.WindowInsets)) when Window Insets animation makes progress. |
| WindowInsetsAnimationCallback.AnimationBounds (/reference/android/view/WindowInsetsAnimationCallback.AnimationBounds) | dispatchWindowInsetsAnimationStart (/reference/android/view/View#dispatchWindowInsetsAnimationStart(android.view.WindowInsetsAnimationCallback.InsetsAnimation,%20android.view.WindowInsetsAnimationCallback.AnimationBounds)) (WindowInsetsAnimationCallback.InsetsAnimation (/reference/android/view/WindowInsetsAnimationCallback.InsetsAnimation) animation, WindowInsetsAnimationCallback.AnimationBounds (/reference/android/view/WindowInsetsAnimationCallback.AnimationBounds) bounds)<br><br>Dispatches **WindowInsetsAnimationCallback#onStart(InsetsAnimation,** |

| | |
|---|---|
| | **AnimationBounds)** (/reference/android/view/WindowInsetsAnimationCallback#onStart(android.view.WindowInsetsAnimationCallback.InsetsAnimation,%20android.view.WindowInsetsAnimationCallback.AnimationBounds)) when Window Insets animation is started. |
| **void** | **dispatchWindowSystemUiVisiblityChanged** (/reference/android/view/View#dispatchWindowSystemUiVisiblityChanged(int))**(int visible)**<br><br>Dispatch callbacks to **onWindowSystemUiVisibilityChanged(int)** (/reference/android/view/View#onWindowSystemUiVisibilityChanged(int)) down the view hierarchy. |
| **void** | **dispatchWindowVisibilityChanged** (/reference/android/view/View#dispatchWindowVisibilityChanged(int))**(int visibility)**<br><br>Dispatch a window visibility change down the view hierarchy. |
| **void** | **draw** (/reference/android/view/View#draw(android.graphics.Canvas))**(**Canvas (/reference/android/graphics/Canvas) **canvas)**<br><br>Manually render this view (and all of its children) to the given Canvas. |
| **void** | **drawableHotspotChanged** (/reference/android/view/View#drawableHotspotChanged(float,%20float))**(float x, float y)**<br><br>This function is called whenever the view hotspot changes and needs to be propagated to drawables or child views managed by the view. |
| **void** | **drawableStateChanged** (/reference/android/view/View#drawableStateChanged())**()** |

| | |
|---|---|
| | This function is called whenever the state of the view changes in such a way that it impacts the state of drawables being shown. |
| View (/reference/android/view/View) | findFocus (/reference/android/view/View#findFocus())() <br><br> Find the view in the hierarchy rooted at this view that currently has focus. |
| final <T extends View (/reference/android/view/View)> T | findViewById (/reference/android/view/View#findViewById(int))(int id) <br><br> Finds the first descendant view with the given ID, the view itself if the ID matches getId() (/reference/android/view/View#getId()), or null if the ID is invalid (< 0) or there is no matching view in the hierarchy. |
| final <T extends View (/reference/android/view/View)> T | findViewWithTag (/reference/android/view/View#findViewWithTag(java.lang.Object))(Object (/reference/java/lang/Object) tag) <br><br> Look for a child view with the given tag. |
| void | findViewsWithText (/reference/android/view/View#findViewsWithText(java.util.ArrayList<android.view.View>,%20java.lang.CharSequence,%20int))(ArrayList (/reference/java/util/ArrayList)<View (/reference/android/view/View)> outViews, CharSequence (/reference/java/lang/CharSequence) searched, int flags) <br><br> Finds the Views that contain given text. |
| boolean | fitSystemWindows (/reference/android/view/View#fitSystemWindows(android.graphics.Rect))(Rect (/reference/android/graphics/Rect) insets) <br><br> *This method was deprecated in API level 20. As of API 20 use* dispatchApplyWindowInsets(android.view.WindowInsets) |

| | |
|---|---|
| | (/reference/android/view/View#dispatchApplyWindowInsets(android.view.WindowInsets)) *to apply insets to views. Views should override* **onApplyWindowInsets(android.view.WindowInsets)** (/reference/android/view/View#onApplyWindowInsets(android.view.WindowInsets)) *or use* **setOnApplyWindowInsetsListener(android.view.View.OnApplyWindowInsetsListener)** (/reference/android/view/View#setOnApplyWindowInsetsListener(android.view.View.OnApplyWindowInsetsListener)) *to implement handling their own insets.* |
| **View** (/reference/android/view/View) | **focusSearch** (/reference/android/view/View#focusSearch(int))**(int direction)**<br><br>Find the nearest view in the specified direction that can take focus. |
| **void** | **forceHasOverlappingRendering** (/reference/android/view/View#forceHasOverlappingRendering(boolean))**(boolean hasOverlappingRendering)**<br><br>Sets the behavior for overlapping rendering for this view (see **hasOverlappingRendering()** (/reference/android/view/View#hasOverlappingRendering()) for more details on this behavior). |
| **void** | **forceLayout** (/reference/android/view/View#forceLayout())**()**<br><br>Forces this view to be laid out during the next layout pass. |
| **static int** | **generateViewId** (/reference/android/view/View#generateViewId())**()**<br><br>Generate a value suitable for use in **setId(int)** (/reference/android/view/View#setId(int)). |

| | |
|---|---|
| CharSequence (/reference/java/lang/CharSequence) | **getAccessibilityClassName** (/reference/android/view/View#getAccessibilityClassName())() <br><br> Return the class name of this object to be used for accessibility purposes. |
| View.AccessibilityDelegate (/reference/android/view/View.AccessibilityDelegate) | **getAccessibilityDelegate** (/reference/android/view/View#getAccessibilityDelegate())() <br><br> Returns the delegate for implementing accessibility support via composition. |
| int | **getAccessibilityLiveRegion** (/reference/android/view/View#getAccessibilityLiveRegion())() <br><br> Gets the live region mode for this View. |
| AccessibilityNodeProvider (/reference/android/view/accessibility/AccessibilityNodeProvider) | **getAccessibilityNodeProvider** (/reference/android/view/View#getAccessibilityNodeProvider())() <br><br> Gets the provider for managing a virtual view hierarchy rooted at this View and reported to AccessibilityService (/reference/android/accessibilityservice/AccessibilityService)s that explore the window content. |
| CharSequence (/reference/java/lang/CharSequence) | **getAccessibilityPaneTitle** (/reference/android/view/View#getAccessibilityPaneTitle())() <br><br> Get the title of the pane for purposes of accessibility. |
| int | **getAccessibilityTraversalAfter** (/reference/android/view/View#getAccessibilityTraversalAfter())() <br><br> Gets the id of a view after which this one is visited in accessibility traversal. |
| int | **getAccessibilityTraversalBefore** |

| | |
|---|---|
| | (/reference/android/view/View#getAccessibilityTraversalBefore())() <br><br> Gets the id of a view before which this one is visited in accessibility traversal. |
| `float` | **getAlpha** (/reference/android/view/View#getAlpha())() <br><br> The opacity of the view. |
| **Animation** (/reference/android/view/animation/Animation) | **getAnimation** (/reference/android/view/View#getAnimation())() <br><br> Get the animation currently associated with this view. |
| **Matrix** (/reference/android/graphics/Matrix) | **getAnimationMatrix** (/reference/android/view/View#getAnimationMatrix())() <br><br> Return the current transformation matrix of the view. |
| **IBinder** (/reference/android/os/IBinder) | **getApplicationWindowToken** (/reference/android/view/View#getApplicationWindowToken())() <br><br> Retrieve a unique token identifying the top-level "real" window of the window that this view is attached to. |
| `int[]` | **getAttributeResolutionStack** (/reference/android/view/View#getAttributeResolutionStack(int))(`int attribute`) <br><br> Returns the ordered list of resource ID that are considered when resolving attribute values for this **View** (/reference/android/view/View). |
| **Map** (/reference/java/util/Map)<**Integer** (/reference/java/lang/Integer), **Integer** (/reference/java/lang/Integer)> | **getAttributeSourceResourceMap** (/reference/android/view/View#getAttributeSourceResourceMap())() <br><br> Returns the mapping of attribute resource ID to source resource ID where the attribute value was set. |

| | |
|---|---|
| **String[]** (/reference/java/lang/String) | **getAutofillHints** (/reference/android/view/View#getAutofillHints())()<br><br>Gets the hints that help an **AutofillService** (/reference/android/service/autofill/AutofillService) determine how to autofill the view with the user's data. |
| **final AutofillId** (/reference/android/view/autofill/AutofillId) | **getAutofillId** (/reference/android/view/View#getAutofillId())()<br><br>Gets the unique, logical identifier of this view in the activity, for autofill purposes. |
| **int** | **getAutofillType** (/reference/android/view/View#getAutofillType())()<br><br>Describes the autofill type of this view, so an **AutofillService** (/reference/android/service/autofill/AutofillService) can create the proper **AutofillValue** (/reference/android/view/autofill/AutofillValue) when autofilling the view. |
| **AutofillValue** (/reference/android/view/autofill/AutofillValue) | **getAutofillValue** (/reference/android/view/View#getAutofillValue())()<br><br>Gets the **View** (/reference/android/view/View)'s current autofill value. |
| **Drawable** (/reference/android/graphics/drawable/Drawable) | **getBackground** (/reference/android/view/View#getBackground())()<br><br>Gets the background drawable |
| **BlendMode** (/reference/android/graphics/BlendMode) | **getBackgroundTintBlendMode** (/reference/android/view/View#getBackgroundTintBlendMode())()<br><br>Return the blending mode used to apply the tint to the background drawable, if specified. |
| **ColorStateList** (/reference/android/content/res/ColorStateList) | **getBackgroundTintList** (/reference/android/view/View#getBackgroundTintList())()<br><br>Return the tint applied to the background drawable, if specified. |

| | |
|---|---|
| PorterDuff.Mode (/reference/android/graphics/PorterDuff.Mode) | **getBackgroundTintMode** (/reference/android/view/View#getBackgroundTintMode())() Return the blending mode used to apply the tint to the background drawable, if specified. |
| int | **getBaseline** (/reference/android/view/View#getBaseline())() Return the offset of the widget's text baseline from the widget's top boundary. |
| final int | **getBottom** (/reference/android/view/View#getBottom())() Bottom position of this view relative to its parent. |
| float | **getBottomFadingEdgeStrength** (/reference/android/view/View#getBottomFadingEdgeStrength())() Returns the strength, or intensity, of the bottom faded edge. |
| int | **getBottomPaddingOffset** (/reference/android/view/View#getBottomPaddingOffset())() Amount by which to extend the bottom fading region. |
| float | **getCameraDistance** (/reference/android/view/View#getCameraDistance())() Gets the distance along the Z axis from the camera to this view. |
| boolean | **getClipBounds** (/reference/android/view/View#getClipBounds(android.graphics.Rect))(**Rect** (/reference/android/graphics/Rect) **outRect**) Populates an output rectangle with the clip bounds of the view, returning **true** if successful or **false** if the view's clip bounds are **null**. |
| Rect (/reference/android/graphics/Rect) | **getClipBounds** (/reference/android/view/View#getClipBounds())() |

| | |
|---|---|
| | Returns a copy of the current **clipBounds** (/reference/android/view/View#setClipBounds(android.graphics.Rect)). |
| **final boolean** | **getClipToOutline** (/reference/android/view/View#getClipToOutline())() <br><br> Returns whether the Outline should be used to clip the contents of the View. |
| **final** **ContentCaptureSession** (/reference/android/view/contentcapture/ContentCaptureSession) | **getContentCaptureSession** (/reference/android/view/View#getContentCaptureSession())() <br><br> Gets the session used to notify content capture events. |
| **CharSequence** (/reference/java/lang/CharSequence) | **getContentDescription** (/reference/android/view/View#getContentDescription())() <br><br> Returns the **View** (/reference/android/view/View)'s content description. |
| **final** **Context** (/reference/android/content/Context) | **getContext** (/reference/android/view/View#getContext())() <br><br> Returns the context the view is running in, through which it can access the current theme, resources, etc. |
| **ContextMenu.ContextMenuInfo** (/reference/android/view/ContextMenu.ContextMenuInfo) | **getContextMenuInfo** (/reference/android/view/View#getContextMenuInfo())() <br><br> Views should implement this if they have extra information to associate with the context menu. |
| **final boolean** | **getDefaultFocusHighlightEnabled** (/reference/android/view/View#getDefaultFocusHighlightEnabled())() <br><br> /** Returns whether this View should use a default focus highlight when it gets focused but doesn't have **R.attr.state_focused** (/reference/android/R.attr#state_focused) defined in its background. |

| | |
|---|---|
| `static int` | **`getDefaultSize`** (/reference/android/view/View#getDefaultSize(int,%20int))**`(int size, int measureSpec)`**<br><br>Utility to return a default size. |
| **`Display`** (/reference/android/view/Display) | **`getDisplay`** (/reference/android/view/View#getDisplay())**`()`**<br><br>Gets the logical display to which the view's window has been attached. |
| `final int[]` | **`getDrawableState`** (/reference/android/view/View#getDrawableState())**`()`**<br><br>Return an array of resource IDs of the drawable states representing the current state of the view. |
| **`Bitmap`** (/reference/android/graphics/Bitmap) | **`getDrawingCache`** (/reference/android/view/View#getDrawingCache())**`()`**<br><br>*This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations,* **`setLayerType(int, android.graphics.Paint)`** (/reference/android/view/View#setLayerType(int,%20android.graphics.Paint)) *handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a* **`Canvas`** (/reference/android/graphics/Canvas) *from either a* **`Bitmap`** (/reference/android/graphics/Bitmap) *or* **`Picture`** (/reference/android/graphics/Picture) *and call* **`draw(android.graphics.Canvas)`** (/reference/android/view/View#draw(android.graphics.Canvas)) *on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as* **`Config.HARDWARE`** (/reference/android/graphics/Bitmap.Config#HARDWARE) *bitmaps, real-time shadows,* |

| | |
|---|---|
| | *and outline clipping. For screenshots of the UI for feedback reports or unit testing the* **PixelCopy** (/reference/android/view/PixelCopy) *API is recommended.* |
| **Bitmap** (/reference/android/graphics/Bitmap) | **getDrawingCache** (/reference/android/view/View#getDrawingCache(boolean)) **(boolean autoScale)**<br><br>*This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations,* **setLayerType(int, android.graphics.Paint)** (/reference/android/view/View#setLayerType(int,%20android.graphics.Paint)) *handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a* **Canvas** (/reference/android/graphics/Canvas) *from either a* **Bitmap** (/reference/android/graphics/Bitmap) *or* **Picture** (/reference/android/graphics/Picture) *and call* **draw(android.graphics.Canvas)** (/reference/android/view/View#draw(android.graphics.Canvas)) *on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as* **Config.HARDWARE** (/reference/android/graphics/Bitmap.Config#HARDWARE) *bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the* **PixelCopy** (/reference/android/view/PixelCopy) *API is recommended.* |
| **int** | **getDrawingCacheBackgroundColor** (/reference/android/view/View#getDrawingCacheBackgroundColor())**( )**<br><br>*This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases* |

*where caching layers are useful, such as for alpha animations,* **setLayerType(int, android.graphics.Paint)** (/reference/android/view/View#setLayerType(int,%20android.graphics.Paint)) *handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a* **Canvas** (/reference/android/graphics/Canvas) *from either a* **Bitmap** (/reference/android/graphics/Bitmap) *or* **Picture** (/reference/android/graphics/Picture) *and call* **draw(android.graphics.Canvas)** (/reference/android/view/View#draw(android.graphics.Canvas)) *on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as* **Config.HARDWARE** (/reference/android/graphics/Bitmap.Config#HARDWARE) *bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the* **PixelCopy** (/reference/android/view/PixelCopy) *API is recommended.*

---

`int`

**getDrawingCacheQuality** (/reference/android/view/View#getDrawingCacheQuality()) ()

*This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations,* **setLayerType(int, android.graphics.Paint)** (/reference/android/view/View#setLayerType(int,%20android.graphics.Paint)) *handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a* **Canvas** (/reference/android/graphics/Canvas) *from either a* **Bitmap** (/reference/android/graphics/Bitmap) *or* **Picture** (/reference/android/graphics/Picture) *and call* **draw(android.graphics.Canvas)** (/reference/android/view/View#draw(android.graphics.Canvas)) *on the View. However*

| | |
|---|---|
| | *these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as* **Config.HARDWARE** (/reference/android/graphics/Bitmap.Config#HARDWARE) *bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the* **PixelCopy** (/reference/android/view/PixelCopy) *API is recommended.* |
| **void** | **getDrawingRect** (/reference/android/view/View#getDrawingRect(android.graphics.Rect))**(Rect** (/reference/android/graphics/Rect) **outRect)**<br><br>Return the visible drawing bounds of your view. |
| **long** | **getDrawingTime** (/reference/android/view/View#getDrawingTime())**()**<br><br>Return the time at which the drawing of the view hierarchy started. |
| **float** | **getElevation** (/reference/android/view/View#getElevation())**()**<br><br>The base elevation of this view relative to its parent, in pixels. |
| **int** | **getExplicitStyle** (/reference/android/view/View#getExplicitStyle())**()**<br><br>Returns the resource ID for the style specified using **style="..."** in the **AttributeSet** (/reference/android/util/AttributeSet)'s backing XML element or **Resources#ID_NULL** (/reference/android/content/res/Resources#ID_NULL) otherwise if not specified or otherwise not applicable. |
| **boolean** | **getFilterTouchesWhenObscured** (/reference/android/view/View#getFilterTouchesWhenObscured())**()**<br><br>Gets whether the framework should discard touches when the view's window is obscured by another visible window. |

| | |
|---|---|
| `boolean` | **`getFitsSystemWindows`** (/reference/android/view/View#getFitsSystemWindows())**`()`**<br><br>Check for state of **`setFitsSystemWindows(boolean)`** (/reference/android/view/View#setFitsSystemWindows(boolean)). |
| `int` | **`getFocusable`** (/reference/android/view/View#getFocusable())**`()`**<br><br>Returns the focusable setting for this view. |
| **`ArrayList`** (/reference/java/util/ArrayList)**`<`**`View`** (/reference/android/view/View)**`>`** | **`getFocusables`** (/reference/android/view/View#getFocusables(int))**`(int direction)`**<br><br>Find and return all focusable views that are descendants of this view, possibly including this view if it is focusable itself. |
| `void` | **`getFocusedRect`** (/reference/android/view/View#getFocusedRect(android.graphics.Rect))**`(`**`Rect`** (/reference/android/graphics/Rect) **`r)`**<br><br>When a view has focus and the user navigates away from it, the next view is searched for starting from the rectangle filled in by this method. |
| **`Drawable`** (/reference/android/graphics/drawable/Drawable) | **`getForeground`** (/reference/android/view/View#getForeground())**`()`**<br><br>Returns the drawable used as the foreground of this View. |
| `int` | **`getForegroundGravity`** (/reference/android/view/View#getForegroundGravity())**`()`**<br><br>Describes how the foreground is positioned. |
| **`BlendMode`** (/reference/android/graphics/BlendMode) | **`getForegroundTintBlendMode`** (/reference/android/view/View#getForegroundTintBlendMode())**`()`**<br><br>Return the blending mode used to apply the tint to the foreground drawable, if specified. |

| | |
|---|---|
| ColorStateList (/reference/android/content/res/ColorStateList) | getForegroundTintList (/reference/android/view/View#getForegroundTintList())() Return the tint applied to the foreground drawable, if specified. |
| PorterDuff.Mode (/reference/android/graphics/PorterDuff.Mode) | getForegroundTintMode (/reference/android/view/View#getForegroundTintMode())() Return the blending mode used to apply the tint to the foreground drawable, if specified. |
| final boolean | getGlobalVisibleRect (/reference/android/view/View#getGlobalVisibleRect(android.graphics.Rect))(Rect (/reference/android/graphics/Rect) r) |
| boolean | getGlobalVisibleRect (/reference/android/view/View#getGlobalVisibleRect(android.graphics.Rect,%20android.graphics.Point)) (Rect (/reference/android/graphics/Rect) r, Point (/reference/android/graphics/Point) globalOffset) If some part of this view is not clipped by any of its parents, then return that area in r in global (root) coordinates. |
| Handler (/reference/android/os/Handler) | getHandler (/reference/android/view/View#getHandler())() |
| final boolean | getHasOverlappingRendering (/reference/android/view/View#getHasOverlappingRendering())() Returns the value for overlapping rendering that is used internally. |
| final int | getHeight (/reference/android/view/View#getHeight())() Return the height of your view. |

| | |
|---|---|
| `void` | **getHitRect** (/reference/android/view/View#getHitRect(android.graphics.Rect))**(Rect** (/reference/android/graphics/Rect) **outRect)**<br><br>Hit rectangle in parent's coordinates |
| `int` | **getHorizontalFadingEdgeLength** (/reference/android/view/View#getHorizontalFadingEdgeLength())**()**<br><br>Returns the size of the horizontal faded edges used to indicate that more content in this view is visible. |
| `int` | **getHorizontalScrollbarHeight** (/reference/android/view/View#getHorizontalScrollbarHeight())**()**<br><br>Returns the height of the horizontal scrollbar. |
| **Drawable** (/reference/android/graphics/drawable/Drawable) | **getHorizontalScrollbarThumbDrawable** (/reference/android/view/View#getHorizontalScrollbarThumbDrawable())**()**<br><br>Returns the currently configured Drawable for the thumb of the horizontal scroll bar if it exists, null otherwise. |
| **Drawable** (/reference/android/graphics/drawable/Drawable) | **getHorizontalScrollbarTrackDrawable** (/reference/android/view/View#getHorizontalScrollbarTrackDrawable())**()**<br><br>Returns the currently configured Drawable for the track of the horizontal scroll bar if it exists, null otherwise. |
| `int` | **getId** (/reference/android/view/View#getId())**()**<br><br>Returns this view's identifier. |
| `int` | **getImportantForAccessibility** |

| | |
|---|---|
| | (/reference/android/view/View#getImportantForAccessibility())() Gets the mode for determining whether this View is important for accessibility. |
| int | getImportantForAutofill (/reference/android/view/View#getImportantForAutofill())() Gets the mode for determining whether this view is important for autofill. |
| int | getImportantForContentCapture (/reference/android/view/View#getImportantForContentCapture())() Gets the mode for determining whether this view is important for content capture. |
| boolean | getKeepScreenOn (/reference/android/view/View#getKeepScreenOn())() Returns whether the screen should remain on, corresponding to the current value of KEEP_SCREEN_ON (/reference/android/view/View#KEEP_SCREEN_ON). |
| KeyEvent.DispatcherState (/reference/android/view/KeyEvent.DispatcherState) | getKeyDispatcherState (/reference/android/view/View#getKeyDispatcherState())() Return the global KeyEvent.DispatcherState (/reference/android/view/KeyEvent.DispatcherState) for this view's window. |
| int | getLabelFor (/reference/android/view/View#getLabelFor())() Gets the id of a view for which this view serves as a label for accessibility purposes. |
| int | getLayerType (/reference/android/view/View#getLayerType())() Indicates what type of layer is currently associated with this view. |
| int | getLayoutDirection (/reference/android/view/View#getLayoutDirection())() |

| | |
|---|---|
| | Returns the resolved layout direction for this view. |
| ViewGroup.LayoutParams (/reference/android/view/ViewGroup.LayoutParams) | getLayoutParams (/reference/android/view/View#getLayoutParams())()<br><br>Get the LayoutParams associated with this view. |
| final int | getLeft (/reference/android/view/View#getLeft())()<br><br>Left position of this view relative to its parent. |
| float | getLeftFadingEdgeStrength (/reference/android/view/View#getLeftFadingEdgeStrength())()<br><br>Returns the strength, or intensity, of the left faded edge. |
| int | getLeftPaddingOffset (/reference/android/view/View#getLeftPaddingOffset())()<br><br>Amount by which to extend the left fading region. |
| final boolean | getLocalVisibleRect (/reference/android/view/View#getLocalVisibleRect(android.graphics.Rect))(Rect (/reference/android/graphics/Rect) r) |
| void | getLocationInSurface (/reference/android/view/View#getLocationInSurface(int[]))(int[] location)<br><br>Compute the view's coordinate within the surface. |
| void | getLocationInWindow (/reference/android/view/View#getLocationInWindow(int[]))(int[] outLocation)<br><br>Computes the coordinates of this view in its window. |
| void | getLocationOnScreen (/reference/android/view/View#getLocationOnScreen(int[])) |

| | |
|---|---|
| | **`(int[] outLocation)`**<br><br>Computes the coordinates of this view on the screen. |
| **`Matrix`** (/reference/android/graphics/Matrix) | **`getMatrix`** (/reference/android/view/View#getMatrix())**`()`**<br><br>The transform matrix of this view, which is calculated based on the current rotation, scale, and pivot properties. |
| **`final int`** | **`getMeasuredHeight`** (/reference/android/view/View#getMeasuredHeight())**`()`**<br><br>Like **`getMeasuredHeightAndState()`** (/reference/android/view/View#getMeasuredHeightAndState()), but only returns the raw height component (that is the result is masked by **`MEASURED_SIZE_MASK`** (/reference/android/view/View#MEASURED_SIZE_MASK)). |
| **`final int`** | **`getMeasuredHeightAndState`** (/reference/android/view/View#getMeasuredHeightAndState())**`()`**<br><br>Return the full height measurement information for this view as computed by the most recent call to **`measure(int, int)`** (/reference/android/view/View#measure(int,%20int)). |
| **`final int`** | **`getMeasuredState`** (/reference/android/view/View#getMeasuredState())**`()`**<br><br>Return only the state bits of **`getMeasuredWidthAndState()`** (/reference/android/view/View#getMeasuredWidthAndState()) and **`getMeasuredHeightAndState()`** (/reference/android/view/View#getMeasuredHeightAndState()), combined into one integer. |
| **`final int`** | **`getMeasuredWidth`** (/reference/android/view/View#getMeasuredWidth())**`()`** |

| | |
|---|---|
| | Like **getMeasuredWidthAndState()** (/reference/android/view/View#getMeasuredWidthAndState()), but only returns the raw width component (that is the result is masked by **MEASURED_SIZE_MASK** (/reference/android/view/View#MEASURED_SIZE_MASK)). |
| `final int` | **getMeasuredWidthAndState** (/reference/android/view/View#getMeasuredWidthAndState())() Return the full width measurement information for this view as computed by the most recent call to **measure(int, int)** (/reference/android/view/View#measure(int,%20int)). |
| `int` | **getMinimumHeight** (/reference/android/view/View#getMinimumHeight())() Returns the minimum height of the view. |
| `int` | **getMinimumWidth** (/reference/android/view/View#getMinimumWidth())() Returns the minimum width of the view. |
| `int` | **getNextClusterForwardId** (/reference/android/view/View#getNextClusterForwardId())() Gets the id of the root of the next keyboard navigation cluster. |
| `int` | **getNextFocusDownId** (/reference/android/view/View#getNextFocusDownId())() Gets the id of the view to use when the next focus is **FOCUS_DOWN** (/reference/android/view/View#FOCUS_DOWN). |
| `int` | **getNextFocusForwardId** (/reference/android/view/View#getNextFocusForwardId())() Gets the id of the view to use when the next focus is **FOCUS_FORWARD** |

| | |
|---|---|
| | (/reference/android/view/View#FOCUS_FORWARD). |
| `int` | **`getNextFocusLeftId`** (/reference/android/view/View#getNextFocusLeftId())() Gets the id of the view to use when the next focus is **`FOCUS_LEFT`** (/reference/android/view/View#FOCUS_LEFT). |
| `int` | **`getNextFocusRightId`** (/reference/android/view/View#getNextFocusRightId())() Gets the id of the view to use when the next focus is **`FOCUS_RIGHT`** (/reference/android/view/View#FOCUS_RIGHT). |
| `int` | **`getNextFocusUpId`** (/reference/android/view/View#getNextFocusUpId())() Gets the id of the view to use when the next focus is **`FOCUS_UP`** (/reference/android/view/View#FOCUS_UP). |
| **`View.OnFocusChangeListener`** (/reference/android/view/View.OnFocusChangeListener) | **`getOnFocusChangeListener`** (/reference/android/view/View#getOnFocusChangeListener())() Returns the focus-change callback registered for this view. |
| `int` | **`getOutlineAmbientShadowColor`** (/reference/android/view/View#getOutlineAmbientShadowColor())() |
| **`ViewOutlineProvider`** (/reference/android/view/ViewOutlineProvider) | **`getOutlineProvider`** (/reference/android/view/View#getOutlineProvider())() Returns the current **`ViewOutlineProvider`** (/reference/android/view/ViewOutlineProvider) of the view, which generates the Outline that defines the shape of the shadow it casts, and enables outline clipping. |
| `int` | **`getOutlineSpotShadowColor`** (/reference/android/view/View#getOutlineSpotShadowColor())() |

| | |
|---|---|
| `int` | **`getOverScrollMode`** (/reference/android/view/View#getOverScrollMode())**`()`**<br>Returns the over-scroll mode for this view. |
| **`ViewOverlay`** (/reference/android/view/ViewOverlay) | **`getOverlay`** (/reference/android/view/View#getOverlay())**`()`**<br>Returns the overlay for this view, creating it if it does not yet exist. |
| `int` | **`getPaddingBottom`** (/reference/android/view/View#getPaddingBottom())**`()`**<br>Returns the bottom padding of this view. |
| `int` | **`getPaddingEnd`** (/reference/android/view/View#getPaddingEnd())**`()`**<br>Returns the end padding of this view depending on its resolved layout direction. |
| `int` | **`getPaddingLeft`** (/reference/android/view/View#getPaddingLeft())**`()`**<br>Returns the left padding of this view. |
| `int` | **`getPaddingRight`** (/reference/android/view/View#getPaddingRight())**`()`**<br>Returns the right padding of this view. |
| `int` | **`getPaddingStart`** (/reference/android/view/View#getPaddingStart())**`()`**<br>Returns the start padding of this view depending on its resolved layout direction. |
| `int` | **`getPaddingTop`** (/reference/android/view/View#getPaddingTop())**`()`**<br>Returns the top padding of this view. |
| `final` **`ViewParent`** (/reference/android/view/ViewParent) | **`getParent`** (/reference/android/view/View#getParent())**`()`**<br>Gets the parent of this view. |

| | |
|---|---|
| ViewParent (/reference/android/view/ViewParent) | getParentForAccessibility (/reference/android/view/View#getParentForAccessibility())() Gets the parent for accessibility purposes. |
| float | getPivotX (/reference/android/view/View#getPivotX())() The x location of the point around which the view is rotated (/reference/android/view/View#setRotation(float)) and scaled (/reference/android/view/View#setScaleX(float)). |
| float | getPivotY (/reference/android/view/View#getPivotY())() The y location of the point around which the view is rotated (/reference/android/view/View#setRotation(float)) and scaled (/reference/android/view/View#setScaleY(float)). |
| PointerIcon (/reference/android/view/PointerIcon) | getPointerIcon (/reference/android/view/View#getPointerIcon())() Gets the pointer icon for the current view. |
| Resources (/reference/android/content/res/Resources) | getResources (/reference/android/view/View#getResources())() Returns the resources associated with this view. |
| final boolean | getRevealOnFocusHint (/reference/android/view/View#getRevealOnFocusHint())() Returns this view's preference for reveal behavior when it gains focus. |
| final int | getRight (/reference/android/view/View#getRight())() Right position of this view relative to its parent. |

| | |
|---|---|
| float | getRightFadingEdgeStrength (/reference/android/view/View#getRightFadingEdgeStrength())() Returns the strength, or intensity, of the right faded edge. |
| int | getRightPaddingOffset (/reference/android/view/View#getRightPaddingOffset())() Amount by which to extend the right fading region. |
| View (/reference/android/view/View) | getRootView (/reference/android/view/View#getRootView())() Finds the topmost view in the current view hierarchy. |
| WindowInsets (/reference/android/view/WindowInsets) | getRootWindowInsets (/reference/android/view/View#getRootWindowInsets())() Provide original WindowInsets that are dispatched to the view hierarchy. |
| float | getRotation (/reference/android/view/View#getRotation())() The degrees that the view is rotated around the pivot point. |
| float | getRotationX (/reference/android/view/View#getRotationX())() The degrees that the view is rotated around the horizontal axis through the pivot point. |
| float | getRotationY (/reference/android/view/View#getRotationY())() The degrees that the view is rotated around the vertical axis through the pivot point. |
| float | getScaleX (/reference/android/view/View#getScaleX())() The amount that the view is scaled in x around the pivot point, as a proportion of the view's unscaled width. |

| | |
|---|---|
| `float` | **getScaleY** (/reference/android/view/View#getScaleY())() <br><br> The amount that the view is scaled in y around the pivot point, as a proportion of the view's unscaled height. |
| `int` | **getScrollBarDefaultDelayBeforeFade** (/reference/android/view/View#getScrollBarDefaultDelayBeforeFade())() <br><br> Returns the delay before scrollbars fade. |
| `int` | **getScrollBarFadeDuration** (/reference/android/view/View#getScrollBarFadeDuration())() <br><br> Returns the scrollbar fade duration. |
| `int` | **getScrollBarSize** (/reference/android/view/View#getScrollBarSize())() <br><br> Returns the scrollbar size. |
| `int` | **getScrollBarStyle** (/reference/android/view/View#getScrollBarStyle())() <br><br> Returns the current scrollbar style. |
| `int` | **getScrollIndicators** (/reference/android/view/View#getScrollIndicators())() <br><br> Returns a bitmask representing the enabled scroll indicators. |
| `final int` | **getScrollX** (/reference/android/view/View#getScrollX())() <br><br> Return the scrolled left position of this view. |
| `final int` | **getScrollY** (/reference/android/view/View#getScrollY())() <br><br> Return the scrolled top position of this view. |

| | |
|---|---|
| **int** | **getSolidColor** (/reference/android/view/View#getSolidColor())() <br><br> Override this if your view is known to always be drawn on top of a solid color background, and needs to draw fading edges. |
| **int** | **getSourceLayoutResId** (/reference/android/view/View#getSourceLayoutResId())() <br><br> A **View** (/reference/android/view/View) can be inflated from an XML layout. |
| final CharSequence (/reference/java/lang/CharSequence) | getStateDescription (/reference/android/view/View#getStateDescription())() <br><br> Returns the **View** (/reference/android/view/View)'s state description. |
| **StateListAnimator** (/reference/android/animation/StateListAnimator) | **getStateListAnimator** (/reference/android/view/View#getStateListAnimator())() <br><br> Returns the current StateListAnimator if exists. |
| **int** | **getSuggestedMinimumHeight** (/reference/android/view/View#getSuggestedMinimumHeight())() <br><br> Returns the suggested minimum height that the view should use. |
| **int** | **getSuggestedMinimumWidth** (/reference/android/view/View#getSuggestedMinimumWidth())() <br><br> Returns the suggested minimum width that the view should use. |
| **List** (/reference/java/util/List)<**Rect** (/reference/android/graphics/Rect)> | **getSystemGestureExclusionRects** (/reference/android/view/View#getSystemGestureExclusionRects())() <br><br> Retrieve the list of areas within this view's post-layout coordinate space where the system should not intercept touch or other pointing device gestures. |
| **int** | **getSystemUiVisibility** (/reference/android/view/View#getSystemUiVisibility())() |

| | |
|---|---|
| | Returns the last **setSystemUiVisibility(int)** (/reference/android/view/View#setSystemUiVisibility(int)) that this view has requested. |
| **Object** (/reference/java/lang/Object) | **getTag** (/reference/android/view/View#getTag())() Returns this view's tag. |
| **Object** (/reference/java/lang/Object) | **getTag** (/reference/android/view/View#getTag(int))(**int key**) Returns the tag associated with this view and the specified key. |
| **int** | **getTextAlignment** (/reference/android/view/View#getTextAlignment())() Return the resolved text alignment. |
| **int** | **getTextDirection** (/reference/android/view/View#getTextDirection())() Return the resolved text direction. |
| **CharSequence** (/reference/java/lang/CharSequence) | **getTooltipText** (/reference/android/view/View#getTooltipText())() Returns the view's tooltip text. |
| **final int** | **getTop** (/reference/android/view/View#getTop())() Top position of this view relative to its parent. |
| **float** | **getTopFadingEdgeStrength** (/reference/android/view/View#getTopFadingEdgeStrength())() Returns the strength, or intensity, of the top faded edge. |
| **int** | **getTopPaddingOffset** (/reference/android/view/View#getTopPaddingOffset())() |

| | |
|---|---|
| | Amount by which to extend the top fading region. |
| **TouchDelegate** (/reference/android/view/TouchDelegate) | **getTouchDelegate** (/reference/android/view/View#getTouchDelegate())()<br><br>Gets the TouchDelegate for this View. |
| **ArrayList** (/reference/java/util/ArrayList)<**View** (/reference/android/view/View)> | **getTouchables** (/reference/android/view/View#getTouchables())()<br><br>Find and return all touchable views that are descendants of this view, possibly including this view if it is touchable itself. |
| **float** | **getTransitionAlpha** (/reference/android/view/View#getTransitionAlpha())()<br><br>This property is intended only for use by the Fade transition, which animates it to produce a visual translucency that does not side-effect (or get affected by) the real alpha property. |
| **String** (/reference/java/lang/String) | **getTransitionName** (/reference/android/view/View#getTransitionName())()<br><br>Returns the name of the View to be used to identify Views in Transitions. |
| **float** | **getTranslationX** (/reference/android/view/View#getTranslationX())()<br><br>The horizontal location of this view relative to its **left** (/reference/android/view/View#getLeft()) position. |
| **float** | **getTranslationY** (/reference/android/view/View#getTranslationY())()<br><br>The vertical location of this view relative to its **top** (/reference/android/view/View#getTop()) position. |
| **float** | **getTranslationZ** (/reference/android/view/View#getTranslationZ())()<br><br>The depth location of this view relative to its **elevation** (/reference/android/view/View#getElevation()). |

| | |
|---|---|
| `long` | **`getUniqueDrawingId`** (/reference/android/view/View#getUniqueDrawingId())() <br><br> Get the identifier used for this view by the drawing system. |
| `int` | **`getVerticalFadingEdgeLength`** (/reference/android/view/View#getVerticalFadingEdgeLength())() <br><br> Returns the size of the vertical faded edges used to indicate that more content in this view is visible. |
| `int` | **`getVerticalScrollbarPosition`** (/reference/android/view/View#getVerticalScrollbarPosition())() |
| **`Drawable`** (/reference/android/graphics/drawable/Drawable) | **`getVerticalScrollbarThumbDrawable`** (/reference/android/view/View#getVerticalScrollbarThumbDrawable())() <br><br> Returns the currently configured Drawable for the thumb of the vertical scroll bar if it exists, null otherwise. |
| **`Drawable`** (/reference/android/graphics/drawable/Drawable) | **`getVerticalScrollbarTrackDrawable`** (/reference/android/view/View#getVerticalScrollbarTrackDrawable())() <br><br> Returns the currently configured Drawable for the track of the vertical scroll bar if it exists, null otherwise. |
| `int` | **`getVerticalScrollbarWidth`** (/reference/android/view/View#getVerticalScrollbarWidth())() <br><br> Returns the width of the vertical scrollbar. |
| **`ViewTreeObserver`** (/reference/android/view/ViewTreeObserver) | **`getViewTreeObserver`** (/reference/android/view/View#getViewTreeObserver())() |

| | |
|---|---|
| | Returns the ViewTreeObserver for this view's hierarchy. |
| `int` | **getVisibility** (/reference/android/view/View#getVisibility())() <br><br> Returns the visibility status for this view. |
| `final int` | **getWidth** (/reference/android/view/View#getWidth())() <br><br> Return the width of your view. |
| `int` | **getWindowAttachCount** (/reference/android/view/View#getWindowAttachCount())() |
| **WindowId** (/reference/android/view/WindowId) | **getWindowId** (/reference/android/view/View#getWindowId())() <br><br> Retrieve the **WindowId** (/reference/android/view/WindowId) for the window this view is currently attached to. |
| WindowInsetsController (/reference/android/view/WindowInsetsController) | getWindowInsetsController (/reference/android/view/View#getWindowInsetsController())() <br><br> Retrieves the single **WindowInsetsController** (/reference/android/view/WindowInsetsController) of the window this view is attached to. |
| `int` | **getWindowSystemUiVisibility** (/reference/android/view/View#getWindowSystemUiVisibility())() <br><br> Returns the current system UI visibility that is currently set for the entire window. |
| **IBinder** (/reference/android/os/IBinder) | **getWindowToken** (/reference/android/view/View#getWindowToken())() <br><br> Retrieve a unique token identifying the window this view is attached to. |
| `int` | **getWindowVisibility** (/reference/android/view/View#getWindowVisibility())() |

| | |
|---|---|
| | Returns the current visibility of the window this view is attached to (either **GONE** (/reference/android/view/View#GONE), **INVISIBLE** (/reference/android/view/View#INVISIBLE), or **VISIBLE** (/reference/android/view/View#VISIBLE)). |
| **void** | **getWindowVisibleDisplayFrame** (/reference/android/view/View#getWindowVisibleDisplayFrame(android.graphics.Rect))**(Rect** (/reference/android/graphics/Rect) **outRect)**<br><br>Retrieve the overall visible display size in which the window this view is attached to has been positioned in. |
| **float** | **getX** (/reference/android/view/View#getX())**()**<br><br>The visual x position of this view, in pixels. |
| **float** | **getY** (/reference/android/view/View#getY())**()**<br><br>The visual y position of this view, in pixels. |
| **float** | **getZ** (/reference/android/view/View#getZ())**()**<br><br>The visual z position of this view, in pixels. |
| **boolean** | **hasExplicitFocusable** (/reference/android/view/View#hasExplicitFocusable())**()**<br><br>Returns true if this view is focusable or if it contains a reachable View for which **hasExplicitFocusable()** (/reference/android/view/View#hasExplicitFocusable()) returns **true**. |
| **boolean** | **hasFocus** (/reference/android/view/View#hasFocus())**()**<br><br>Returns true if this view has focus itself, or is the ancestor of the view that has focus. |

| | |
|---|---|
| **boolean** | **hasFocusable** (/reference/android/view/View#hasFocusable())() <br><br> Returns true if this view is focusable or if it contains a reachable View for which **hasFocusable()** (/reference/android/view/View#hasFocusable()) returns **true**. |
| **boolean** | **hasNestedScrollingParent** (/reference/android/view/View#hasNestedScrollingParent())() <br><br> Returns true if this view has a nested scrolling parent. |
| **boolean** | **hasOnClickListeners** (/reference/android/view/View#hasOnClickListeners())() <br><br> Return whether this view has an attached OnClickListener. |
| **boolean** | **hasOnLongClickListeners** (/reference/android/view/View#hasOnLongClickListeners())() <br><br> Return whether this view has an attached OnLongClickListener. |
| **boolean** | **hasOverlappingRendering** (/reference/android/view/View#hasOverlappingRendering())() <br><br> Returns whether this View has content which overlaps. |
| **boolean** | **hasPointerCapture** (/reference/android/view/View#hasPointerCapture())() <br><br> Checks pointer capture status. |
| **boolean** | **hasTransientState** (/reference/android/view/View#hasTransientState())() <br><br> Indicates whether the view is currently tracking transient state that the app should not need to concern itself with saving and restoring, but that the framework should take special note to preserve when possible. |

| | |
|---|---|
| **boolean** | **hasWindowFocus** (/reference/android/view/View#hasWindowFocus())**()**<br><br>Returns true if this view is in a window that currently has window focus. |
| **static View** (/reference/android/view/View) | **inflate** (/reference/android/view/View#inflate(android.content.Context,%20int,%20android.view.ViewGroup))**(Context** (/reference/android/content/Context) **context, int resource, ViewGroup** (/reference/android/view/ViewGroup) **root)**<br><br>Inflate a view from an XML resource. |
| **void** | **invalidate** (/reference/android/view/View#invalidate())**()**<br><br>Invalidate the whole view. |
| **void** | **invalidate** (/reference/android/view/View#invalidate(android.graphics.Rect))**(Rect** (/reference/android/graphics/Rect) **dirty)**<br><br>*This method was deprecated in API level 28. The switch to hardware accelerated rendering in API 14 reduced the importance of the dirty rectangle. In API 21 the given rectangle is ignored entirely in favor of an internally-calculated area instead. Because of this, clients are encouraged to just call* **invalidate()** (/reference/android/view/View#invalidate()). |
| **void** | **invalidate** (/reference/android/view/View#invalidate(int,%20int,%20int,%20int))**(int l, int t, int r, int b)**<br><br>*This method was deprecated in API level 28. The switch to hardware accelerated rendering in API 14 reduced the importance of the dirty rectangle. In API 21 the given rectangle is ignored entirely in favor of an internally-calculated area instead. Because of this, clients are encouraged to just call* **invalidate()** (/reference/android/view/View#invalidate()). |

| | |
|---|---|
| **void** | **invalidateDrawable** (/reference/android/view/View#invalidateDrawable(android.graphics.drawable.Drawable)) (**Drawable** (/reference/android/graphics/drawable/Drawable) **drawable**)<br><br>Invalidates the specified Drawable. |
| **void** | **invalidateOutline** (/reference/android/view/View#invalidateOutline())()<br><br>Called to rebuild this View's Outline from its **ViewOutlineProvider** (/reference/android/view/ViewOutlineProvider) |
| **boolean** | **isAccessibilityFocused** (/reference/android/view/View#isAccessibilityFocused())()<br><br>Returns whether this View is accessibility focused. |
| **boolean** | **isAccessibilityHeading** (/reference/android/view/View#isAccessibilityHeading())()<br><br>Gets whether this view is a heading for accessibility purposes. |
| **boolean** | **isActivated** (/reference/android/view/View#isActivated())()<br><br>Indicates the activation state of this view. |
| **boolean** | **isAttachedToWindow** (/reference/android/view/View#isAttachedToWindow())()<br><br>Returns true if this view is currently attached to a window. |
| **boolean** | **isClickable** (/reference/android/view/View#isClickable())()<br><br>Indicates whether this view reacts to click events or not. |

| | |
|---|---|
| **boolean** | **isContextClickable** (/reference/android/view/View#isContextClickable())() <br><br> Indicates whether this view reacts to context clicks or not. |
| **boolean** | **isDirty** (/reference/android/view/View#isDirty())() <br><br> True if this view has changed since the last time being drawn. |
| **boolean** | **isDrawingCacheEnabled** (/reference/android/view/View#isDrawingCacheEnabled())() <br><br> *This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations,* **setLayerType(int, android.graphics.Paint)** (/reference/android/view/View#setLayerType(int,%20android.graphics.Paint)) *handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a* **Canvas** (/reference/android/graphics/Canvas) *from either a* **Bitmap** (/reference/android/graphics/Bitmap) *or* **Picture** (/reference/android/graphics/Picture) *and call* **draw(android.graphics.Canvas)** (/reference/android/view/View#draw(android.graphics.Canvas)) *on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as* **Config.HARDWARE** (/reference/android/graphics/Bitmap.Config#HARDWARE) *bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the* **PixelCopy** (/reference/android/view/PixelCopy) *API is recommended.* |
| **boolean** | **isDuplicateParentStateEnabled** (/reference/android/view/View#isDuplicateParentStateEnabled())() |

| | |
|---|---|
| | Indicates whether this duplicates its drawable state from its parent. |
| **boolean** | **isEnabled** (/reference/android/view/View#isEnabled())() Returns the enabled status for this view. |
| **final boolean** | **isFocusable** (/reference/android/view/View#isFocusable())() Returns whether this View is currently able to take focus. |
| **final boolean** | **isFocusableInTouchMode** (/reference/android/view/View#isFocusableInTouchMode())() When a view is focusable, it may not want to take focus when in touch mode. |
| **boolean** | **isFocused** (/reference/android/view/View#isFocused())() Returns true if this view has focus |
| **final boolean** | **isFocusedByDefault** (/reference/android/view/View#isFocusedByDefault())() Returns whether this View should receive focus when the focus is restored for the view hierarchy containing this view. |
| **boolean** | **isForceDarkAllowed** (/reference/android/view/View#isForceDarkAllowed())() See **setForceDarkAllowed(boolean)** (/reference/android/view/View#setForceDarkAllowed(boolean)) |
| **boolean** | **isHapticFeedbackEnabled** (/reference/android/view/View#isHapticFeedbackEnabled())() |
| **boolean** | **isHardwareAccelerated** (/reference/android/view/View#isHardwareAccelerated())() |

| | |
|---|---|
| | Indicates whether this view is attached to a hardware accelerated window or not. |
| **boolean** | **isHorizontalFadingEdgeEnabled** (/reference/android/view/View#isHorizontalFadingEdgeEnabled())() <br><br> Indicate whether the horizontal edges are faded when the view is scrolled horizontally. |
| **boolean** | **isHorizontalScrollBarEnabled** (/reference/android/view/View#isHorizontalScrollBarEnabled())() <br><br> Indicate whether the horizontal scrollbar should be drawn or not. |
| **boolean** | **isHovered** (/reference/android/view/View#isHovered())() <br><br> Returns true if the view is currently hovered. |
| **boolean** | **isImportantForAccessibility** (/reference/android/view/View#isImportantForAccessibility())() <br><br> Computes whether this view should be exposed for accessibility. |
| **final boolean** | **isImportantForAutofill** (/reference/android/view/View#isImportantForAutofill())() <br><br> Hints the Android System whether the **AssistStructure.ViewNode** (/reference/android/app/assist/AssistStructure.ViewNode) associated with this view is considered important for autofill purposes. |
| final boolean | isImportantForContentCapture (/reference/android/view/View#isImportantForContentCapture())() <br><br> Hints the Android System whether this view is considered important for content capture, based on the value explicitly set by **setImportantForContentCapture(int)** (/reference/android/view/View#setImportantForContentCapture(int)) and heuristics when it's **IMPORTANT_FOR_CONTENT_CAPTURE_AUTO** |

| | |
|---|---|
| | (/reference/android/view/View#IMPORTANT_FOR_CONTENT_CAPTURE_AUTO). |
| boolean | **isInEditMode** (/reference/android/view/View#isInEditMode())() <br> Indicates whether this View is currently in edit mode. |
| boolean | **isInLayout** (/reference/android/view/View#isInLayout())() <br> Returns whether the view hierarchy is currently undergoing a layout pass. |
| boolean | **isInTouchMode** (/reference/android/view/View#isInTouchMode())() <br> Returns whether the device is currently in touch mode. |
| final boolean | **isKeyboardNavigationCluster** (/reference/android/view/View#isKeyboardNavigationCluster())() <br> Returns whether this View is a root of a keyboard navigation cluster. |
| boolean | **isLaidOut** (/reference/android/view/View#isLaidOut())() <br> Returns true if this view has been through at least one layout since it was last attached to or detached from a window. |
| boolean | **isLayoutDirectionResolved** (/reference/android/view/View#isLayoutDirectionResolved())() |
| boolean | **isLayoutRequested** (/reference/android/view/View#isLayoutRequested())() <br> Indicates whether or not this view's layout will be requested during the next hierarchy layout pass. |
| boolean | **isLongClickable** (/reference/android/view/View#isLongClickable())() |

| | |
|---|---|
| | Indicates whether this view reacts to long click events or not. |
| **boolean** | **isNestedScrollingEnabled** (/reference/android/view/View#isNestedScrollingEnabled())() <br><br> Returns true if nested scrolling is enabled for this view. |
| **boolean** | **isOpaque** (/reference/android/view/View#isOpaque())() <br><br> Indicates whether this View is opaque. |
| **boolean** | **isPaddingOffsetRequired** (/reference/android/view/View#isPaddingOffsetRequired())() <br><br> If the View draws content inside its padding and enables fading edges, it needs to support padding offsets. |
| **boolean** | **isPaddingRelative** (/reference/android/view/View#isPaddingRelative())() <br><br> Return if the padding has been set through relative values **setPaddingRelative(int, int, int, int)** (/reference/android/view/View#setPaddingRelative(int,%20int,%20int,%20int)) or through |
| **boolean** | **isPivotSet** (/reference/android/view/View#isPivotSet())() <br><br> Returns whether or not a pivot has been set by a call to **setPivotX(float)** (/reference/android/view/View#setPivotX(float)) or **setPivotY(float)** (/reference/android/view/View#setPivotY(float)). |
| **boolean** | **isPressed** (/reference/android/view/View#isPressed())() <br><br> Indicates whether the view is currently in pressed state. |

| | |
|---|---|
| **boolean** | **isSaveEnabled** (/reference/android/view/View#isSaveEnabled())() <br><br> Indicates whether this view will save its state (that is, whether its **onSaveInstanceState()** (/reference/android/view/View#onSaveInstanceState()) method will be called). |
| **boolean** | **isSaveFromParentEnabled** (/reference/android/view/View#isSaveFromParentEnabled())() <br><br> Indicates whether the entire hierarchy under this view will save its state when a state saving traversal occurs from its parent. |
| **boolean** | **isScreenReaderFocusable** (/reference/android/view/View#isScreenReaderFocusable())() <br><br> Returns whether the view should be treated as a focusable unit by screen reader accessibility tools. |
| **boolean** | **isScrollContainer** (/reference/android/view/View#isScrollContainer())() <br><br> Indicates whether this view is one of the set of scrollable containers in its window. |
| **boolean** | **isScrollbarFadingEnabled** (/reference/android/view/View#isScrollbarFadingEnabled())() <br><br> Returns true if scrollbars will fade when this view is not scrolling |
| **boolean** | **isSelected** (/reference/android/view/View#isSelected())() <br><br> Indicates the selection state of this view. |
| final boolean | isShowingLayoutBounds (/reference/android/view/View#isShowingLayoutBounds())() |

| | |
|---|---|
| | Returns **true** when the View is attached and the system developer setting to show the layout bounds is enabled or **false** otherwise. |
| **boolean** | **isShown** (/reference/android/view/View#isShown())() <br><br> Returns the visibility of this view and all of its ancestors |
| **boolean** | **isSoundEffectsEnabled** (/reference/android/view/View#isSoundEffectsEnabled())() |
| **final boolean** | **isTemporarilyDetached** (/reference/android/view/View#isTemporarilyDetached())() <br><br> Tells whether the **View** (/reference/android/view/View) is in the state between **onStartTemporaryDetach()** (/reference/android/view/View#onStartTemporaryDetach()) and **onFinishTemporaryDetach()** (/reference/android/view/View#onFinishTemporaryDetach()). |
| **boolean** | **isTextAlignmentResolved** (/reference/android/view/View#isTextAlignmentResolved())() |
| **boolean** | **isTextDirectionResolved** (/reference/android/view/View#isTextDirectionResolved())() |
| **boolean** | **isVerticalFadingEdgeEnabled** (/reference/android/view/View#isVerticalFadingEdgeEnabled())() <br><br> Indicate whether the vertical edges are faded when the view is scrolled horizontally. |
| **boolean** | **isVerticalScrollBarEnabled** (/reference/android/view/View#isVerticalScrollBarEnabled())() <br><br> Indicate whether the vertical scrollbar should be drawn or not. |

| | |
|---|---|
| `boolean` | **isVisibleToUserForAutofill** (/reference/android/view/View#isVisibleToUserForAutofill(int))**(int virtualId)**<br><br>Computes whether this virtual autofill view is visible to the user. |
| `void` | **jumpDrawablesToCurrentState** (/reference/android/view/View#jumpDrawablesToCurrentState())**()**<br><br>Call **Drawable#jumpToCurrentState()** (/reference/android/graphics/drawable/Drawable#jumpToCurrentState()) on all Drawable objects associated with this view. |
| **View** (/reference/android/view/View) | **keyboardNavigationClusterSearch** (/reference/android/view/View#keyboardNavigationClusterSearch(android.view.View,%20int))<br>**(View** (/reference/android/view/View) **currentCluster, int direction)**<br><br>Find the nearest keyboard navigation cluster in the specified direction. |
| `void` | **layout** (/reference/android/view/View#layout(int,%20int,%20int,%20int))**(int l, int t, int r, int b)**<br><br>Assign a size and position to a view and all of its descendants<br><br>This is the second phase of the layout mechanism. |
| `final void` | **measure** (/reference/android/view/View#measure(int,%20int))**(int widthMeasureSpec, int heightMeasureSpec)**<br><br>This is called to find out how big a view should be. |
| `static int[]` | **mergeDrawableStates** (/reference/android/view/View#mergeDrawableStates(int[],%20int[]))**(int[]** |

| | |
|---|---|
| | **baseState, int[] additionalState)**<br><br>Merge your own state values in ***additionalState*** into the base state values ***baseState*** that were returned by **onCreateDrawableState(int)** (/reference/android/view/View#onCreateDrawableState(int)). |
| **void** | **offsetLeftAndRight** (/reference/android/view/View#offsetLeftAndRight(int))**(int offset)**<br><br>Offset this view's horizontal location by the specified amount of pixels. |
| **void** | **offsetTopAndBottom** (/reference/android/view/View#offsetTopAndBottom(int))**(int offset)**<br><br>Offset this view's vertical location by the specified number of pixels. |
| **void** | **onAnimationEnd** (/reference/android/view/View#onAnimationEnd())**()**<br><br>Invoked by a parent ViewGroup to notify the end of the animation currently associated with this view. |
| **void** | **onAnimationStart** (/reference/android/view/View#onAnimationStart())**()**<br><br>Invoked by a parent ViewGroup to notify the start of the animation currently associated with this view. |
| **WindowInsets** (/reference/android/view/WindowInsets) | **onApplyWindowInsets** (/reference/android/view/View#onApplyWindowInsets(android.view.WindowInsets))**(WindowInsets** (/reference/android/view/WindowInsets) **insets)**<br><br>Called when the view should apply **WindowInsets** (/reference/android/view/WindowInsets) according to its internal policy. |

| | |
|---|---|
| `void` | **`onAttachedToWindow`** (/reference/android/view/View#onAttachedToWindow())**`()`**<br><br>This is called when the view is attached to a window. |
| `void` | **`onCancelPendingInputEvents`** (/reference/android/view/View#onCancelPendingInputEvents())**`()`**<br><br>Called as the result of a call to **`cancelPendingInputEvents()`** (/reference/android/view/View#cancelPendingInputEvents()) on this view or a parent view. |
| `boolean` | **`onCapturedPointerEvent`** (/reference/android/view/View#onCapturedPointerEvent(android.view.MotionEvent))**`(`**`MotionEvent` (/reference/android/view/MotionEvent) **`event)`**<br><br>Implement this method to handle captured pointer events |
| `boolean` | **`onCheckIsTextEditor`** (/reference/android/view/View#onCheckIsTextEditor())**`()`**<br><br>Check whether the called view is a text editor, in which case it would make sense to automatically display a soft input window for it. |
| `void` | **`onConfigurationChanged`** (/reference/android/view/View#onConfigurationChanged(android.content.res.Configuration))<br>**`(`**`Configuration` (/reference/android/content/res/Configuration) **`newConfig)`**<br><br>Called when the current configuration of the resources being used by the application have changed. |
| `void` | **`onCreateContextMenu`** (/reference/android/view/View#onCreateContextMenu(android.view.ContextMenu))<br>**`(`**`ContextMenu` (/reference/android/view/ContextMenu) **`menu)`** |

| | |
|---|---|
| | Views should implement this if the view itself is going to add items to the context menu. |
| `int[]` | **onCreateDrawableState** (/reference/android/view/View#onCreateDrawableState(int))<br>**(int extraSpace)**<br><br>Generate the new **Drawable** (/reference/android/graphics/drawable/Drawable) state for this view. |
| **InputConnection** (/reference/android/view/inputmethod/InputConnection) | **onCreateInputConnection** (/reference/android/view/View#onCreateInputConnection(android.view.inputmethod.EditorInfo))<br>**(EditorInfo** (/reference/android/view/inputmethod/EditorInfo) **outAttrs)**<br><br>Create a new InputConnection for an InputMethod to interact with the view. |
| `void` | **onDetachedFromWindow** (/reference/android/view/View#onDetachedFromWindow())<br>**()**<br><br>This is called when the view is detached from a window. |
| `void` | **onDisplayHint** (/reference/android/view/View#onDisplayHint(int))**(int hint)**<br><br>Gives this view a hint about whether is displayed or not. |
| `boolean` | **onDragEvent** (/reference/android/view/View#onDragEvent(android.view.DragEvent))<br>**(DragEvent** (/reference/android/view/DragEvent) **event)**<br><br>Handles drag events sent by the system following a call to **startDragAndDrop()** (/reference/android/view/View#startDragAndDrop(android.content.ClipData,%20android.view.View.DragShadowBuilder,%20java.lang.Object,%20int))<br>. |

| | |
|---|---|
| `void` | **onDraw** (/reference/android/view/View#onDraw(android.graphics.Canvas))**(Canvas** (/reference/android/graphics/Canvas) **canvas)**<br><br>Implement this to do your drawing. |
| `void` | **onDrawForeground** (/reference/android/view/View#onDrawForeground(android.graphics.Canvas))**(Canvas** (/reference/android/graphics/Canvas) **canvas)**<br><br>Draw any foreground content for this view. |
| `final void` | **onDrawScrollBars** (/reference/android/view/View#onDrawScrollBars(android.graphics.Canvas))**(Canvas** (/reference/android/graphics/Canvas) **canvas)**<br><br>Request the drawing of the horizontal and the vertical scrollbar. |
| `boolean` | **onFilterTouchEventForSecurity** (/reference/android/view/View#onFilterTouchEventForSecurity(android.view.MotionEvent))<br>**(MotionEvent** (/reference/android/view/MotionEvent) **event)**<br><br>Filter the touch event to apply security policies. |
| `void` | **onFinishInflate** (/reference/android/view/View#onFinishInflate())**()**<br><br>Finalize inflating a view from XML. |
| `void` | **onFinishTemporaryDetach** (/reference/android/view/View#onFinishTemporaryDetach())**()**<br><br>Called after **onStartTemporaryDetach()** (/reference/android/view/View#onStartTemporaryDetach()) when the container is done |

| | |
|---|---|
| | changing the view. |
| `void` | **onFocusChanged** (/reference/android/view/View#onFocusChanged(boolean,%20int,%20android.graphics.Rect))<br>**(boolean gainFocus, int direction, Rect** (/reference/android/graphics/Rect) **previouslyFocusedRect)**<br><br>Called by the view system when the focus state of this view changes. |
| `boolean` | **onGenericMotionEvent** (/reference/android/view/View#onGenericMotionEvent(android.view.MotionEvent))<br>**(MotionEvent** (/reference/android/view/MotionEvent) **event)**<br><br>Implement this method to handle generic motion events. |
| `void` | **onHoverChanged** (/reference/android/view/View#onHoverChanged(boolean))<br>**(boolean hovered)**<br><br>Implement this method to handle hover state changes. |
| `boolean` | **onHoverEvent** (/reference/android/view/View#onHoverEvent(android.view.MotionEvent))<br>**(MotionEvent** (/reference/android/view/MotionEvent) **event)**<br><br>Implement this method to handle hover events. |
| `void` | **onInitializeAccessibilityEvent** (/reference/android/view/View#onInitializeAccessibilityEvent(android.view.accessibility.AccessibilityEvent))<br>**(AccessibilityEvent** (/reference/android/view/accessibility/AccessibilityEvent) **event)** |

| | |
|---|---|
| | Initializes an **AccessibilityEvent** (/reference/android/view/accessibility/AccessibilityEvent) with information about this View which is the event source. |
| **void** | **onInitializeAccessibilityNodeInfo** (/reference/android/view/View#onInitializeAccessibilityNodeInfo(android.view.accessibility.AccessibilityNodeInfo)) **(AccessibilityNodeInfo** (/reference/android/view/accessibility/AccessibilityNodeInfo) **info)**<br><br>Initializes an **AccessibilityNodeInfo** (/reference/android/view/accessibility/AccessibilityNodeInfo) with information about this view. |
| **boolean** | **onKeyDown** (/reference/android/view/View#onKeyDown(int,%20android.view.KeyEvent)) **(int keyCode, KeyEvent** (/reference/android/view/KeyEvent) **event)**<br><br>Default implementation of **KeyEvent.Callback#onKeyDown(int, KeyEvent)** (/reference/android/view/KeyEvent.Callback#onKeyDown(int,%20android.view.KeyEvent)) : perform press of the view when **KeyEvent#KEYCODE_DPAD_CENTER** (/reference/android/view/KeyEvent#KEYCODE_DPAD_CENTER) or **KeyEvent#KEYCODE_ENTER** (/reference/android/view/KeyEvent#KEYCODE_ENTER) is released, if the view is enabled and clickable. |
| **boolean** | **onKeyLongPress** (/reference/android/view/View#onKeyLongPress(int,%20android.view.KeyEvent))**(int keyCode, KeyEvent** (/reference/android/view/KeyEvent) **event)**<br><br>Default implementation of **KeyEvent.Callback#onKeyLongPress(int, KeyEvent)** |

| | |
|---|---|
| | (/reference/android/view/KeyEvent.Callback#onKeyLongPress(int,%20android.view.KeyEvent))<br>: always returns false (doesn't handle the event). |
| **boolean** | **onKeyMultiple**<br>(/reference/android/view/View#onKeyMultiple(int,%20int,%20android.view.KeyEvent))**(int keyCode, int repeatCount, KeyEvent** (/reference/android/view/KeyEvent) **event)**<br><br>Default implementation of **KeyEvent.Callback#onKeyMultiple(int, int, KeyEvent)**<br>(/reference/android/view/KeyEvent.Callback#onKeyMultiple(int,%20int,%20android.view.KeyEvent))<br>: always returns false (doesn't handle the event). |
| **boolean** | **onKeyPreIme**<br>(/reference/android/view/View#onKeyPreIme(int,%20android.view.KeyEvent))**(int keyCode, KeyEvent** (/reference/android/view/KeyEvent) **event)**<br><br>Handle a key event before it is processed by any input method associated with the view hierarchy. |
| **boolean** | **onKeyShortcut**<br>(/reference/android/view/View#onKeyShortcut(int,%20android.view.KeyEvent))**(int keyCode, KeyEvent** (/reference/android/view/KeyEvent) **event)**<br><br>Called on the focused view when a key shortcut event is not handled. |
| **boolean** | **onKeyUp** (/reference/android/view/View#onKeyUp(int,%20android.view.KeyEvent))**(int keyCode, KeyEvent** (/reference/android/view/KeyEvent) **event)**<br><br>Default implementation of **KeyEvent.Callback#onKeyUp(int, KeyEvent)**<br>(/reference/android/view/KeyEvent.Callback#onKeyUp(int,%20android.view.KeyEvent)): |

| | |
|---|---|
| | perform clicking of the view when **KeyEvent#KEYCODE_DPAD_CENTER** (/reference/android/view/KeyEvent#KEYCODE_DPAD_CENTER), **KeyEvent#KEYCODE_ENTER** (/reference/android/view/KeyEvent#KEYCODE_ENTER) or **KeyEvent#KEYCODE_SPACE** (/reference/android/view/KeyEvent#KEYCODE_SPACE) is released. |
| **void** | **onLayout** (/reference/android/view/View#onLayout(boolean,%20int,%20int,%20int,%20int))**(boolean changed, int left, int top, int right, int bottom)**<br><br>Called from layout when this view should assign a size and position to each of its children. |
| **void** | **onMeasure** (/reference/android/view/View#onMeasure(int,%20int))**(int widthMeasureSpec, int heightMeasureSpec)**<br><br>Measure the view and its content to determine the measured width and the measured height. |
| **void** | **onOverScrolled** (/reference/android/view/View#onOverScrolled(int,%20int,%20boolean,%20boolean))**(int scrollX, int scrollY, boolean clampedX, boolean clampedY)**<br><br>Called by **overScrollBy(int, int, int, int, int, int, int, int, boolean)** (/reference/android/view/View#overScrollBy(int,%20int,%20int,%20int,%20int,%20int,%20int,%20int,%20boolean))<br>to respond to the results of an over-scroll operation. |
| **void** | **onPointerCaptureChange** (/reference/android/view/View#onPointerCaptureChange(boolean))**(boolean hasCapture)**<br><br>Called when the window has just acquired or lost pointer capture. |

| | |
|---|---|
| `void` | **onPopulateAccessibilityEvent** (/reference/android/view/View#onPopulateAccessibilityEvent(android.view.accessibility.AccessibilityEvent)) **(AccessibilityEvent** (/reference/android/view/accessibility/AccessibilityEvent) **event)**<br><br>Called from **dispatchPopulateAccessibilityEvent(android.view.accessibility.AccessibilityEvent)** (/reference/android/view/View#dispatchPopulateAccessibilityEvent(android.view.accessibility.AccessibilityEvent)) giving a chance to this View to populate the accessibility event with its text content. |
| `void` | **onProvideAutofillStructure** (/reference/android/view/View#onProvideAutofillStructure(android.view.ViewStructure,%20int)) **(ViewStructure** (/reference/android/view/ViewStructure) **structure, int flags)**<br><br>Populates a **ViewStructure** (/reference/android/view/ViewStructure) to fullfil an autofill request. |
| `void` | **onProvideAutofillVirtualStructure** (/reference/android/view/View#onProvideAutofillVirtualStructure(android.view.ViewStructure,%20int)) **(ViewStructure** (/reference/android/view/ViewStructure) **structure, int flags)**<br><br>Populates a **ViewStructure** (/reference/android/view/ViewStructure) containing virtual children to fullfil an autofill request. |

| | |
|---|---|
| void | onProvideContentCaptureStructure (/reference/android/view/View#onProvideContentCaptureStructure(android.view.ViewStructure,%20int)) (ViewStructure (/reference/android/view/ViewStructure) structure, int flags)<br><br>Populates a **ViewStructure** (/reference/android/view/ViewStructure) for content capture. |
| **void** | **onProvideStructure** (/reference/android/view/View#onProvideStructure(android.view.ViewStructure)) **(ViewStructure** (/reference/android/view/ViewStructure) **structure)**<br><br>Called when assist structure is being retrieved from a view as part of **Activity.onProvideAssistData** (/reference/android/app/Activity#onProvideAssistData(android.os.Bundle)). |
| **void** | **onProvideVirtualStructure** (/reference/android/view/View#onProvideVirtualStructure(android.view.ViewStructure)) **(ViewStructure** (/reference/android/view/ViewStructure) **structure)**<br><br>Called when assist structure is being retrieved from a view as part of **Activity.onProvideAssistData** (/reference/android/app/Activity#onProvideAssistData(android.os.Bundle)) to generate additional virtual structure under this view. |
| **PointerIcon** (/reference/android/view/PointerIcon) | **onResolvePointerIcon** (/reference/android/view/View#onResolvePointerIcon(android.view.MotionEvent,%20int)) **(MotionEvent** (/reference/android/view/MotionEvent) **event, int pointerIndex)** |

| | |
|---|---|
| | Returns the pointer icon for the motion event, or null if it doesn't specify the icon. |
| `void` | **onRestoreInstanceState** (/reference/android/view/View#onRestoreInstanceState(android.os.Parcelable))**(Parcelable** (/reference/android/os/Parcelable) **state)**<br><br>Hook allowing a view to re-apply a representation of its internal state that had previously been generated by **onSaveInstanceState()** (/reference/android/view/View#onSaveInstanceState()). |
| `void` | **onRtlPropertiesChanged** (/reference/android/view/View#onRtlPropertiesChanged(int))**(int layoutDirection)**<br><br>Called when any RTL property (layout direction or text direction or text alignment) has been changed. |
| **Parcelable** (/reference/android/os/Parcelable) | **onSaveInstanceState** (/reference/android/view/View#onSaveInstanceState())**()**<br><br>Hook allowing a view to generate a representation of its internal state that can later be used to create a new instance with that same state. |
| `void` | **onScreenStateChanged** (/reference/android/view/View#onScreenStateChanged(int))**(int screenState)**<br><br>This method is called whenever the state of the screen this view is attached to changes. |
| `void` | **onScrollChanged** (/reference/android/view/View#onScrollChanged(int,%20int,%20int,%20int))**(int l, int t, int oldl, int oldt)**<br><br>This is called in response to an internal scroll in this view (i.e., the view scrolled its own contents). |

| | |
|---|---|
| **boolean** | **onSetAlpha** (/reference/android/view/View#onSetAlpha(int))**(int alpha)**<br><br>Invoked if there is a Transform that involves alpha. |
| **void** | **onSizeChanged** (/reference/android/view/View#onSizeChanged(int,%20int,%20int,%20int))**(int w, int h, int oldw, int oldh)**<br><br>This is called during layout when the size of this view has changed. |
| **void** | **onStartTemporaryDetach** (/reference/android/view/View#onStartTemporaryDetach())**()**<br><br>This is called when a container is going to temporarily detach a child, with **ViewGroup#detachViewFromParent(View)** (/reference/android/view/ViewGroup#detachViewFromParent(android.view.View)). |
| **boolean** | **onTouchEvent** (/reference/android/view/View#onTouchEvent(android.view.MotionEvent))<br>**(MotionEvent** (/reference/android/view/MotionEvent) **event)**<br><br>Implement this method to handle touch screen motion events. |
| **boolean** | **onTrackballEvent** (/reference/android/view/View#onTrackballEvent(android.view.MotionEvent))<br>**(MotionEvent** (/reference/android/view/MotionEvent) **event)**<br><br>Implement this method to handle trackball motion events. |
| **void** | **onVisibilityAggregated** (/reference/android/view/View#onVisibilityAggregated(boolean))**(boolean isVisible)** |

| | |
|---|---|
| | Called when the user-visibility of this View is potentially affected by a change to this view itself, an ancestor view or the window this view is attached to. |
| `void` | **onVisibilityChanged** (/reference/android/view/View#onVisibilityChanged(android.view.View,%20int))(**View** (/reference/android/view/View) **changedView, int visibility**)<br><br>Called when the visibility of the view or an ancestor of the view has changed. |
| `void` | **onWindowFocusChanged** (/reference/android/view/View#onWindowFocusChanged(boolean))(**boolean hasWindowFocus**)<br><br>Called when the window containing this view gains or loses focus. |
| `void` | **onWindowSystemUiVisibilityChanged** (/reference/android/view/View#onWindowSystemUiVisibilityChanged(int))(**int visible**)<br><br>Override to find out when the window's requested system UI visibility has changed, that is the value returned by **getWindowSystemUiVisibility()** (/reference/android/view/View#getWindowSystemUiVisibility()). |
| `void` | **onWindowVisibilityChanged** (/reference/android/view/View#onWindowVisibilityChanged(int))(**int visibility**)<br><br>Called when the window containing has change its visibility (between **GONE** (/reference/android/view/View#GONE), **INVISIBLE** (/reference/android/view/View#INVISIBLE), and **VISIBLE** (/reference/android/view/View#VISIBLE)). |
| `boolean` | **overScrollBy** |

| | |
|---|---|
| | (/reference/android/view/View#overScrollBy(int,%20int,%20int,%20int,%20int,%20int,%20int,%20int,%20boolean))<br>**(int deltaX, int deltaY, int scrollX, int scrollY, int scrollRangeX, int scrollRangeY, int maxOverScrollX, int maxOverScrollY, boolean isTouchEvent)**<br><br>Scroll the view with standard behavior for scrolling beyond the normal content boundaries. |
| **boolean** | **performAccessibilityAction** (/reference/android/view/View#performAccessibilityAction(int,%20android.os.Bundle))**(int action, Bundle** (/reference/android/os/Bundle) **arguments)**<br><br>Performs the specified accessibility action on the view. |
| **boolean** | **performClick** (/reference/android/view/View#performClick())**()**<br><br>Call this view's OnClickListener, if it is defined. |
| **boolean** | **performContextClick** (/reference/android/view/View#performContextClick(float,%20float))**(float x, float y)**<br><br>Call this view's OnContextClickListener, if it is defined. |
| **boolean** | **performContextClick** (/reference/android/view/View#performContextClick())**()**<br><br>Call this view's OnContextClickListener, if it is defined. |
| **boolean** | **performHapticFeedback** (/reference/android/view/View#performHapticFeedback(int))**(int feedbackConstant)** |

| | |
|---|---|
| | BZZZTT!!1!<br><br>Provide haptic feedback to the user for this view. |
| **boolean** | **performHapticFeedback** (/reference/android/view/View#performHapticFeedback(int,%20int))**(int feedbackConstant, int flags)**<br><br>BZZZTT!!1!<br><br>Like **performHapticFeedback(int)** (/reference/android/view/View#performHapticFeedback(int)), with additional options. |
| **boolean** | **performLongClick** (/reference/android/view/View#performLongClick(float,%20float))**(float x, float y)**<br><br>Calls this view's OnLongClickListener, if it is defined. |
| **boolean** | **performLongClick** (/reference/android/view/View#performLongClick())**()**<br><br>Calls this view's OnLongClickListener, if it is defined. |
| **void** | **playSoundEffect** (/reference/android/view/View#playSoundEffect(int))**(int soundConstant)**<br><br>Play a sound effect for this view. |
| **boolean** | **post** (/reference/android/view/View#post(java.lang.Runnable))**(Runnable** (/reference/java/lang/Runnable) **action)**<br><br>Causes the Runnable to be added to the message queue. |
| **boolean** | **postDelayed** (/reference/android/view/View#postDelayed(java.lang.Runnable,%20long))**(Runnable** |

| | |
|---|---|
| | (/reference/java/lang/Runnable) **`action, long delayMillis)`**<br><br>Causes the Runnable to be added to the message queue, to be run after the specified amount of time elapses. |
| **`void`** | **`postInvalidate`** (/reference/android/view/View#postInvalidate())**`()`**<br><br>Cause an invalidate to happen on a subsequent cycle through the event loop. |
| **`void`** | **`postInvalidate`**<br>(/reference/android/view/View#postInvalidate(int,%20int,%20int,%20int))**`(int left, int top, int right, int bottom)`**<br><br>Cause an invalidate of the specified area to happen on a subsequent cycle through the event loop. |
| **`void`** | **`postInvalidateDelayed`**<br>(/reference/android/view/View#postInvalidateDelayed(long,%20int,%20int,%20int,%20int))<br>**`(long delayMilliseconds, int left, int top, int right, int bottom)`**<br><br>Cause an invalidate of the specified area to happen on a subsequent cycle through the event loop. |
| **`void`** | **`postInvalidateDelayed`**<br>(/reference/android/view/View#postInvalidateDelayed(long))**`(long delayMilliseconds)`**<br><br>Cause an invalidate to happen on a subsequent cycle through the event loop. |
| **`void`** | **`postInvalidateOnAnimation`**<br>(/reference/android/view/View#postInvalidateOnAnimation(int,%20int,%20int,%20int))<br>**`(int left, int top, int right, int bottom)`** |

| | |
|---|---|
| | Cause an invalidate of the specified area to happen on the next animation time step, typically the next display frame. |
| `void` | **postInvalidateOnAnimation** (/reference/android/view/View#postInvalidateOnAnimation())**()**<br><br>Cause an invalidate to happen on the next animation time step, typically the next display frame. |
| `void` | **postOnAnimation** (/reference/android/view/View#postOnAnimation(java.lang.Runnable))**(Runnable** (/reference/java/lang/Runnable) **action)**<br><br>Causes the Runnable to execute on the next animation time step. |
| `void` | **postOnAnimationDelayed** (/reference/android/view/View#postOnAnimationDelayed(java.lang.Runnable,%20long))**(Runnable** (/reference/java/lang/Runnable) **action, long delayMillis)**<br><br>Causes the Runnable to execute on the next animation time step, after the specified amount of time elapses. |
| `void` | **refreshDrawableState** (/reference/android/view/View#refreshDrawableState())**()**<br><br>Call this to force a view to update its drawable state. |
| `void` | **releasePointerCapture** (/reference/android/view/View#releasePointerCapture())**()**<br><br>Releases the pointer capture. |
| `boolean` | **removeCallbacks** (/reference/android/view/View#removeCallbacks(java.lang.Runnable))**(Runnable** (/reference/java/lang/Runnable) **action)** |

| | |
|---|---|
| | Removes the specified Runnable from the message queue. |
| **void** | **removeOnAttachStateChangeListener** (/reference/android/view/View#removeOnAttachStateChangeListener(android.view.View.OnAttachStateChangeListener)) **(View.OnAttachStateChangeListener** (/reference/android/view/View.OnAttachStateChangeListener) **listener)**<br><br>Remove a listener for attach state changes. |
| **void** | **removeOnLayoutChangeListener** (/reference/android/view/View#removeOnLayoutChangeListener(android.view.View.OnLayoutChangeListener)) **(View.OnLayoutChangeListener** (/reference/android/view/View.OnLayoutChangeListener) **listener)**<br><br>Remove a listener for layout changes. |
| **void** | **removeOnUnhandledKeyEventListener** (/reference/android/view/View#removeOnUnhandledKeyEventListener(android.view.View.OnUnhandledKeyEventListener)) **(View.OnUnhandledKeyEventListener** (/reference/android/view/View.OnUnhandledKeyEventListener) **listener)**<br><br>Removes a listener which will receive unhandled **KeyEvent** (/reference/android/view/KeyEvent)s. |
| **void** | **requestApplyInsets** (/reference/android/view/View#requestApplyInsets())**()**<br><br>Ask that a new dispatch of **onApplyWindowInsets(android.view.WindowInsets)** (/reference/android/view/View#onApplyWindowInsets(android.view.WindowInsets)) be performed. |

| | |
|---|---|
| `void` | **`requestFitSystemWindows`** (/reference/android/view/View#requestFitSystemWindows())**`()`**<br><br>*This method was deprecated in API level 20. Use* **`requestApplyInsets()`** (/reference/android/view/View#requestApplyInsets()) *for newer platform versions.* |
| `final boolean` | **`requestFocus`** (/reference/android/view/View#requestFocus(int))**`(int direction)`**<br><br>Call this to try to give focus to a specific view or to one of its descendants and give it a hint about what direction focus is heading. |
| `final boolean` | **`requestFocus`** (/reference/android/view/View#requestFocus())**`()`**<br><br>Call this to try to give focus to a specific view or to one of its descendants. |
| `boolean` | **`requestFocus`** (/reference/android/view/View#requestFocus(int,%20android.graphics.Rect))**`(int direction, Rect`** (/reference/android/graphics/Rect) **`previouslyFocusedRect)`**<br><br>Call this to try to give focus to a specific view or to one of its descendants and give it hints about the direction and a specific rectangle that the focus is coming from. |
| `final boolean` | **`requestFocusFromTouch`** (/reference/android/view/View#requestFocusFromTouch())**`()`**<br><br>Call this to try to give focus to a specific view or to one of its descendants. |
| `void` | **`requestLayout`** (/reference/android/view/View#requestLayout())**`()`**<br><br>Call this when something has changed which has invalidated the layout of this view. |
| `void` | **`requestPointerCapture`** (/reference/android/view/View#requestPointerCapture())**`()`** |

| | |
|---|---|
| | Requests pointer capture mode. |
| `boolean` | **requestRectangleOnScreen** (/reference/android/view/View#requestRectangleOnScreen(android.graphics.Rect))**(Rect** (/reference/android/graphics/Rect) **rectangle)**<br><br>Request that a rectangle of this view be visible on the screen, scrolling if necessary just enough. |
| `boolean` | **requestRectangleOnScreen** (/reference/android/view/View#requestRectangleOnScreen(android.graphics.Rect,%20boolean))**(Rect** (/reference/android/graphics/Rect) **rectangle, boolean immediate)**<br><br>Request that a rectangle of this view be visible on the screen, scrolling if necessary just enough. |
| `final void` | **requestUnbufferedDispatch** (/reference/android/view/View#requestUnbufferedDispatch(android.view.MotionEvent))**(MotionEvent** (/reference/android/view/MotionEvent) **event)**<br><br>Request unbuffered dispatch of the given stream of MotionEvents to this View. |
| `final <T extends View` (/reference/android/view/View)`> T` | **requireViewById** (/reference/android/view/View#requireViewById(int))**(int id)**<br><br>Finds the first descendant view with the given ID, the view itself if the ID matches **getId()** (/reference/android/view/View#getId()), or throws an IllegalArgumentException if the ID is invalid or there is no matching view in the hierarchy. |
| `void` | **resetPivot** (/reference/android/view/View#resetPivot())**()**<br><br>Clears any pivot previously set by a call to **setPivotX(float)** (/reference/android/view/View#setPivotX(float)) or **setPivotY(float)** |

| | |
|---|---|
| | (/reference/android/view/View#setPivotY(float)). |
| `static int` | **resolveSize** (/reference/android/view/View#resolveSize(int,%20int))**(int size, int measureSpec)**<br><br>Version of **resolveSizeAndState(int, int, int)** (/reference/android/view/View#resolveSizeAndState(int,%20int,%20int)) returning only the **MEASURED_SIZE_MASK** (/reference/android/view/View#MEASURED_SIZE_MASK) bits of the result. |
| `static int` | **resolveSizeAndState** (/reference/android/view/View#resolveSizeAndState(int,%20int,%20int))**(int size, int measureSpec, int childMeasuredState)**<br><br>Utility to reconcile a desired size and state, with constraints imposed by a MeasureSpec. |
| `boolean` | **restoreDefaultFocus** (/reference/android/view/View#restoreDefaultFocus())**()**<br><br>Gives focus to the default-focus view in the view hierarchy that has this view as a root. |
| `void` | **restoreHierarchyState** (/reference/android/view/View#restoreHierarchyState(android.util.SparseArray<android.os.Parcelable>))<br>**(SparseArray** (/reference/android/util/SparseArray)**<Parcelable** (/reference/android/os/Parcelable)**> container)**<br><br>Restore this view hierarchy's frozen state from the given container. |
| `final void` | **saveAttributeDataForStyleable** (/reference/android/view/View#saveAttributeDataForStyleable(android.content.Context,%20int[],%20android.util.AttributeSet,%20android.content.res.TypedArray,%20int,%20int))<br>**(Context** (/reference/android/content/Context) **context, int[] styleable, AttributeSet** (/reference/android/util/AttributeSet) **attrs, TypedArray** |

| | |
|---|---|
| | (/reference/android/content/res/TypedArray) **`t, int defStyleAttr, int defStyleRes)`**<br><br>Stores debugging information about attributes. |
| **`void`** | **`saveHierarchyState`**<br>(/reference/android/view/View#saveHierarchyState(android.util.SparseArray<android.os.Parcelable>))<br>**`(`**`SparseArray` (/reference/android/util/SparseArray)**`<`**`Parcelable`<br>(/reference/android/os/Parcelable)**`> container)`**<br><br>Store this view hierarchy's frozen state into the given container. |
| **`void`** | **`scheduleDrawable`**<br>(/reference/android/view/View#scheduleDrawable(android.graphics.drawable.Drawable,%20java.lang.Runnable,%20long))<br>**`(`**`Drawable` (/reference/android/graphics/drawable/Drawable) **`who,`** `Runnable`<br>(/reference/java/lang/Runnable) **`what, long when)`**<br><br>Schedules an action on a drawable to occur at a specified time. |
| **`void`** | `scrollBy` (/reference/android/view/View#scrollBy(int,%20int))**`(int x, int y)`**<br><br>Move the scrolled position of your view. |
| **`void`** | `scrollTo` (/reference/android/view/View#scrollTo(int,%20int))**`(int x, int y)`**<br><br>Set the scrolled position of your view. |
| **`void`** | **`sendAccessibilityEvent`**<br>(/reference/android/view/View#sendAccessibilityEvent(int))**`(int eventType)`**<br><br>Sends an accessibility event of the given type. |

| | |
|---|---|
| `void` | **`sendAccessibilityEventUnchecked`** (/reference/android/view/View#sendAccessibilityEventUnchecked(android.view.accessibility.AccessibilityEvent)) **`(AccessibilityEvent`** (/reference/android/view/accessibility/AccessibilityEvent) **`event)`**<br><br>This method behaves exactly as **`sendAccessibilityEvent(int)`** (/reference/android/view/View#sendAccessibilityEvent(int)) but takes as an argument an empty **`AccessibilityEvent`** (/reference/android/view/accessibility/AccessibilityEvent) and does not perform a check whether accessibility is enabled. |
| `void` | **`setAccessibilityDelegate`** (/reference/android/view/View#setAccessibilityDelegate(android.view.View.AccessibilityDelegate)) **`(View.AccessibilityDelegate`** (/reference/android/view/View.AccessibilityDelegate) **`delegate)`**<br><br>Sets a delegate for implementing accessibility support via composition (as opposed to inheritance). |
| `void` | **`setAccessibilityHeading`** (/reference/android/view/View#setAccessibilityHeading(boolean))**`(boolean isHeading)`**<br><br>Set if view is a heading for a section of content for accessibility purposes. |
| `void` | **`setAccessibilityLiveRegion`** (/reference/android/view/View#setAccessibilityLiveRegion(int))**`(int mode)`**<br><br>Sets the live region mode for this view. |
| `void` | **`setAccessibilityPaneTitle`** |

| | |
|---|---|
| | (/reference/android/view/View#setAccessibilityPaneTitle(java.lang.CharSequence)) (**CharSequence** (/reference/java/lang/CharSequence) **accessibilityPaneTitle**)<br><br>Visually distinct portion of a window with window-like semantics are considered panes for accessibility purposes. |
| **void** | **setAccessibilityTraversalAfter** (/reference/android/view/View#setAccessibilityTraversalAfter(int))(**int afterId**)<br><br>Sets the id of a view after which this one is visited in accessibility traversal. |
| **void** | **setAccessibilityTraversalBefore** (/reference/android/view/View#setAccessibilityTraversalBefore(int))(**int beforeId**)<br><br>Sets the id of a view before which this one is visited in accessibility traversal. |
| **void** | **setActivated** (/reference/android/view/View#setActivated(boolean))(**boolean activated**)<br><br>Changes the activated state of this view. |
| **void** | **setAlpha** (/reference/android/view/View#setAlpha(float))(**float alpha**)<br><br>Sets the opacity of the view to a value from 0 to 1, where 0 means the view is completely transparent and 1 means the view is completely opaque. |
| **void** | **setAnimation** (/reference/android/view/View#setAnimation(android.view.animation.Animation)) (**Animation** (/reference/android/view/animation/Animation) **animation**)<br><br>Sets the next animation to play for this view. |
| **void** | **setAnimationMatrix** |

| | |
|---|---|
| | (/reference/android/view/View#setAnimationMatrix(android.graphics.Matrix))(**Matrix** (/reference/android/graphics/Matrix) **matrix**)<br><br>Changes the transformation matrix on the view. |
| **void** | **setAutofillHints** (/reference/android/view/View#setAutofillHints(java.lang.String...))(**String...** (/reference/java/lang/String) **autofillHints**)<br><br>Sets the hints that help an **AutofillService** (/reference/android/service/autofill/AutofillService) determine how to autofill the view with the user's data. |
| **void** | **setAutofillId** (/reference/android/view/View#setAutofillId(android.view.autofill.AutofillId))(**AutofillId** (/reference/android/view/autofill/AutofillId) **id**)<br><br>Sets the unique, logical identifier of this view in the activity, for autofill purposes. |
| **void** | **setBackground** (/reference/android/view/View#setBackground(android.graphics.drawable.Drawable))(**Drawable** (/reference/android/graphics/drawable/Drawable) **background**)<br><br>Set the background to a given Drawable, or remove the background. |
| **void** | **setBackgroundColor** (/reference/android/view/View#setBackgroundColor(int))(**int color**)<br><br>Sets the background color for this view. |
| **void** | **setBackgroundDrawable** (/reference/android/view/View#setBackgroundDrawable(android.graphics.drawable.Drawable))(**Drawable** (/reference/android/graphics/drawable/Drawable) **background**) |

| | |
|---|---|
| | *This method was deprecated in API level 16. use* **setBackground(android.graphics.drawable.Drawable)** (/reference/android/view/View#setBackground(android.graphics.drawable.Drawable)) *instead* |
| **void** | **setBackgroundResource** (/reference/android/view/View#setBackgroundResource(int))**(int resid)**<br><br>Set the background to a given resource. |
| **void** | **setBackgroundTintBlendMode** (/reference/android/view/View#setBackgroundTintBlendMode(android.graphics.BlendMode))<br>**(BlendMode** (/reference/android/graphics/BlendMode) **blendMode)**<br><br>Specifies the blending mode used to apply the tint specified by **setBackgroundTintList(android.content.res.ColorStateList)** (/reference/android/view/View#setBackgroundTintList(android.content.res.ColorStateList))<br>} to the background drawable. |
| **void** | **setBackgroundTintList** (/reference/android/view/View#setBackgroundTintList(android.content.res.ColorStateList))<br>**(ColorStateList** (/reference/android/content/res/ColorStateList) **tint)**<br><br>Applies a tint to the background drawable. |
| **void** | **setBackgroundTintMode** (/reference/android/view/View#setBackgroundTintMode(android.graphics.PorterDuff.Mode))<br>**(PorterDuff.Mode** (/reference/android/graphics/PorterDuff.Mode) **tintMode)** |

| | |
|---|---|
| | Specifies the blending mode used to apply the tint specified by **setBackgroundTintList(android.content.res.ColorStateList)** (/reference/android/view/View#setBackgroundTintList(android.content.res.ColorStateList)) } to the background drawable. |
| **final void** | **setBottom** (/reference/android/view/View#setBottom(int))**(int bottom)**<br><br>Sets the bottom position of this view relative to its parent. |
| **void** | **setCameraDistance** (/reference/android/view/View#setCameraDistance(float))**(float distance)**<br><br>Sets the distance along the Z axis (orthogonal to the X/Y plane on which views are drawn) from the camera to this view. |
| **void** | **setClickable** (/reference/android/view/View#setClickable(boolean))**(boolean clickable)**<br><br>Enables or disables click events for this view. |
| **void** | **setClipBounds** (/reference/android/view/View#setClipBounds(android.graphics.Rect))**(Rect** (/reference/android/graphics/Rect) **clipBounds)**<br><br>Sets a rectangular area on this view to which the view will be clipped when it is drawn. |
| **void** | **setClipToOutline** (/reference/android/view/View#setClipToOutline(boolean))**(boolean clipToOutline)**<br><br>Sets whether the View's Outline should be used to clip the contents of the View. |
| **void** | **setContentCaptureSession** |

| | |
|---|---|
| | (/reference/android/view/View#setContentCaptureSession(android.view.contentcapture.ContentCaptureSession))<br>**(ContentCaptureSession** (/reference/android/view/contentcapture/ContentCaptureSession) **contentCaptureSession)**<br><br>Sets the (optional) **ContentCaptureSession** (/reference/android/view/contentcapture/ContentCaptureSession) associated with this view. |
| **void** | **setContentDescription** (/reference/android/view/View#setContentDescription(java.lang.CharSequence))<br>**(CharSequence** (/reference/java/lang/CharSequence) **contentDescription)**<br><br>Sets the **View** (/reference/android/view/View)'s content description. |
| **void** | **setContextClickable** (/reference/android/view/View#setContextClickable(boolean))<br>**(boolean contextClickable)**<br><br>Enables or disables context clicking for this view. |
| **void** | **setDefaultFocusHighlightEnabled** (/reference/android/view/View#setDefaultFocusHighlightEnabled(boolean))**(boolean defaultFocusHighlightEnabled)**<br><br>Sets whether this View should use a default focus highlight when it gets focused but doesn't have **R.attr.state_focused** (/reference/android/R.attr#state_focused) defined in its background. |
| **void** | **setDrawingCacheBackgroundColor** (/reference/android/view/View#setDrawingCacheBackgroundColor(int))**(int color)**<br><br>*This method was deprecated in API level 28. The view drawing cache was largely made* |

*obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations,* **setLayerType(int, android.graphics.Paint)** (/reference/android/view/View#setLayerType(int,%20android.graphics.Paint)) *handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a* **Canvas** (/reference/android/graphics/Canvas) *from either a* **Bitmap** (/reference/android/graphics/Bitmap) *or* **Picture** (/reference/android/graphics/Picture) *and call* **draw(android.graphics.Canvas)** (/reference/android/view/View#draw(android.graphics.Canvas)) *on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as* **Config.HARDWARE** (/reference/android/graphics/Bitmap.Config#HARDWARE) *bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the* **PixelCopy** (/reference/android/view/PixelCopy) *API is recommended.*

---

**void**

**setDrawingCacheEnabled** (/reference/android/view/View#setDrawingCacheEnabled(boolean))**(boolean enabled)**

*This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations,* **setLayerType(int, android.graphics.Paint)** (/reference/android/view/View#setLayerType(int,%20android.graphics.Paint)) *handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a* **Canvas** (/reference/android/graphics/Canvas) *from either a* **Bitmap**

(/reference/android/graphics/Bitmap) *or* **Picture** (/reference/android/graphics/Picture) *and call* **draw(android.graphics.Canvas)** (/reference/android/view/View#draw(android.graphics.Canvas)) *on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as* **Config.HARDWARE** (/reference/android/graphics/Bitmap.Config#HARDWARE) *bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the* **PixelCopy** (/reference/android/view/PixelCopy) *API is recommended.*

---

**void**

**setDrawingCacheQuality** (/reference/android/view/View#setDrawingCacheQuality(int))**(int quality)**

*This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations,* **setLayerType(int, android.graphics.Paint)** (/reference/android/view/View#setLayerType(int,%20android.graphics.Paint)) *handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a* **Canvas** (/reference/android/graphics/Canvas) *from either a* **Bitmap** (/reference/android/graphics/Bitmap) *or* **Picture** (/reference/android/graphics/Picture) *and call* **draw(android.graphics.Canvas)** (/reference/android/view/View#draw(android.graphics.Canvas)) *on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as* **Config.HARDWARE** (/reference/android/graphics/Bitmap.Config#HARDWARE) *bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the* **PixelCopy** (/reference/android/view/PixelCopy) *API is recommended.*

---

| | |
|---|---|
| **void** | **setDuplicateParentStateEnabled** (/reference/android/view/View#setDuplicateParentStateEnabled(boolean))**(boolean enabled)**<br><br>Enables or disables the duplication of the parent's state into this view. |
| **void** | **setElevation** (/reference/android/view/View#setElevation(float))**(float elevation)**<br><br>Sets the base elevation of this view, in pixels. |
| **void** | **setEnabled** (/reference/android/view/View#setEnabled(boolean))**(boolean enabled)**<br><br>Set the enabled state of this view. |
| **void** | **setFadingEdgeLength** (/reference/android/view/View#setFadingEdgeLength(int))**(int length)**<br><br>Set the size of the faded edge used to indicate that more content in this view is available. |
| **void** | **setFilterTouchesWhenObscured** (/reference/android/view/View#setFilterTouchesWhenObscured(boolean))**(boolean enabled)**<br><br>Sets whether the framework should discard touches when the view's window is obscured by another visible window. |
| **void** | **setFitsSystemWindows** (/reference/android/view/View#setFitsSystemWindows(boolean))**(boolean fitSystemWindows)**<br><br>Sets whether or not this view should account for system screen decorations such as the |

| | |
|---|---|
| | status bar and inset its content; that is, controlling whether the default implementation of **`fitSystemWindows(android.graphics.Rect)`** (/reference/android/view/View#fitSystemWindows(android.graphics.Rect)) will be executed. |
| **`void`** | **`setFocusable`** (/reference/android/view/View#setFocusable(boolean))**`(boolean focusable)`**<br><br>Set whether this view can receive the focus. |
| **`void`** | **`setFocusable`** (/reference/android/view/View#setFocusable(int))**`(int focusable)`**<br><br>Sets whether this view can receive focus. |
| **`void`** | **`setFocusableInTouchMode`** (/reference/android/view/View#setFocusableInTouchMode(boolean))**`(boolean focusableInTouchMode)`**<br><br>Set whether this view can receive focus while in touch mode. |
| **`void`** | **`setFocusedByDefault`** (/reference/android/view/View#setFocusedByDefault(boolean))**`(boolean isFocusedByDefault)`**<br><br>Sets whether this View should receive focus when the focus is restored for the view hierarchy containing this view. |
| **`void`** | **`setForceDarkAllowed`** (/reference/android/view/View#setForceDarkAllowed(boolean))**`(boolean allow)`**<br><br>Sets whether or not to allow force dark to apply to this view. |
| **`void`** | **`setForeground`** |

| | |
|---|---|
| | (/reference/android/view/View#setForeground(android.graphics.drawable.Drawable))<br>**(Drawable** (/reference/android/graphics/drawable/Drawable) **foreground)**<br><br>Supply a Drawable that is to be rendered on top of all of the content in the view. |
| **void** | **setForegroundGravity** (/reference/android/view/View#setForegroundGravity(int))<br>**(int gravity)**<br><br>Describes how the foreground is positioned. |
| **void** | **setForegroundTintBlendMode**<br>(/reference/android/view/View#setForegroundTintBlendMode(android.graphics.BlendMode))<br>**(BlendMode** (/reference/android/graphics/BlendMode) **blendMode)**<br><br>Specifies the blending mode used to apply the tint specified by<br>**setForegroundTintList(android.content.res.ColorStateList)**<br>(/reference/android/view/View#setForegroundTintList(android.content.res.ColorStateList))<br>} to the background drawable. |
| **void** | **setForegroundTintList**<br>(/reference/android/view/View#setForegroundTintList(android.content.res.ColorStateList))<br>**(ColorStateList** (/reference/android/content/res/ColorStateList) **tint)**<br><br>Applies a tint to the foreground drawable. |
| **void** | **setForegroundTintMode**<br>(/reference/android/view/View#setForegroundTintMode(android.graphics.PorterDuff.Mode))<br>**(PorterDuff.Mode** (/reference/android/graphics/PorterDuff.Mode) **tintMode)** |

| | |
|---|---|
| | Specifies the blending mode used to apply the tint specified by **setForegroundTintList(android.content.res.ColorStateList)** (/reference/android/view/View#setForegroundTintList(android.content.res.ColorStateList)) } to the background drawable. |
| **void** | **setHapticFeedbackEnabled** (/reference/android/view/View#setHapticFeedbackEnabled(boolean))**(boolean hapticFeedbackEnabled)**<br><br>Set whether this view should have haptic feedback for events such as long presses. |
| **void** | **setHasTransientState** (/reference/android/view/View#setHasTransientState(boolean))**(boolean hasTransientState)**<br><br>Set whether this view is currently tracking transient state that the framework should attempt to preserve when possible. |
| **void** | **setHorizontalFadingEdgeEnabled** (/reference/android/view/View#setHorizontalFadingEdgeEnabled(boolean))**(boolean horizontalFadingEdgeEnabled)**<br><br>Define whether the horizontal edges should be faded when this view is scrolled horizontally. |
| **void** | **setHorizontalScrollBarEnabled** (/reference/android/view/View#setHorizontalScrollBarEnabled(boolean))**(boolean horizontalScrollBarEnabled)**<br><br>Define whether the horizontal scrollbar should be drawn or not. |

| | |
|---|---|
| **void** | **setHorizontalScrollbarThumbDrawable** (/reference/android/view/View#setHorizontalScrollbarThumbDrawable(android.graphics.drawable.Drawable)) (**Drawable** (/reference/android/graphics/drawable/Drawable) **drawable**)<br><br>Defines the horizontal thumb drawable |
| **void** | **setHorizontalScrollbarTrackDrawable** (/reference/android/view/View#setHorizontalScrollbarTrackDrawable(android.graphics.drawable.Drawable)) (**Drawable** (/reference/android/graphics/drawable/Drawable) **drawable**)<br><br>Defines the horizontal track drawable |
| **void** | **setHovered** (/reference/android/view/View#setHovered(boolean))(**boolean hovered**)<br><br>Sets whether the view is currently hovered. |
| **void** | **setId** (/reference/android/view/View#setId(int))(**int id**)<br><br>Sets the identifier for this view. |
| **void** | **setImportantForAccessibility** (/reference/android/view/View#setImportantForAccessibility(int))(**int mode**)<br><br>Sets how to determine whether this view is important for accessibility which is if it fires accessibility events and if it is reported to accessibility services that query the screen. |
| **void** | **setImportantForAutofill** (/reference/android/view/View#setImportantForAutofill(int))(**int mode**)<br><br>Sets the mode for determining whether this view is considered important for autofill. |

| | |
|---|---|
| void | setImportantForContentCapture (/reference/android/view/View#setImportantForContentCapture(int))(int mode)<br><br>Sets the mode for determining whether this view is considered important for content capture. |
| **void** | **setKeepScreenOn** (/reference/android/view/View#setKeepScreenOn(boolean))**(boolean keepScreenOn)**<br><br>Controls whether the screen should remain on, modifying the value of **KEEP_SCREEN_ON** (/reference/android/view/View#KEEP_SCREEN_ON). |
| **void** | **setKeyboardNavigationCluster** (/reference/android/view/View#setKeyboardNavigationCluster(boolean))**(boolean isCluster)**<br><br>Set whether this view is a root of a keyboard navigation cluster. |
| **void** | **setLabelFor** (/reference/android/view/View#setLabelFor(int))**(int id)**<br><br>Sets the id of a view for which this view serves as a label for accessibility purposes. |
| **void** | **setLayerPaint** (/reference/android/view/View#setLayerPaint(android.graphics.Paint))**(Paint** (/reference/android/graphics/Paint) **paint)**<br><br>Updates the **Paint** (/reference/android/graphics/Paint) object used with the current layer (used only if the current layer type is not set to **LAYER_TYPE_NONE** (/reference/android/view/View#LAYER_TYPE_NONE)). |
| **void** | **setLayerType** (/reference/android/view/View#setLayerType(int,%20android.graphics.Paint))**(int layerType, Paint** (/reference/android/graphics/Paint) **paint)** |

| | |
|---|---|
| | Specifies the type of layer backing this view. |
| `void` | **setLayoutDirection** (/reference/android/view/View#setLayoutDirection(int))**(int layoutDirection)**<br><br>Set the layout direction for this view. |
| `void` | **setLayoutParams** (/reference/android/view/View#setLayoutParams(android.view.ViewGroup.LayoutParams))<br>**(ViewGroup.LayoutParams** (/reference/android/view/ViewGroup.LayoutParams) **params)**<br><br>Set the layout parameters associated with this view. |
| `final void` | **setLeft** (/reference/android/view/View#setLeft(int))**(int left)**<br><br>Sets the left position of this view relative to its parent. |
| `final void` | **setLeftTopRightBottom** (/reference/android/view/View#setLeftTopRightBottom(int,%20int,%20int,%20int))**(int left, int top, int right, int bottom)**<br><br>Assign a size and position to this view. |
| `void` | **setLongClickable** (/reference/android/view/View#setLongClickable(boolean))<br>**(boolean longClickable)**<br><br>Enables or disables long click events for this view. |
| `final void` | **setMeasuredDimension** (/reference/android/view/View#setMeasuredDimension(int,%20int))**(int measuredWidth, int measuredHeight)** |

| | |
|---|---|
| | This method must be called by **onMeasure(int, int)** (/reference/android/view/View#onMeasure(int,%20int)) to store the measured width and measured height. |
| **void** | **setMinimumHeight** (/reference/android/view/View#setMinimumHeight(int))**(int minHeight)**<br><br>Sets the minimum height of the view. |
| **void** | **setMinimumWidth** (/reference/android/view/View#setMinimumWidth(int))**(int minWidth)**<br><br>Sets the minimum width of the view. |
| **void** | **setNestedScrollingEnabled** (/reference/android/view/View#setNestedScrollingEnabled(boolean))**(boolean enabled)**<br><br>Enable or disable nested scrolling for this view. |
| **void** | **setNextClusterForwardId** (/reference/android/view/View#setNextClusterForwardId(int))**(int nextClusterForwardId)**<br><br>Sets the id of the view to use as the root of the next keyboard navigation cluster. |
| **void** | **setNextFocusDownId** (/reference/android/view/View#setNextFocusDownId(int))**(int nextFocusDownId)**<br><br>Sets the id of the view to use when the next focus is **FOCUS_DOWN** (/reference/android/view/View#FOCUS_DOWN). |
| **void** | **setNextFocusForwardId** |

| | |
|---|---|
| | (/reference/android/view/View#setNextFocusForwardId(int))**(int nextFocusForwardId)**<br><br>Sets the id of the view to use when the next focus is **FOCUS_FORWARD** (/reference/android/view/View#FOCUS_FORWARD). |
| **void** | **setNextFocusLeftId** (/reference/android/view/View#setNextFocusLeftId(int))**(int nextFocusLeftId)**<br><br>Sets the id of the view to use when the next focus is **FOCUS_LEFT** (/reference/android/view/View#FOCUS_LEFT). |
| **void** | **setNextFocusRightId** (/reference/android/view/View#setNextFocusRightId(int))**(int nextFocusRightId)**<br><br>Sets the id of the view to use when the next focus is **FOCUS_RIGHT** (/reference/android/view/View#FOCUS_RIGHT). |
| **void** | **setNextFocusUpId** (/reference/android/view/View#setNextFocusUpId(int))**(int nextFocusUpId)**<br><br>Sets the id of the view to use when the next focus is **FOCUS_UP** (/reference/android/view/View#FOCUS_UP). |
| **void** | **setOnApplyWindowInsetsListener** (/reference/android/view/View#setOnApplyWindowInsetsListener(android.view.View.OnApplyWindowInsetsListener))**(View.OnApplyWindowInsetsListener** (/reference/android/view/View.OnApplyWindowInsetsListener) **listener)**<br><br>Set an **OnApplyWindowInsetsListener** (/reference/android/view/View.OnApplyWindowInsetsListener) to take over the policy for applying window insets to this view. |

| | |
|---|---|
| void | **setOnCapturedPointerListener** (/reference/android/view/View#setOnCapturedPointerListener(android.view.View.OnCapturedPointerListener)) (**View.OnCapturedPointerListener** (/reference/android/view/View.OnCapturedPointerListener) l)<br><br>Set a listener to receive callbacks when the pointer capture state of a view changes. |
| void | **setOnClickListener** (/reference/android/view/View#setOnClickListener(android.view.View.OnClickListener)) (**View.OnClickListener** (/reference/android/view/View.OnClickListener) l)<br><br>Register a callback to be invoked when this view is clicked. |
| void | **setOnContextClickListener** (/reference/android/view/View#setOnContextClickListener(android.view.View.OnContextClickListener)) (**View.OnContextClickListener** (/reference/android/view/View.OnContextClickListener) l)<br><br>Register a callback to be invoked when this view is context clicked. |
| void | **setOnCreateContextMenuListener** (/reference/android/view/View#setOnCreateContextMenuListener(android.view.View.OnCreateContextMenuListener)) (**View.OnCreateContextMenuListener** (/reference/android/view/View.OnCreateContextMenuListener) l)<br><br>Register a callback to be invoked when the context menu for this view is being built. |
| void | **setOnDragListener** (/reference/android/view/View#setOnDragListener(android.view.View.OnDragListener)) |

| | |
|---|---|
| | (View.OnDragListener (/reference/android/view/View.OnDragListener) l)<br><br>Register a drag event listener callback object for this View. |
| void | setOnFocusChangeListener (/reference/android/view/View#setOnFocusChangeListener(android.view.View.OnFocusChangeListener))<br>(View.OnFocusChangeListener (/reference/android/view/View.OnFocusChangeListener) l)<br><br>Register a callback to be invoked when focus of this view changed. |
| void | setOnGenericMotionListener (/reference/android/view/View#setOnGenericMotionListener(android.view.View.OnGenericMotionListener))<br>(View.OnGenericMotionListener (/reference/android/view/View.OnGenericMotionListener) l)<br><br>Register a callback to be invoked when a generic motion event is sent to this view. |
| void | setOnHoverListener (/reference/android/view/View#setOnHoverListener(android.view.View.OnHoverListener))<br>(View.OnHoverListener (/reference/android/view/View.OnHoverListener) l)<br><br>Register a callback to be invoked when a hover event is sent to this view. |
| void | setOnKeyListener (/reference/android/view/View#setOnKeyListener(android.view.View.OnKeyListener))<br>(View.OnKeyListener (/reference/android/view/View.OnKeyListener) l)<br><br>Register a callback to be invoked when a hardware key is pressed in this view. |

| | |
|---|---|
| void | **setOnLongClickListener** (/reference/android/view/View#setOnLongClickListener(android.view.View.OnLongClickListener)) (**View.OnLongClickListener** (/reference/android/view/View.OnLongClickListener) l)<br><br>Register a callback to be invoked when this view is clicked and held. |
| void | **setOnScrollChangeListener** (/reference/android/view/View#setOnScrollChangeListener(android.view.View.OnScrollChangeListener)) (**View.OnScrollChangeListener** (/reference/android/view/View.OnScrollChangeListener) l)<br><br>Register a callback to be invoked when the scroll X or Y positions of this view change. |
| void | **setOnSystemUiVisibilityChangeListener** (/reference/android/view/View#setOnSystemUiVisibilityChangeListener(android.view.View.OnSystemUiVisibilityChangeListener)) (**View.OnSystemUiVisibilityChangeListener** (/reference/android/view/View.OnSystemUiVisibilityChangeListener) l)<br><br>Set a listener to receive callbacks when the visibility of the system bar changes. |
| void | **setOnTouchListener** (/reference/android/view/View#setOnTouchListener(android.view.View.OnTouchListener)) (**View.OnTouchListener** (/reference/android/view/View.OnTouchListener) l)<br><br>Register a callback to be invoked when a touch event is sent to this view. |
| void | **setOutlineAmbientShadowColor** |

| | |
|---|---|
| | (/reference/android/view/View#setOutlineAmbientShadowColor(int))**(int color)**<br><br>Sets the color of the ambient shadow that is drawn when the view has a positive Z or elevation value. |
| **void** | **setOutlineProvider** (/reference/android/view/View#setOutlineProvider(android.view.ViewOutlineProvider))**(ViewOutlineProvider** (/reference/android/view/ViewOutlineProvider) **provider)**<br><br>Sets the **ViewOutlineProvider** (/reference/android/view/ViewOutlineProvider) of the view, which generates the Outline that defines the shape of the shadow it casts, and enables outline clipping. |
| **void** | **setOutlineSpotShadowColor** (/reference/android/view/View#setOutlineSpotShadowColor(int))**(int color)**<br><br>Sets the color of the spot shadow that is drawn when the view has a positive Z or elevation value. |
| **void** | **setOverScrollMode** (/reference/android/view/View#setOverScrollMode(int))**(int overScrollMode)**<br><br>Set the over-scroll mode for this view. |
| **void** | **setPadding** (/reference/android/view/View#setPadding(int,%20int,%20int,%20int))**(int left, int top, int right, int bottom)**<br><br>Sets the padding. |
| **void** | **setPaddingRelative** (/reference/android/view/View#setPaddingRelative(int,%20int,%20int,%20int))**(int start, int top, int end, int bottom)** |

| | |
|---|---|
| | Sets the relative padding. |
| `void` | **setPivotX** (/reference/android/view/View#setPivotX(float))**(float pivotX)**<br><br>Sets the x location of the point around which the view is **rotated** (/reference/android/view/View#setRotation(float)) and **scaled** (/reference/android/view/View#setScaleX(float)). |
| `void` | **setPivotY** (/reference/android/view/View#setPivotY(float))**(float pivotY)**<br><br>Sets the y location of the point around which the view is **rotated** (/reference/android/view/View#setRotation(float)) and **scaled** (/reference/android/view/View#setScaleY(float)). |
| `void` | **setPointerIcon** (/reference/android/view/View#setPointerIcon(android.view.PointerIcon)) **(PointerIcon** (/reference/android/view/PointerIcon) **pointerIcon)**<br><br>Set the pointer icon for the current view. |
| `void` | **setPressed** (/reference/android/view/View#setPressed(boolean))**(boolean pressed)**<br><br>Sets the pressed state for this view. |
| `final void` | **setRevealOnFocusHint** (/reference/android/view/View#setRevealOnFocusHint(boolean))**(boolean revealOnFocus)**<br><br>Sets this view's preference for reveal behavior when it gains focus. |
| `final void` | **setRight** (/reference/android/view/View#setRight(int))**(int right)** |

| | |
|---|---|
| | Sets the right position of this view relative to its parent. |
| **void** | **setRotation** (/reference/android/view/View#setRotation(float))**(float rotation)**<br><br>Sets the degrees that the view is rotated around the pivot point. |
| **void** | **setRotationX** (/reference/android/view/View#setRotationX(float))**(float rotationX)**<br><br>Sets the degrees that the view is rotated around the horizontal axis through the pivot point. |
| **void** | **setRotationY** (/reference/android/view/View#setRotationY(float))**(float rotationY)**<br><br>Sets the degrees that the view is rotated around the vertical axis through the pivot point. |
| **void** | **setSaveEnabled** (/reference/android/view/View#setSaveEnabled(boolean))**(boolean enabled)**<br><br>Controls whether the saving of this view's state is enabled (that is, whether its **onSaveInstanceState()** (/reference/android/view/View#onSaveInstanceState()) method will be called). |
| **void** | **setSaveFromParentEnabled** (/reference/android/view/View#setSaveFromParentEnabled(boolean))**(boolean enabled)**<br><br>Controls whether the entire hierarchy under this view will save its state when a state saving traversal occurs from its parent. |
| **void** | **setScaleX** (/reference/android/view/View#setScaleX(float))**(float scaleX)**<br><br>Sets the amount that the view is scaled in x around the pivot point, as a proportion of the |

| | |
|---|---|
| | view's unscaled width. |
| **void** | **setScaleY** (/reference/android/view/View#setScaleY(float))**(float scaleY)**<br><br>Sets the amount that the view is scaled in Y around the pivot point, as a proportion of the view's unscaled width. |
| **void** | **setScreenReaderFocusable** (/reference/android/view/View#setScreenReaderFocusable(boolean))**(boolean screenReaderFocusable)**<br><br>Sets whether this View should be a focusable element for screen readers and include non-focusable Views from its subtree when providing feedback. |
| **void** | **setScrollBarDefaultDelayBeforeFade** (/reference/android/view/View#setScrollBarDefaultDelayBeforeFade(int))**(int scrollBarDefaultDelayBeforeFade)**<br><br>Define the delay before scrollbars fade. |
| **void** | **setScrollBarFadeDuration** (/reference/android/view/View#setScrollBarFadeDuration(int))**(int scrollBarFadeDuration)**<br><br>Define the scrollbar fade duration. |
| **void** | **setScrollBarSize** (/reference/android/view/View#setScrollBarSize(int))**(int scrollBarSize)**<br><br>Define the scrollbar size. |
| **void** | **setScrollBarStyle** (/reference/android/view/View#setScrollBarStyle(int))**(int style)** |

| | |
|---|---|
| | Specify the style of the scrollbars. |
| **void** | **setScrollContainer** (/reference/android/view/View#setScrollContainer(boolean))**(boolean isScrollContainer)**<br><br>Change whether this view is one of the set of scrollable containers in its window. |
| **void** | **setScrollIndicators** (/reference/android/view/View#setScrollIndicators(int,%20int))**(int indicators, int mask)**<br><br>Sets the state of the scroll indicators specified by the mask. |
| **void** | **setScrollIndicators** (/reference/android/view/View#setScrollIndicators(int))**(int indicators)**<br><br>Sets the state of all scroll indicators. |
| **void** | **setScrollX** (/reference/android/view/View#setScrollX(int))**(int value)**<br><br>Set the horizontal scrolled position of your view. |
| **void** | **setScrollY** (/reference/android/view/View#setScrollY(int))**(int value)**<br><br>Set the vertical scrolled position of your view. |
| **void** | **setScrollbarFadingEnabled** (/reference/android/view/View#setScrollbarFadingEnabled(boolean))**(boolean fadeScrollbars)**<br><br>Define whether scrollbars will fade when the view is not scrolling. |
| **void** | **setSelected** (/reference/android/view/View#setSelected(boolean))**(boolean selected)** |

| | |
|---|---|
| | Changes the selection state of this view. |
| **void** | **setSoundEffectsEnabled** (/reference/android/view/View#setSoundEffectsEnabled(boolean))**(boolean soundEffectsEnabled)**<br><br>Set whether this view should have sound effects enabled for events such as clicking and touching. |
| void | setStateDescription (/reference/android/view/View#setStateDescription(java.lang.CharSequence))(CharSequence (/reference/java/lang/CharSequence) stateDescription)<br><br>Sets the **View** (/reference/android/view/View)'s state description. |
| **void** | **setStateListAnimator** (/reference/android/view/View#setStateListAnimator(android.animation.StateListAnimator))<br>**(StateListAnimator** (/reference/android/animation/StateListAnimator) **stateListAnimator)**<br><br>Attaches the provided StateListAnimator to this View. |
| **void** | **setSystemGestureExclusionRects** (/reference/android/view/View#setSystemGestureExclusionRects(java.util.List<android.graphics.Rect>))<br>**(List** (/reference/java/util/List)**<Rect** (/reference/android/graphics/Rect)**> rects)**<br><br>Sets a list of areas within this view's post-layout coordinate space where the system should not intercept touch or other pointing device gestures. |
| **void** | **setSystemUiVisibility** (/reference/android/view/View#setSystemUiVisibility(int)) |

| | |
|---|---|
| | **(int visibility)**<br><br>Request that the visibility of the status bar or other screen/window decorations be changed. |
| **void** | **setTag** (/reference/android/view/View#setTag(int,%20java.lang.Object))**(int key, Object** (/reference/java/lang/Object) **tag)**<br><br>Sets a tag associated with this view and a key. |
| **void** | **setTag** (/reference/android/view/View#setTag(java.lang.Object))**(Object** (/reference/java/lang/Object) **tag)**<br><br>Sets the tag associated with this view. |
| **void** | **setTextAlignment** (/reference/android/view/View#setTextAlignment(int))**(int textAlignment)**<br><br>Set the text alignment. |
| **void** | **setTextDirection** (/reference/android/view/View#setTextDirection(int))**(int textDirection)**<br><br>Set the text direction. |
| **void** | **setTooltipText** (/reference/android/view/View#setTooltipText(java.lang.CharSequence))**(CharSequence** (/reference/java/lang/CharSequence) **tooltipText)**<br><br>Sets the tooltip text which will be displayed in a small popup next to the view. |
| **final void** | **setTop** (/reference/android/view/View#setTop(int))**(int top)**<br><br>Sets the top position of this view relative to its parent. |

| | |
|---|---|
| `void` | **setTouchDelegate** (/reference/android/view/View#setTouchDelegate(android.view.TouchDelegate))(**TouchDelegate** (/reference/android/view/TouchDelegate) **delegate**)<br><br>Sets the TouchDelegate for this View. |
| `void` | **setTransitionAlpha** (/reference/android/view/View#setTransitionAlpha(float))(**float alpha**)<br><br>This property is intended only for use by the Fade transition, which animates it to produce a visual translucency that does not side-effect (or get affected by) the real alpha property. |
| `final void` | **setTransitionName** (/reference/android/view/View#setTransitionName(java.lang.String))(**String** (/reference/java/lang/String) **transitionName**)<br><br>Sets the name of the View to be used to identify Views in Transitions. |
| `void` | **setTransitionVisibility** (/reference/android/view/View#setTransitionVisibility(int))(**int visibility**)<br><br>Changes the visibility of this View without triggering any other changes. |
| `void` | **setTranslationX** (/reference/android/view/View#setTranslationX(float))(**float translationX**)<br><br>Sets the horizontal location of this view relative to its **left** (/reference/android/view/View#getLeft()) position. |
| `void` | **setTranslationY** (/reference/android/view/View#setTranslationY(float))(**float translationY**) |

| | |
|---|---|
| | Sets the vertical location of this view relative to its **top** (/reference/android/view/View#getTop()) position. |
| **void** | **setTranslationZ** (/reference/android/view/View#setTranslationZ(float))**(float translationZ)**<br><br>Sets the depth location of this view relative to its **elevation** (/reference/android/view/View#getElevation()). |
| **void** | **setVerticalFadingEdgeEnabled** (/reference/android/view/View#setVerticalFadingEdgeEnabled(boolean))**(boolean verticalFadingEdgeEnabled)**<br><br>Define whether the vertical edges should be faded when this view is scrolled vertically. |
| **void** | **setVerticalScrollBarEnabled** (/reference/android/view/View#setVerticalScrollBarEnabled(boolean))**(boolean verticalScrollBarEnabled)**<br><br>Define whether the vertical scrollbar should be drawn or not. |
| **void** | **setVerticalScrollbarPosition** (/reference/android/view/View#setVerticalScrollbarPosition(int))**(int position)**<br><br>Set the position of the vertical scroll bar. |
| **void** | **setVerticalScrollbarThumbDrawable** (/reference/android/view/View#setVerticalScrollbarThumbDrawable(android.graphics.drawable.Drawable))<br>**(Drawable** (/reference/android/graphics/drawable/Drawable) **drawable)**<br><br>Defines the vertical scrollbar thumb drawable |

| | |
|---|---|
| **void** | **setVerticalScrollbarTrackDrawable** (/reference/android/view/View#setVerticalScrollbarTrackDrawable(android.graphics.drawable.Drawable))**(Drawable** (/reference/android/graphics/drawable/Drawable) **drawable)**<br><br>Defines the vertical scrollbar track drawable |
| **void** | **setVisibility** (/reference/android/view/View#setVisibility(int))**(int visibility)**<br><br>Set the visibility state of this view. |
| **void** | **setWillNotCacheDrawing** (/reference/android/view/View#setWillNotCacheDrawing(boolean))**(boolean willNotCacheDrawing)**<br><br>*This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations,* **setLayerType(int, android.graphics.Paint)** (/reference/android/view/View#setLayerType(int,%20android.graphics.Paint)) *handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a* **Canvas** (/reference/android/graphics/Canvas) *from either a* **Bitmap** (/reference/android/graphics/Bitmap) *or* **Picture** (/reference/android/graphics/Picture) *and call* **draw(android.graphics.Canvas)** (/reference/android/view/View#draw(android.graphics.Canvas)) *on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as* **Config.HARDWARE** (/reference/android/graphics/Bitmap.Config#HARDWARE) *bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the* |

| | |
|---|---|
| | **PixelCopy** (/reference/android/view/PixelCopy) *API is recommended.* |
| **void** | **setWillNotDraw** (/reference/android/view/View#setWillNotDraw(boolean))**(boolean willNotDraw)**<br><br>If this view doesn't do any drawing on its own, set this flag to allow further optimizations. |
| void | setWindowInsetsAnimationCallback (/reference/android/view/View#setWindowInsetsAnimationCallback(android.view.WindowInsetsAnimationCallback))(WindowInsetsAnimationCallback (/reference/android/view/WindowInsetsAnimationCallback) listener)<br><br>Sets a **WindowInsetsAnimationCallback** (/reference/android/view/WindowInsetsAnimationCallback) to be notified about animations of windows that cause insets. |
| **void** | **setX** (/reference/android/view/View#setX(float))**(float x)**<br><br>Sets the visual x position of this view, in pixels. |
| **void** | **setY** (/reference/android/view/View#setY(float))**(float y)**<br><br>Sets the visual y position of this view, in pixels. |
| **void** | **setZ** (/reference/android/view/View#setZ(float))**(float z)**<br><br>Sets the visual z position of this view, in pixels. |
| **boolean** | **showContextMenu** (/reference/android/view/View#showContextMenu())**()**<br><br>Shows the context menu for this view. |
| **boolean** | **showContextMenu** (/reference/android/view/View#showContextMenu(float,%20float)) |

| | |
|---|---|
| | **(float x, float y)**<br><br>Shows the context menu for this view anchored to the specified view-relative coordinate. |
| **ActionMode** (/reference/android/view/ActionMode) | **startActionMode** (/reference/android/view/View#startActionMode(android.view.ActionMode.Callback,%20int)) **(ActionMode.Callback** (/reference/android/view/ActionMode.Callback) **callback, int type)**<br><br>Start an action mode with the given type. |
| **ActionMode** (/reference/android/view/ActionMode) | **startActionMode** (/reference/android/view/View#startActionMode(android.view.ActionMode.Callback)) **(ActionMode.Callback** (/reference/android/view/ActionMode.Callback) **callback)**<br><br>Start an action mode with the default type **ActionMode#TYPE_PRIMARY** (/reference/android/view/ActionMode#TYPE_PRIMARY). |
| **void** | **startAnimation** (/reference/android/view/View#startAnimation(android.view.animation.Animation)) **(Animation** (/reference/android/view/animation/Animation) **animation)**<br><br>Start the specified animation now. |
| **final boolean** | **startDrag** (/reference/android/view/View#startDrag(android.content.ClipData,%20android.view.View.DragShadowBuilder,%20java.lang.Object,%20int)) **(ClipData** (/reference/android/content/ClipData) **data, View.DragShadowBuilder** (/reference/android/view/View.DragShadowBuilder) **shadowBuilder, Object** (/reference/java/lang/Object) **myLocalState, int flags)** |

| | |
|---|---|
| | *This method was deprecated in API level 24. Use* **startDragAndDrop()** (/reference/android/view/View#startDragAndDrop(android.content.ClipData,%20android.view.View.DragShadowBuilder,%20java.lang.Object,%20int)) *for newer platform versions.* |
| **final boolean** | **startDragAndDrop** (/reference/android/view/View#startDragAndDrop(android.content.ClipData,%20android.view.View.DragShadowBuilder,%20java.lang.Object,%20int)) **(ClipData** (/reference/android/content/ClipData) **data, View.DragShadowBuilder** (/reference/android/view/View.DragShadowBuilder) **shadowBuilder, Object** (/reference/java/lang/Object) **myLocalState, int flags)**<br><br>Starts a drag and drop operation. |
| **boolean** | **startNestedScroll** (/reference/android/view/View#startNestedScroll(int))**(int axes)**<br><br>Begin a nestable scroll operation along the given axes. |
| **void** | **stopNestedScroll** (/reference/android/view/View#stopNestedScroll())**()**<br><br>Stop a nested scroll in progress. |
| **String** (/reference/java/lang/String) | **toString** (/reference/android/view/View#toString())**()**<br><br>Returns a string representation of the object. |
| **void** | **transformMatrixToGlobal** (/reference/android/view/View#transformMatrixToGlobal(android.graphics.Matrix)) **(Matrix** (/reference/android/graphics/Matrix) **matrix)**<br><br>Modifies the input matrix such that it maps view-local coordinates to on-screen |

| | |
|---|---|
| | coordinates. |
| `void` | **transformMatrixToLocal** (/reference/android/view/View#transformMatrixToLocal(android.graphics.Matrix)) (**Matrix** (/reference/android/graphics/Matrix) **matrix**)<br><br>Modifies the input matrix such that it maps on-screen coordinates to view-local coordinates. |
| `void` | **unscheduleDrawable** (/reference/android/view/View#unscheduleDrawable(android.graphics.drawable.Drawable,%20java.lang.Runnable)) (**Drawable** (/reference/android/graphics/drawable/Drawable) **who**, **Runnable** (/reference/java/lang/Runnable) **what**)<br><br>Cancels a scheduled action on a drawable. |
| `void` | **unscheduleDrawable** (/reference/android/view/View#unscheduleDrawable(android.graphics.drawable.Drawable)) (**Drawable** (/reference/android/graphics/drawable/Drawable) **who**)<br><br>Unschedule any events associated with the given Drawable. |
| `final void` | **updateDragShadow** (/reference/android/view/View#updateDragShadow(android.view.View.DragShadowBuilder)) (**View.DragShadowBuilder** (/reference/android/view/View.DragShadowBuilder) **shadowBuilder**)<br><br>Updates the drag shadow for the ongoing drag and drop operation. |
| `boolean` | **verifyDrawable** |

(/reference/android/view/View#verifyDrawable(android.graphics.drawable.Drawable))
(**Drawable** (/reference/android/graphics/drawable/Drawable) **who**)

If your view subclass is displaying its own Drawable objects, it should override this function and return true for any Drawable it is displaying.

---

**boolean**

**willNotCacheDrawing** (/reference/android/view/View#willNotCacheDrawing())()

*This method was deprecated in API level 28. The view drawing cache was largely made obsolete with the introduction of hardware-accelerated rendering in API 11. With hardware-acceleration, intermediate cache layers are largely unnecessary and can easily result in a net loss in performance due to the cost of creating and updating the layer. In the rare cases where caching layers are useful, such as for alpha animations,* **setLayerType(int, android.graphics.Paint)** (/reference/android/view/View#setLayerType(int,%20android.graphics.Paint)) *handles this with hardware rendering. For software-rendered snapshots of a small part of the View hierarchy or individual Views it is recommended to create a* **Canvas** (/reference/android/graphics/Canvas) *from either a* **Bitmap** (/reference/android/graphics/Bitmap) *or* **Picture** (/reference/android/graphics/Picture) *and call* **draw(android.graphics.Canvas)** (/reference/android/view/View#draw(android.graphics.Canvas)) *on the View. However these software-rendered usages are discouraged and have compatibility issues with hardware-only rendering features such as* **Config.HARDWARE** (/reference/android/graphics/Bitmap.Config#HARDWARE) *bitmaps, real-time shadows, and outline clipping. For screenshots of the UI for feedback reports or unit testing the* **PixelCopy** (/reference/android/view/PixelCopy) *API is recommended.*

---

**boolean**

**willNotDraw** (/reference/android/view/View#willNotDraw())()

Returns whether or not this View draws on its own.

---

From class **java.lang.Object** (/reference/java/lang/Object)

| | |
|---|---|
| `Object` (/reference/java/lang/Object) | `clone` (/reference/java/lang/Object#clone())`()`<br>Creates and returns a copy of this object. |
| `boolean` | `equals` (/reference/java/lang/Object#equals(java.lang.Object))`(Object` (/reference/java/lang/Object) `obj)`<br>Indicates whether some other object is "equal to" this one. |
| `void` | `finalize` (/reference/java/lang/Object#finalize())`()`<br>Called by the garbage collector on an object when garbage collection determines that there are no more references to the object. |
| `final Class` (/reference/java/lang/Class)`<?>` | `getClass` (/reference/java/lang/Object#getClass())`()`<br>Returns the runtime class of this `Object`. |
| `int` | `hashCode` (/reference/java/lang/Object#hashCode())`()`<br>Returns a hash code value for the object. |
| `final void` | `notify` (/reference/java/lang/Object#notify())`()`<br>Wakes up a single thread that is waiting on this object's monitor. |
| `final void` | `notifyAll` (/reference/java/lang/Object#notifyAll())`()`<br>Wakes up all threads that are waiting on this object's monitor. |
| `String` (/reference/java/lang/String) | `toString` (/reference/java/lang/Object#toString())`()`<br>Returns a string representation of the object. |
| `final void` | `wait` (/reference/java/lang/Object#wait(long,%20int))`(long timeout, int nanos)` |

| | |
|---|---|
| | Causes the current thread to wait until another thread invokes the **notify()** (/reference/java/lang/Object#notify()) method or the **notifyAll()** (/reference/java/lang/Object#notifyAll()) method for this object, or some other thread interrupts the current thread, or a certain amount of real time has elapsed. |
| `final void` | **wait** (/reference/java/lang/Object#wait(long))**(long timeout)**<br><br>Causes the current thread to wait until either another thread invokes the **notify()** (/reference/java/lang/Object#notify()) method or the **notifyAll()** (/reference/java/lang/Object#notifyAll()) method for this object, or a specified amount of time has elapsed. |
| `final void` | **wait** (/reference/java/lang/Object#wait())**()**<br><br>Causes the current thread to wait until another thread invokes the **notify()** (/reference/java/lang/Object#notify()) method or the **notifyAll()** (/reference/java/lang/Object#notifyAll()) method for this object. |

From interface **android.graphics.drawable.Drawable.Callback** (/reference/android/graphics/drawable/Drawable.Callback)

| | |
|---|---|
| `abstract void` | **invalidateDrawable** (/reference/android/graphics/drawable/Drawable.Callback#invalidateDrawable(android.graphics.drawable.Drawable))**(Drawable** (/reference/android/graphics/drawable/Drawable) **who)**<br><br>Called when the drawable needs to be redrawn. |
| `abstract void` | **scheduleDrawable** (/reference/android/graphics/drawable/Drawable.Callback#scheduleDrawable(android.graphics.drawable.Drawable,%20java.lang.Runnable,%20long))**(Drawable** (/reference/android/graphics/drawable/Drawable) **who, Runnable** (/reference/java/lang/Runnable) **what, long when)**<br><br>A Drawable can call this to schedule the next frame of its animation. |
| `abstract void` | **unscheduleDrawable** (/reference/android/graphics/drawable/Drawable.Callback#unscheduleDrawable(android.graphics.drawable.Drawable,%20java.lang.Runnable))**(Drawable** (/reference/android/graphics/drawable/Drawable) **who, Runnable** (/reference/java/lang/Runnable) **what)**<br><br>A Drawable can call this to unschedule an action previously scheduled with **scheduleDrawable(Drawable, Runnable, long)** (/reference/android/graphics/drawable/Drawable.Callback#scheduleDrawable(android.graphics.drawable.Drawable,%20java.lang.Runnable,%20long)). |

From interface **android.view.KeyEvent.Callback** (/reference/android/view/KeyEvent.Callback)

| | |
|---|---|
| **abstract boolean** | **onKeyDown** (/reference/android/view/KeyEvent.Callback#onKeyDown(int,%20android.view.KeyEvent))**(int keyCode, KeyEvent** (/reference/android/view/KeyEvent) **event)**<br><br>Called when a key down event has occurred. |
| **abstract boolean** | **onKeyLongPress** (/reference/android/view/KeyEvent.Callback#onKeyLongPress(int,%20android.view.KeyEvent))**(int keyCode, KeyEvent** (/reference/android/view/KeyEvent) **event)**<br><br>Called when a long press has occurred. |
| **abstract boolean** | **onKeyMultiple** (/reference/android/view/KeyEvent.Callback#onKeyMultiple(int,%20int,%20android.view.KeyEvent))**(int keyCode, int count, KeyEvent** (/reference/android/view/KeyEvent) **event)**<br><br>Called when a user's interaction with an analog control, such as flinging a trackball, generates simulated down/up events for the same key multiple times in quick succession. |
| **abstract boolean** | **onKeyUp** (/reference/android/view/KeyEvent.Callback#onKeyUp(int,%20android.view.KeyEvent))**(int keyCode, KeyEvent** (/reference/android/view/KeyEvent) **event)**<br><br>Called when a key up event has occurred. |

From interface **android.view.accessibility.AccessibilityEventSource** (/reference/android/view/accessibility/AccessibilityEventSource)

| | |
|---|---|
| **abstract void** | **sendAccessibilityEvent** (/reference/android/view/accessibility/AccessibilityEventSource#sendAccessibilityEvent(int))**(int eventType)**<br><br>Handles the request for sending an **AccessibilityEvent** (/reference/android/view/accessibility/AccessibilityEvent) given the event type. |
| **abstract void** | **sendAccessibilityEventUnchecked** (/reference/android/view/accessibility/AccessibilityEventSource#sendAccessibilityEventUnchecked(android.view.accessibility.AccessibilityEvent))**(AccessibilityEvent** (/reference/android/view/accessibility/AccessibilityEvent) **event)** |

Handles the request for sending an **`AccessibilityEvent`** (/reference/android/view/accessibility/AccessibilityEvent).

From interface **`android.view.ViewTreeObserver.OnPreDrawListener`** (/reference/android/view/ViewTreeObserver.OnPreDrawListener)

| | |
|---|---|
| `abstract boolean` | **`onPreDraw`** (/reference/android/view/ViewTreeObserver.OnPreDrawListener#onPreDraw())**`()`** |

Callback method to be invoked when the view tree is about to be drawn.

# XML attributes

## android:allowUndo

Whether undo should be allowed for editable text. Defaults to true.

May be a boolean value, such as "**`true`**" or "**`false`**".

## android:autoLink

Controls whether links such as urls and email addresses are automatically found and converted to clickable links. The default value is "none", disabling this feature.

Must be one or more (separated by '|') of the following constant values.

| Constant | Value | Description |
|---|---|---|

| | | |
|---|---|---|
| all | f | Match all patterns (equivalent to web\|email\|phone\|map). |
| email | 2 | Match email addresses. |
| map | 8 | Match map addresses. Deprecated: see `Linkify.MAP_ADDRESSES` (/reference/android/text/util/Linkify#MAP_ADDRESSES). |
| none | 0 | Match no patterns (default). |
| phone | 4 | Match phone numbers. |
| web | 1 | Match Web URLs. |

**Related methods:**

`setAutoLinkMask(int)` (/reference/android/widget/TextView#setAutoLinkMask(int))

## android:autoSizeMaxTextSize

The maximum text size constraint to be used when auto-sizing text.

May be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), and mm (millimeters).

**Related methods:**

`setAutoSizeTextTypeUniformWithConfiguration(int,int,int,int)` (/reference/android/widget/TextView#setAutoSizeTextTypeUniformWithConfiguration(int,%20int,%20int,%20int))

## android:autoSizeMinTextSize

The minimum text size constraint to be used when auto-sizing text.

May be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), and mm (millimeters).

**Related methods:**

`setAutoSizeTextTypeUniformWithConfiguration(int,int,int,int)` (/reference/android/widget/TextView#setAutoSizeTextTypeUniformWithConfiguration(int,%20int,%20int,%20int))

## android:autoSizePresetSizes

Resource array of dimensions to be used in conjunction with `autoSizeTextType` set to `uniform`. Overrides `autoSizeStepGranularity` if set.

May be a reference to another resource, in the form "`@[+][`*`package`*`:]`*`type`*`/`*`name`*" or a theme attribute in the form "`?[`*`package`*`:]`*`type`*`/`*`name`*".

May be a string value, using '\\;' to escape characters such as '\\n' or '\\uxxxx' for a unicode character;

May be an integer value, such as "`100`".

May be a boolean value, such as "`true`" or "`false`".

May be a color value, in the form of "`#`*`rgb`*", "`#`*`argb`*", "`#`*`rrggbb`*", or "`#`*`aarrggbb`*".

May be a floating point value, such as "`1.2`".

May be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp

(scaled pixels based on preferred font size), in (inches), and mm (millimeters).

May be a fractional value, which is a floating point number appended with either % or %p, such as "`14.5%`". The % suffix always means a percentage of the base size; the optional %p suffix provides a size relative to some parent container.

**Related methods:**

`setAutoSizeTextTypeUniformWithPresetSizes(int,int)` (/reference/android/widget/TextView#setAutoSizeTextTypeUniformWithPresetSizes(int[],%20int))

## android:autoSizeStepGranularity

Specify the auto-size step size if `autoSizeTextType` is set to `uniform`. The default is 1px. Overwrites `autoSizePresetSizes` if set.

May be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), and mm (millimeters).

**Related methods:**

`setAutoSizeTextTypeUniformWithConfiguration(int,int,int,int)` (/reference/android/widget/TextView#setAutoSizeTextTypeUniformWithConfiguration(int,%20int,%20int,%20int))

## android:autoSizeTextType

Specify the type of auto-size. Note that this feature is not supported by EditText, works only for TextView.

Must be one of the following constant values.

| Constant | Value | Description |
|---|---|---|
| none | 0 | No auto-sizing (default). |
| uniform | 1 | Uniform horizontal and vertical text size scaling to fit within the container. |

**Related methods:**

`setAutoSizeTextTypeWithDefaults(int)` (/reference/android/widget/TextView#setAutoSizeTextTypeWithDefaults(int))

## android:autoText

If set, specifies that this TextView has a textual input method and automatically corrects some common spelling errors. The default is "false".

May be a boolean value, such as "`true`" or "`false`".

**Related methods:**

`setKeyListener(KeyListener)` (/reference/android/widget/TextView#setKeyListener(android.text.method.KeyListener))

## android:breakStrategy

Break strategy (control over paragraph layout).

Must be one of the following constant values.

| Constant | Value | Description |
|---|---|---|
| balanced | 2 | Line breaking strategy balances line lengths. |
| high_quality | 1 | Line breaking uses high-quality strategy, including hyphenation. |
| simple | 0 | Line breaking uses simple strategy. |

**Related methods:**

`setBreakStrategy(int)` (/reference/android/widget/TextView#setBreakStrategy(int))

## android:bufferType

Determines the minimum type that getText() will return. The default is "normal". Note that EditText and LogTextBox always return Editable, even if you specify something less powerful here.

Must be one of the following constant values.

| Constant | Value | Description |
|---|---|---|
| editable | 2 | Can only return Spannable and Editable. |
| normal | 0 | Can return any CharSequence, possibly a Spanned one if the source text was Spanned. |
| spannable | 1 | Can only return Spannable. |

**Related methods:**

`setText(int,TextView.BufferType)` (/reference/android/widget/TextView#setText(int,%20android.widget.TextView.BufferType))

## android:capitalize

If set, specifies that this TextView has a textual input method and should automatically capitalize what the user types. The default is "none".

Must be one of the following constant values.

| Constant | Value | Description |
|---|---|---|
| characters | 3 | Capitalize every character. |
| none | 0 | Don't automatically capitalize anything. |
| sentences | 1 | Capitalize the first word of each sentence. |
| words | 2 | Capitalize the first letter of every word. |

**Related methods:**

`setKeyListener(KeyListener)` (/reference/android/widget/TextView#setKeyListener(android.text.method.KeyListener))

## android:cursorVisible

Makes the cursor visible (the default) or invisible.

May be a boolean value, such as "`true`" or "`false`".

**Related methods:**

`setCursorVisible(boolean)` (/reference/android/widget/TextView#setCursorVisible(boolean))

## android:digits

If set, specifies that this TextView has a numeric input method and that these specific characters are the ones that it will accept. If this is set, numeric is implied to be true. The default is false.

May be a string value, using '\\;' to escape characters such as '\\n' or '\\uxxxx' for a unicode character;

**Related methods:**

`setKeyListener(KeyListener)` (/reference/android/widget/TextView#setKeyListener(android.text.method.KeyListener))

## android:drawableBottom

The drawable to be drawn below the text.

May be a reference to another resource, in the form "`@[+][`*`package`*`:]`*`type`*`/`*`name`*" or a theme attribute in the form "`?[`*`package`*`:]`*`type`*`/`*`name`*".

May be a color value, in the form of "`#`*`rgb`*", "`#`*`argb`*", "`#`*`rrggbb`*", or "`#`*`aarrggbb`*".

**Related methods:**

`setCompoundDrawablesWithIntrinsicBounds(int,int,int,int)` (/reference/android/widget/TextView#setCompoundDrawablesWithIntrinsicBounds(int,%20int,%20int,%20int))

## android:drawableEnd

The drawable to be drawn to the end of the text.

May be a reference to another resource, in the form "`@[+][`*`package`*`:]`*`type`*`/`*`name`*" or a theme attribute in the form "`?[`*`package`*`:]`*`type`*`/`*`name`*".

May be a color value, in the form of "*`#rgb`*", "*`#argb`*", "*`#rrggbb`*", or "*`#aarrggbb`*".

**Related methods:**

`setCompoundDrawablesRelativeWithIntrinsicBounds(int,int,int,int)`
(/reference/android/widget/TextView#setCompoundDrawablesRelativeWithIntrinsicBounds(int,%20int,%20int,%20int))

## android:drawableLeft

The drawable to be drawn to the left of the text.

May be a reference to another resource, in the form "`@[+][`*`package`*`:]`*`type`*`/`*`name`*" or a theme attribute in the form "`?[`*`package`*`:]`*`type`*`/`*`name`*".

May be a color value, in the form of "*`#rgb`*", "*`#argb`*", "*`#rrggbb`*", or "*`#aarrggbb`*".

**Related methods:**

`setCompoundDrawablesWithIntrinsicBounds(int,int,int,int)` (/reference/android/widget/TextView#setCompoundDrawablesWithIntrinsicBounds(int,%20int,%20int,%20int))

## android:drawablePadding

The padding between the drawables and the text.

May be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), and mm (millimeters).

**Related methods:**

`setCompoundDrawablePadding(int)` (/reference/android/widget/TextView#setCompoundDrawablePadding(int))

## android:drawableRight

The drawable to be drawn to the right of the text.

May be a reference to another resource, in the form "`@[+][`*`package`*`:]`*`type`*`/`*`name`*" or a theme attribute in the form "`?[`*`package`*`:]`*`type`*`/`*`name`*".

May be a color value, in the form of "*`#rgb`*", "*`#argb`*", "*`#rrggbb`*", or "*`#aarrggbb`*".

**Related methods:**

`setCompoundDrawablesWithIntrinsicBounds(int,int,int,int)` (/reference/android/widget/TextView#setCompoundDrawablesWithIntrinsicBounds(int,%20int,%20int,%20int))

## android:drawableStart

The drawable to be drawn to the start of the text.

May be a reference to another resource, in the form "`@[+][package:]type/name`" or a theme attribute in the form "`?[package:]type/name`".

May be a color value, in the form of "`#rgb`", "`#argb`", "`#rrggbb`", or "`#aarrggbb`".

**Related methods:**

`setCompoundDrawablesRelativeWithIntrinsicBounds(int,int,int,int)`
(/reference/android/widget/TextView#setCompoundDrawablesRelativeWithIntrinsicBounds(int,%20int,%20int,%20int))

## android:drawableTint

Tint to apply to the compound (left, top, etc.) drawables.

May be a color value, in the form of "`#rgb`", "`#argb`", "`#rrggbb`", or "`#aarrggbb`".

**Related methods:**

`setCompoundDrawableTintList(ColorStateList)` (/reference/android/widget/TextView#setCompoundDrawableTintList(android.content.res.ColorStateList))

## android:drawableTintMode

Blending mode used to apply the compound (left, top, etc.) drawables tint.

Must be one of the following constant values.

| Constant | Value | Description |
|---|---|---|
| add | 10 | Combines the tint and drawable color and alpha channels, clamping the result to valid color values. Saturate(S + D) |
| multiply | e | Multiplies the color and alpha channels of the drawable with those of the tint. [Sa * Da, Sc * Dc] |
| screen | f | [Sa + Da - Sa * Da, Sc + Dc - Sc * Dc] |
| src_atop | 9 | The tint is drawn above the drawable, but with the drawable's alpha channel masking the result. [Da, Sc * Da + (1 - Sa) * Dc] |
| src_in | 5 | The tint is masked by the alpha channel of the drawable. The drawable's color channels are thrown out. [Sa * Da, Sc * Da] |
| src_over | 3 | The tint is drawn on top of the drawable. [Sa + (1 - Sa)*Da, Rc = Sc + (1 - Sa)*Dc] |

**Related methods:**

`setCompoundDrawableTintMode(PorterDuff.Mode)` (/reference/android/widget/TextView#setCompoundDrawableTintMode(android.graphics.PorterDuff.Mode))

## android:drawableTop

The drawable to be drawn above the text.

May be a reference to another resource, in the form "`@[+][package:]type/name`" or a theme attribute in the form "`?[package:]type/name`".

May be a color value, in the form of "`#rgb`", "`#argb`", "`#rrggbb`", or "`#aarrggbb`".

**Related methods:**

`setCompoundDrawablesWithIntrinsicBounds(int,int,int,int)` (/reference/android/widget/TextView#setCompoundDrawablesWithIntrinsicBounds(int,%20int,%20int,%20int))

## android:editable

If set, specifies that this TextView has an input method. It will be a textual one unless it has otherwise been specified. For TextView, this is false by default. For EditText, it is true by default.

May be a boolean value, such as "`true`" or "`false`".

## android:editorExtras

Reference to an `<input-extras>` (/reference/android/R.styleable#InputExtras) XML resource containing additional data to supply to an input method, which is private to the implementation of the input method. This simply fills in the `EditorInfo.extras` (/reference/android/view/inputmethod/EditorInfo#extras) field when the input method is connected.

May be a reference to another resource, in the form "`@[+][package:]type/name`" or a theme attribute in the form "`?[package:]type/name`".

**Related methods:**

`setInputExtras(int)` (/reference/android/widget/TextView#setInputExtras(int))

## android:elegantTextHeight

Elegant text height, especially for less compacted complex script text.

May be a boolean value, such as "`true`" or "`false`".

**Related methods:**

`setElegantTextHeight(boolean)` (/reference/android/widget/TextView#setElegantTextHeight(boolean))

## android:ellipsize

If set, causes words that are longer than the view is wide to be ellipsized instead of broken in the middle. You will often also want to set scrollHorizontally or singleLine as well so that the text as a whole is also constrained to a single line instead of still allowed to be broken onto multiple lines.

Must be one of the following constant values.

| Constant | Value | Description |
|---|---|---|
| end | 3 | |
| marquee | 4 | |

| | |
|---|---|
| middle | 2 |
| none | 0 |
| start | 1 |

**Related methods:**

`setEllipsize(TextUtils.TruncateAt)` (/reference/android/widget/TextView#setEllipsize(android.text.TextUtils.TruncateAt))

## android:ems

Makes the TextView be exactly this many ems wide.

May be an integer value, such as "`100`".

**Related methods:**

`setEms(int)` (/reference/android/widget/TextView#setEms(int))

## android:enabled

Specifies whether the widget is enabled. The interpretation of the enabled state varies by subclass. For example, a non-enabled EditText prevents the user from editing the contained text, and a non-enabled Button prevents the user from tapping the button. The appearance of enabled and non-enabled widgets may differ, if the drawables referenced from evaluating state_enabled differ.

May be a boolean value, such as "`true`" or "`false`".

## android:fallbackLineSpacing

Whether to respect the ascent and descent of the fallback fonts that are used in displaying the text. When true, fallback fonts that end up getting used can increase the ascent and descent of the lines that they are used on.

May be a boolean value, such as "`true`" or "`false`".

**Related methods:**

`setFallbackLineSpacing(boolean)` (/reference/android/widget/TextView#setFallbackLineSpacing(boolean))

## android:firstBaselineToTopHeight

Distance from the top of the TextView to the first text baseline. If set, this overrides the value set for paddingTop.

May be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), and mm (millimeters).

**Related methods:**

`setFirstBaselineToTopHeight(int)` (/reference/android/widget/TextView#setFirstBaselineToTopHeight(int))

## android:fontFamily

Font family (named by string or as a font resource reference) for the text.

May be a string value, using '\\;' to escape characters such as '\\n' or '\\uxxxx' for a unicode character;

**Related methods:**

`setTypeface(Typeface)` (/reference/android/widget/TextView#setTypeface(android.graphics.Typeface))

## android:fontFeatureSettings

Font feature settings.

May be a string value, using '\\;' to escape characters such as '\\n' or '\\uxxxx' for a unicode character;

**Related methods:**

`setFontFeatureSettings(String)` (/reference/android/widget/TextView#setFontFeatureSettings(java.lang.String))

## android:fontVariationSettings

Font variation settings.

May be a string value, using '\\;' to escape characters such as '\\n' or '\\uxxxx' for a unicode character;

**Related methods:**

`setFontVariationSettings(String)` (/reference/android/widget/TextView#setFontVariationSettings(java.lang.String))

## android:freezesText

If set, the text view will include its current complete text inside of its frozen icicle in addition to meta-data such as the current cursor position. By default this is disabled; it can be useful when the contents of a text view is not stored in a persistent place such as a content provider. For `EditText` (/reference/android/widget/EditText) it is always enabled, regardless of the value of the attribute.

May be a boolean value, such as "`true`" or "`false`".

**Related methods:**

`setFreezesText(boolean)` (/reference/android/widget/TextView#setFreezesText(boolean))

## android:gravity

Specifies how to align the text by the view's x- and/or y-axis when the text is smaller than the view.

Must be one or more (separated by '|') of the following constant values.

| Constant | Value | Description |
| --- | --- | --- |
| bottom | 50 | Push object to the bottom of its container, not changing its size. |

| | | |
|---|---|---|
| center | 11 | Place the object in the center of its container in both the vertical and horizontal axis, not changing its size. |
| center_horizontal | 1 | Place object in the horizontal center of its container, not changing its size. |
| center_vertical | 10 | Place object in the vertical center of its container, not changing its size. |
| clip_horizontal | 8 | Additional option that can be set to have the left and/or right edges of the child clipped to its container's bounds. The clip will be based on the horizontal gravity: a left gravity will clip the right edge, a right gravity will clip the left edge, and neither will clip both edges. |
| clip_vertical | 80 | Additional option that can be set to have the top and/or bottom edges of the child clipped to its container's bounds. The clip will be based on the vertical gravity: a top gravity will clip the bottom edge, a bottom gravity will clip the top edge, and neither will clip both edges. |
| end | 800005 | Push object to the end of its container, not changing its size. |
| fill | 77 | Grow the horizontal and vertical size of the object if needed so it completely fills its container. |
| fill_horizontal | 7 | Grow the horizontal size of the object if needed so it completely fills its container. |
| fill_vertical | 70 | Grow the vertical size of the object if needed so it completely fills its container. |
| left | 3 | Push object to the left of its container, not changing its size. |
| right | 5 | Push object to the right of its container, not changing its size. |
| start | 800003 | Push object to the beginning of its container, not changing its size. |
| top | 30 | Push object to the top of its container, not changing its size. |

### Related methods:

`setGravity(int)` (/reference/android/widget/TextView#setGravity(int))

## android:height

Makes the TextView be exactly this tall. You could get the same effect by specifying this number in the layout parameters.

May be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), and mm (millimeters).

**Related methods:**

`setHeight(int)` (/reference/android/widget/TextView#setHeight(int))

## android:hint

Hint text to display when the text is empty.

May be a string value, using '\\;' to escape characters such as '\\n' or '\\uxxxx' for a unicode character;

**Related methods:**

`setHint(int)` (/reference/android/widget/TextView#setHint(int))

## android:hyphenationFrequency

Frequency of automatic hyphenation.

Must be one of the following constant values.

| Constant | Value | Description |
|---|---|---|
| full | 2 | Standard amount of hyphenation, useful for running text and for screens with limited space for text. |
| none | 0 | No hyphenation. |
| normal | 1 | Less frequent hyphenation, useful for informal use cases, such as chat messages. |

**Related methods:**

`setHyphenationFrequency(int)` (/reference/android/widget/TextView#setHyphenationFrequency(int))

## android:imeActionId

Supply a value for `EditorInfo.actionId` (/reference/android/view/inputmethod/EditorInfo#actionId) used when an input method is connected to the text view.

May be an integer value, such as "`100`".

**Related methods:**

`setImeActionLabel(CharSequence,int)` (/reference/android/widget/TextView#setImeActionLabel(java.lang.CharSequence,%20int))

## android:imeActionLabel

Supply a value for `EditorInfo.actionLabel` (/reference/android/view/inputmethod/EditorInfo#actionLabel) used when an input method is connected to the text view.

May be a string value, using '\\;' to escape characters such as '\\n' or '\\uxxxx' for a unicode character;

**Related methods:**

`setImeActionLabel(CharSequence,int)` (/reference/android/widget/TextView#setImeActionLabel(java.lang.CharSequence,%20int))

## android:imeOptions

Additional features you can enable in an IME associated with an editor to improve the integration with your application. The constants here correspond to those defined by `EditorInfo.imeOptions` (/reference/android/view/inputmethod/EditorInfo#imeOptions).

Must be one or more (separated by '|') of the following constant values.

| Constant | Value | Description |
|---|---|---|
| actionDone | 6 | The action key performs a "done" operation, closing the soft input method. Corresponds to `EditorInfo.IME_ACTION_DONE` (/reference/android/view/inputmethod/EditorInfo#IME_ACTION_DONE). |
| actionGo | 2 | The action key performs a "go" operation to take the user to the target of the text they typed. Typically used, for example, when entering a URL. Corresponds to `EditorInfo.IME_ACTION_GO` (/reference/android/view/inputmethod/EditorInfo#IME_ACTION_GO). |
| actionNext | 5 | The action key performs a "next" operation, taking the user to the next field that will accept text. Corresponds to `EditorInfo.IME_ACTION_NEXT` (/reference/android/view/inputmethod/EditorInfo#IME_ACTION_NEXT). |
| actionNone | 1 | This editor has no action associated with it. Corresponds to `EditorInfo.IME_ACTION_NONE` (/reference/android/view/inputmethod/EditorInfo#IME_ACTION_NONE). |

| | | |
|---|---|---|
| actionPrevious | 7 | The action key performs a "previous" operation, taking the user to the previous field that will accept text. Corresponds to **EditorInfo.IME_ACTION_PREVIOUS** (/reference/android/view/inputmethod/EditorInfo#IME_ACTION_PREVIOUS). |
| actionSearch | 3 | The action key performs a "search" operation, taking the user to the results of searching for the text the have typed (in whatever context is appropriate). Corresponds to **EditorInfo.IME_ACTION_SEARCH** (/reference/android/view/inputmethod/EditorInfo#IME_ACTION_SEARCH). |
| actionSend | 4 | The action key performs a "send" operation, delivering the text to its target. This is typically used when composing a message. Corresponds to **EditorInfo.IME_ACTION_SEND** (/reference/android/view/inputmethod/EditorInfo#IME_ACTION_SEND). |
| actionUnspecified | 0 | There is no specific action associated with this editor, let the editor come up with its own if it can. Corresponds to **EditorInfo.IME_NULL** (/reference/android/view/inputmethod/EditorInfo#IME_NULL). |
| flagForceAscii | 80000000 | Used to request that the IME should be capable of inputting ASCII characters. The intention of this flag is to ensure that the user can type Roman alphabet characters in a **TextView** (/reference/android/widget/TextView) used for, typically, account ID or password input. It is expected that IMEs normally are able to input ASCII even without being told so (such IMEs already respect this flag in a sense), but there could be some cases they aren't when, for instance, only non-ASCII input languages like Arabic, Greek, Hebrew, Russian are enabled in the IME. Applications need to be aware that the flag is not a guarantee, and not all IMEs will respect it. However, it is strongly recommended for IME authors to respect this flag especially when their IME could end up with a state that has only non-ASCII input languages enabled.<br><br>Corresponds to **EditorInfo.IME_FLAG_FORCE_ASCII** (/reference/android/view/inputmethod/EditorInfo#IME_FLAG_FORCE_ASCII). |
| flagNavigateNext | 8000000 | Used to specify that there is something interesting that a forward navigation can focus on. This is like using actionNext, except allows the IME to be multiline (with an enter key) as well as provide forward navigation. Note that some IMEs may not be able to do this, especially when running on a small screen where there is little space. In that case it does not need to present a UI for this option. Like actionNext, if the user selects the IME's facility to forward navigate, this will show up in the application at **InputConnection.performEditorAction(int)** (/reference/android/view/inputmethod/InputConnection#performEditorAction(int)).<br><br>Corresponds to **EditorInfo.IME_FLAG_NAVIGATE_NEXT** (/reference/android/view/inputmethod/EditorInfo#IME_FLAG_NAVIGATE_NEXT). |
| flagNavigatePrevious | 4000000 | Like flagNavigateNext, but specifies there is something interesting that a backward navigation can focus on. If the user selects the IME's facility to backward navigate, this will show up in the application as an actionPrevious at **InputConnection.performEditorAction(int)** (/reference/android/view/inputmethod/InputConnection#performEditorAction(int)). |

| | | |
|---|---|---|
| | | Corresponds to **EditorInfo.IME_FLAG_NAVIGATE_PREVIOUS** (/reference/android/view/inputmethod/EditorInfo#IME_FLAG_NAVIGATE_PREVIOUS). |
| flagNoAccessoryAction | 20000000 | Used in conjunction with a custom action, this indicates that the action should not be available as an accessory button when the input method is full-screen. Note that by setting this flag, there can be cases where the action is simply never available to the user. Setting this generally means that you think showing text being edited is more important than the action you have supplied.<br>Corresponds to **EditorInfo.IME_FLAG_NO_ACCESSORY_ACTION** (/reference/android/view/inputmethod/EditorInfo#IME_FLAG_NO_ACCESSORY_ACTION). |
| flagNoEnterAction | 40000000 | Used in conjunction with a custom action, this indicates that the action should not be available in-line as a replacement for the "enter" key. Typically this is because the action has such a significant impact or is not recoverable enough that accidentally hitting it should be avoided, such as sending a message. Note that **TextView** (/reference/android/widget/TextView) will automatically set this flag for you on multi-line text views.<br><br>Corresponds to **EditorInfo.IME_FLAG_NO_ENTER_ACTION** (/reference/android/view/inputmethod/EditorInfo#IME_FLAG_NO_ENTER_ACTION). |
| flagNoExtractUi | 10000000 | Used to specify that the IME does not need to show its extracted text UI. For input methods that may be fullscreen, often when in landscape mode, this allows them to be smaller and let part of the application be shown behind. Though there will likely be limited access to the application available from the user, it can make the experience of a (mostly) fullscreen IME less jarring. Note that when this flag is specified the IME may *not* be set up to be able to display text, so it should only be used in situations where this is not needed.<br><br>Corresponds to **EditorInfo.IME_FLAG_NO_EXTRACT_UI** (/reference/android/view/inputmethod/EditorInfo#IME_FLAG_NO_EXTRACT_UI). |
| flagNoFullscreen | 2000000 | Used to request that the IME never go into fullscreen mode. Applications need to be aware that the flag is not a guarantee, and not all IMEs will respect it.<br>Corresponds to **EditorInfo.IME_FLAG_NO_FULLSCREEN** (/reference/android/view/inputmethod/EditorInfo#IME_FLAG_NO_FULLSCREEN). |
| flagNoPersonalizedLearning | 1000000 | Used to request that the IME should not update any personalized data such as typing history and personalized language model based on what the user typed on this text editing object. Typical use cases are:<br>• When the application is in a special mode, where user's activities are expected to be not recorded in the application's history. Some web browsers and chat applications may have this kind of modes.<br>• When storing typing history does not make much sense. Specifying this flag in typing games may help to avoid typing history from being filled up with words that the user is less likely to type in their daily life. Another example is that when the application already knows that the expected input is not a valid word (e.g. a promotion code that is not a valid word in any natural language). |

| | | |
|---|---|---|
| | | Applications need to be aware that the flag is not a guarantee, and some IMEs may not respect it. |
| normal | 0 | There are no special semantics associated with this editor. |

**Related methods:**

`setImeOptions(int)` (/reference/android/widget/TextView#setImeOptions(int))

## android:includeFontPadding

Leave enough room for ascenders and descenders instead of using the font ascent and descent strictly. (Normally true).

May be a boolean value, such as "`true`" or "`false`".

**Related methods:**

`setIncludeFontPadding(boolean)` (/reference/android/widget/TextView#setIncludeFontPadding(boolean))

## android:inputMethod

If set, specifies that this TextView should use the specified input method (specified by fully-qualified class name).

May be a string value, using '\\;' to escape characters such as '\\n' or '\\uxxxx' for a unicode character;

**Related methods:**

`setKeyListener(KeyListener)` (/reference/android/widget/TextView#setKeyListener(android.text.method.KeyListener))

## android:inputType

The type of data being placed in a text field, used to help an input method decide how to let the user enter text. The constants here correspond to those defined by `InputType` (/reference/android/text/InputType). Generally you can select a single value, though some can be combined together as indicated. Setting this attribute to anything besides ***none*** also implies that the text is editable.

Must be one or more (separated by '|') of the following constant values.

| Constant | Value | Description |
|---|---|---|
| date | 14 | For entering a date. Corresponds to `InputType.TYPE_CLASS_DATETIME` (/reference/android/text/InputType#TYPE_CLASS_DATETIME) \| `InputType.TYPE_DATETIME_VARIATION_DATE` (/reference/android/text/InputType#TYPE_DATETIME_VARIATION_DATE). |
| datetime | 4 | For entering a date and time. Corresponds to `InputType.TYPE_CLASS_DATETIME` (/reference/android/text/InputType#TYPE_CLASS_DATETIME) \| `InputType.TYPE_DATETIME_VARIATION_NORMAL` (/reference/android/text/InputType#TYPE_DATETIME_VARIATION_NORMAL). |
| none | 0 | There is no content type. The text is not editable. |
| number | 2 | A numeric only field. Corresponds to `InputType.TYPE_CLASS_NUMBER` (/reference/android/text/InputType#TYPE_CLASS_NUMBER) \| `InputType.TYPE_NUMBER_VARIATION_NORMAL` (/reference/android/text/InputType#TYPE_NUMBER_VARIATION_NORMAL). |
| numberDecimal | 2002 | Can be combined with ***number*** and its other options to allow a decimal (fractional) number. Corresponds to `InputType.TYPE_CLASS_NUMBER` (/reference/android/text/InputType#TYPE_CLASS_NUMBER) \| `InputType.TYPE_NUMBER_FLAG_DECIMAL` (/reference/android/text/InputType#TYPE_NUMBER_FLAG_DECIMAL). |
| numberPassword | 12 | A numeric password field. Corresponds to `InputType.TYPE_CLASS_NUMBER` (/reference/android/text/InputType#TYPE_CLASS_NUMBER) \| `InputType.` |

| | | |
|---|---|---|
| | | **TYPE_NUMBER_VARIATION_PASSWORD** (/reference/android/text/InputType#TYPE_NUMBER_VARIATION_PASSWORD). |
| numberSigned | 1002 | Can be combined with ***number*** and its other options to allow a signed number. Corresponds to **InputType.TYPE_CLASS_NUMBER** (/reference/android/text/InputType#TYPE_CLASS_NUMBER) \| **InputType.TYPE_NUMBER_FLAG_SIGNED** (/reference/android/text/InputType#TYPE_NUMBER_FLAG_SIGNED). |
| phone | 3 | For entering a phone number. Corresponds to **InputType.TYPE_CLASS_PHONE** (/reference/android/text/InputType#TYPE_CLASS_PHONE). |
| text | 1 | Just plain old text. Corresponds to **InputType.TYPE_CLASS_TEXT** (/reference/android/text/InputType#TYPE_CLASS_TEXT) \| **InputType.TYPE_TEXT_VARIATION_NORMAL** (/reference/android/text/InputType#TYPE_TEXT_VARIATION_NORMAL). |
| textAutoComplete | 10001 | Can be combined with ***text*** and its variations to specify that this field will be doing its own auto-completion and talking with the input method appropriately. Corresponds to **InputType.TYPE_TEXT_FLAG_AUTO_COMPLETE** (/reference/android/text/InputType#TYPE_TEXT_FLAG_AUTO_COMPLETE). |
| textAutoCorrect | 8001 | Can be combined with ***text*** and its variations to request auto-correction of text being input. Corresponds to **InputType.TYPE_TEXT_FLAG_AUTO_CORRECT** (/reference/android/text/InputType#TYPE_TEXT_FLAG_AUTO_CORRECT). |
| textCapCharacters | 1001 | Can be combined with ***text*** and its variations to request capitalization of all characters. Corresponds to **InputType.TYPE_TEXT_FLAG_CAP_CHARACTERS** (/reference/android/text/InputType#TYPE_TEXT_FLAG_CAP_CHARACTERS). |
| textCapSentences | 4001 | Can be combined with ***text*** and its variations to request capitalization of the first character of every sentence. Corresponds to **InputType.TYPE_TEXT_FLAG_CAP_SENTENCES** (/reference/android/text/InputType#TYPE_TEXT_FLAG_CAP_SENTENCES). |
| textCapWords | 2001 | Can be combined with ***text*** and its variations to request capitalization of the first character of every word. Corresponds to **InputType.TYPE_TEXT_FLAG_CAP_WORDS** (/reference/android/text/InputType#TYPE_TEXT_FLAG_CAP_WORDS). |
| textEmailAddress | 21 | Text that will be used as an e-mail address. Corresponds to **InputType.TYPE_CLASS_TEXT** (/reference/android/text/InputType#TYPE_CLASS_TEXT) \| **InputType.TYPE_TEXT_VARIATION_EMAIL_ADDRESS** (/reference/android/text/InputType#TYPE_TEXT_VARIATION_EMAIL_ADDRESS). |
| textEmailSubject | 31 | Text that is being supplied as the subject of an e-mail. Corresponds to **InputType.TYPE_CLASS_TEXT** (/reference/android/text/InputType#TYPE_CLASS_TEXT) \| **InputType.TYPE_TEXT_VARIATION_EMAIL_SUBJECT** (/reference/android/text/InputType#TYPE_TEXT_VARIATION_EMAIL_SUBJECT). |
| textFilter | b1 | Text that is filtering some other data. Corresponds to **InputType.TYPE_CLASS_TEXT** (/reference/android/text/InputType#TYPE_CLASS_TEXT) \| **InputType.** |

| | | |
|---|---|---|
| | | **TYPE_TEXT_VARIATION_FILTER** (/reference/android/text/InputType#TYPE_TEXT_VARIATION_FILTER). |
| textImeMultiLine | 40001 | Can be combined with ***text*** and its variations to indicate that though the regular text view should not be multiple lines, the IME should provide multiple lines if it can. Corresponds to **InputType.TYPE_TEXT_FLAG_IME_MULTI_LINE** (/reference/android/text/InputType#TYPE_TEXT_FLAG_IME_MULTI_LINE). |
| textLongMessage | 51 | Text that is the content of a long message. Corresponds to **InputType.TYPE_CLASS_TEXT** (/reference/android/text/InputType#TYPE_CLASS_TEXT) \| **InputType.TYPE_TEXT_VARIATION_LONG_MESSAGE** (/reference/android/text/InputType#TYPE_TEXT_VARIATION_LONG_MESSAGE). |
| textMultiLine | 20001 | Can be combined with ***text*** and its variations to allow multiple lines of text in the field. If this flag is not set, the text field will be constrained to a single line. Corresponds to **InputType.TYPE_TEXT_FLAG_MULTI_LINE** (/reference/android/text/InputType#TYPE_TEXT_FLAG_MULTI_LINE). |
| textNoSuggestions | 80001 | Can be combined with ***text*** and its variations to indicate that the IME should not show any dictionary-based word suggestions. Corresponds to **InputType.TYPE_TEXT_FLAG_NO_SUGGESTIONS** (/reference/android/text/InputType#TYPE_TEXT_FLAG_NO_SUGGESTIONS). |
| textPassword | 81 | Text that is a password. Corresponds to **InputType.TYPE_CLASS_TEXT** (/reference/android/text/InputType#TYPE_CLASS_TEXT) \| **InputType.TYPE_TEXT_VARIATION_PASSWORD** (/reference/android/text/InputType#TYPE_TEXT_VARIATION_PASSWORD). |
| textPersonName | 61 | Text that is the name of a person. Corresponds to **InputType.TYPE_CLASS_TEXT** (/reference/android/text/InputType#TYPE_CLASS_TEXT) \| **InputType.TYPE_TEXT_VARIATION_PERSON_NAME** (/reference/android/text/InputType#TYPE_TEXT_VARIATION_PERSON_NAME). |
| textPhonetic | c1 | Text that is for phonetic pronunciation, such as a phonetic name field in a contact entry. Corresponds to **InputType.TYPE_CLASS_TEXT** (/reference/android/text/InputType#TYPE_CLASS_TEXT) \| **InputType.TYPE_TEXT_VARIATION_PHONETIC** (/reference/android/text/InputType#TYPE_TEXT_VARIATION_PHONETIC). |
| textPostalAddress | 71 | Text that is being supplied as a postal mailing address. Corresponds to **InputType.TYPE_CLASS_TEXT** (/reference/android/text/InputType#TYPE_CLASS_TEXT) \| **InputType.TYPE_TEXT_VARIATION_POSTAL_ADDRESS** (/reference/android/text/InputType#TYPE_TEXT_VARIATION_POSTAL_ADDRESS). |
| textShortMessage | 41 | Text that is the content of a short message. Corresponds to **InputType.TYPE_CLASS_TEXT** (/reference/android/text/InputType#TYPE_CLASS_TEXT) \| **InputType.TYPE_TEXT_VARIATION_SHORT_MESSAGE** (/reference/android/text/InputType#TYPE_TEXT_VARIATION_SHORT_MESSAGE). |
| textUri | 11 | Text that will be used as a URI. Corresponds to **InputType.TYPE_CLASS_TEXT** (/reference/android/text/InputType#TYPE_CLASS_TEXT) \| **InputType.TYPE_TEXT_VARIATION_URI** (/reference/android/text/InputType#TYPE_TEXT_VARIATION_URI). |

| | | |
|---|---|---|
| textVisiblePassword | 91 | Text that is a password that should be visible. Corresponds to **InputType.TYPE_CLASS_TEXT** (/reference/android/text/InputType#TYPE_CLASS_TEXT) \| **InputType.TYPE_TEXT_VARIATION_VISIBLE_PASSWORD** (/reference/android/text/InputType#TYPE_TEXT_VARIATION_VISIBLE_PASSWORD). |
| textWebEditText | a1 | Text that is being supplied as text in a web form. Corresponds to **InputType.TYPE_CLASS_TEXT** (/reference/android/text/InputType#TYPE_CLASS_TEXT) \| **InputType.TYPE_TEXT_VARIATION_WEB_EDIT_TEXT** (/reference/android/text/InputType#TYPE_TEXT_VARIATION_WEB_EDIT_TEXT). |
| textWebEmailAddress | d1 | Text that will be used as an e-mail address on a web form. Corresponds to **InputType.TYPE_CLASS_TEXT** (/reference/android/text/InputType#TYPE_CLASS_TEXT) \| **InputType.TYPE_TEXT_VARIATION_WEB_EMAIL_ADDRESS** (/reference/android/text/InputType#TYPE_TEXT_VARIATION_WEB_EMAIL_ADDRESS). |
| textWebPassword | e1 | Text that will be used as a password on a web form. Corresponds to **InputType.TYPE_CLASS_TEXT** (/reference/android/text/InputType#TYPE_CLASS_TEXT) \| **InputType.TYPE_TEXT_VARIATION_WEB_PASSWORD** (/reference/android/text/InputType#TYPE_TEXT_VARIATION_WEB_PASSWORD). |
| time | 24 | For entering a time. Corresponds to **InputType.TYPE_CLASS_DATETIME** (/reference/android/text/InputType#TYPE_CLASS_DATETIME) \| **InputType.TYPE_DATETIME_VARIATION_TIME** (/reference/android/text/InputType#TYPE_DATETIME_VARIATION_TIME). |

**Related methods:**

**setRawInputType(int)** (/reference/android/widget/TextView#setRawInputType(int))

## android:justificationMode

Mode for justification.

Must be one of the following constant values.

| Constant | Value | Description |
|---|---|---|
| inter_word | 1 | Justification by stretching word spacing. |

| | | |
|---|---|---|
| none | 0 | No justification. |

## android:lastBaselineToBottomHeight

Distance from the bottom of the TextView to the last text baseline. If set, this overrides the value set for paddingBottom.

May be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), and mm (millimeters).

**Related methods:**

`setLastBaselineToBottomHeight(int)` (/reference/android/widget/TextView#setLastBaselineToBottomHeight(int))

## android:letterSpacing

Text letter-spacing.

May be a floating point value, such as "`1.2`".

**Related methods:**

`setLetterSpacing(float)` (/reference/android/widget/TextView#setLetterSpacing(float))

## android:lineHeight

Explicit height between lines of text. If set, this will override the values set for lineSpacingExtra and lineSpacingMultiplier.

May be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), and mm (millimeters).

**Related methods:**

`setLineHeight(int)` (/reference/android/widget/TextView#setLineHeight(int))

## android:lineSpacingExtra

Extra spacing between lines of text. The value will not be applied for the last line of text.

May be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), and mm (millimeters).

**Related methods:**

`setLineSpacing(float,float)` (/reference/android/widget/TextView#setLineSpacing(float,%20float))

## android:lineSpacingMultiplier

Extra spacing between lines of text, as a multiplier. The value will not be applied for the last line of text.

May be a floating point value, such as "`1.2`".

**Related methods:**

`setLineSpacing(float,float)` (/reference/android/widget/TextView#setLineSpacing(float,%20float))

## android:lines

Makes the TextView be exactly this many lines tall.

May be an integer value, such as "`100`".

**Related methods:**

`setLines(int)` (/reference/android/widget/TextView#setLines(int))

## android:linksClickable

If set to false, keeps the movement method from being set to the link movement method even if autoLink causes links to be found.

May be a boolean value, such as "`true`" or "`false`".

**Related methods:**

`setLinksClickable(boolean)` (/reference/android/widget/TextView#setLinksClickable(boolean))

## android:marqueeRepeatLimit

The number of times to repeat the marquee animation. Only applied if the TextView has marquee enabled.

May be an integer value, such as "`100`".

Must be one of the following constant values.

| Constant | Value | Description |
|---|---|---|
| marquee_forever | ffffffff | Indicates that marquee should repeat indefinitely. |

**Related methods:**

`setMarqueeRepeatLimit(int)` (/reference/android/widget/TextView#setMarqueeRepeatLimit(int))

## android:maxEms

Makes the TextView be at most this many ems wide.

May be an integer value, such as "`100`".

**Related methods:**

`setMaxEms(int)` (/reference/android/widget/TextView#setMaxEms(int))

## android:maxHeight

Makes the TextView be at most this many pixels tall.

May be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), and mm (millimeters).

**Related methods:**

`setMaxHeight(int)` (/reference/android/widget/TextView#setMaxHeight(int))

## android:maxLength

Set an input filter to constrain the text length to the specified number.

May be an integer value, such as "`100`".

**Related methods:**

`setFilters(InputFilter)` (/reference/android/widget/TextView#setFilters(android.text.InputFilter[]))

## android:maxLines

Makes the TextView be at most this many lines tall. When used on an editable text, the `inputType` attribute's value must be combined with the `textMultiLine` flag for the maxLines attribute to apply.

May be an integer value, such as "`100`".

**Related methods:**

`setMaxLines(int)` (/reference/android/widget/TextView#setMaxLines(int))

## android:maxWidth

Makes the TextView be at most this many pixels wide.

May be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), and mm (millimeters).

**Related methods:**

`setMaxWidth(int)` (/reference/android/widget/TextView#setMaxWidth(int))

## android:minEms

Makes the TextView be at least this many ems wide.

May be an integer value, such as "`100`".

**Related methods:**

`setMinEms(int)` (/reference/android/widget/TextView#setMinEms(int))

## android:minHeight

Makes the TextView be at least this many pixels tall.

May be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), and mm (millimeters).

**Related methods:**

`setMinHeight(int)` (/reference/android/widget/TextView#setMinHeight(int))

## android:minLines

Makes the TextView be at least this many lines tall. When used on an editable text, the `inputType` attribute's value must be combined with the `textMultiLine` flag for the minLines attribute to apply.

May be an integer value, such as "`100`".

**Related methods:**

`setMinLines(int)` (/reference/android/widget/TextView#setMinLines(int))

## android:minWidth

Makes the TextView be at least this many pixels wide.

May be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), and mm (millimeters).

**Related methods:**

`setMinWidth(int)` (/reference/android/widget/TextView#setMinWidth(int))

## android:numeric

If set, specifies that this TextView has a numeric input method. The default is false.

Must be one or more (separated by '|') of the following constant values.

| Constant | Value | Description |
|---|---|---|
| decimal | 5 | Input is numeric, with decimals allowed. |
| integer | 1 | Input is numeric. |
| signed | 3 | Input is numeric, with sign allowed. |

**Related methods:**

`setKeyListener(KeyListener)` (/reference/android/widget/TextView#setKeyListener(android.text.method.KeyListener))

## android:password

Whether the characters of the field are displayed as password dots instead of themselves.

May be a boolean value, such as "`true`" or "`false`".

**Related methods:**

`setTransformationMethod(TransformationMethod)` (/reference/android/widget/TextView#setTransformationMethod(android.text.method.TransformationMethod))

## android:phoneNumber

If set, specifies that this TextView has a phone number input method. The default is false.

May be a boolean value, such as "`true`" or "`false`".

**Related methods:**

`setKeyListener(KeyListener)` (/reference/android/widget/TextView#setKeyListener(android.text.method.KeyListener))

## android:privateImeOptions

An addition content type description to supply to the input method attached to the text view, which is private to the implementation of the input method. This simply fills in the `EditorInfo.privateImeOptions` (/reference/android/view/inputmethod/EditorInfo#privateImeOptions) field when the input method is connected.

May be a string value, using '\\;' to escape characters such as '\\n' or '\\uxxxx' for a unicode character;

**Related methods:**

`setPrivateImeOptions(String)` (/reference/android/widget/TextView#setPrivateImeOptions(java.lang.String))

## android:scrollHorizontally

Whether the text is allowed to be wider than the view (and therefore can be scrolled horizontally).

May be a boolean value, such as "`true`" or "`false`".

**Related methods:**

`setHorizontallyScrolling(boolean)` (/reference/android/widget/TextView#setHorizontallyScrolling(boolean))

## android:selectAllOnFocus

If the text is selectable, select it all when the view takes focus.

May be a boolean value, such as "`true`" or "`false`".

**Related methods:**

`setSelectAllOnFocus(boolean)` (/reference/android/widget/TextView#setSelectAllOnFocus(boolean))

## android:shadowColor

Place a blurred shadow of text underneath the text, drawn with the specified color. The text shadow produced does not interact with properties on View that are responsible for real time shadows, `elevation` (/reference/android/R.styleable#View_elevation) and `translationZ` (/reference/android/R.styleable#View_translationZ).

May be a color value, in the form of "*`#rgb`*", "*`#argb`*", "*`#rrggbb`*", or "*`#aarrggbb`*".

**Related methods:**

`setShadowLayer(float,float,float,int)` (/reference/android/widget/TextView#setShadowLayer(float,%20float,%20float,%20int))

## android:shadowDx

Horizontal offset of the text shadow.

May be a floating point value, such as "`1.2`".

**Related methods:**

`setShadowLayer(float,float,float,int)` (/reference/android/widget/TextView#setShadowLayer(float,%20float,%20float,%20int))

## android:shadowDy

Vertical offset of the text shadow.

May be a floating point value, such as "`1.2`".

**Related methods:**

`setShadowLayer(float,float,float,int)` (/reference/android/widget/TextView#setShadowLayer(float,%20float,%20float,%20int))

## android:shadowRadius

Blur radius of the text shadow.

May be a floating point value, such as "`1.2`".

**Related methods:**

`setShadowLayer(float,float,float,int)` (/reference/android/widget/TextView#setShadowLayer(float,%20float,%20float,%20int))

## android:singleLine

Constrains the text to a single horizontally scrolling line instead of letting it wrap onto multiple lines, and advances focus instead of inserting a newline when you press the enter key. The default value is false (multi-line wrapped text mode) for non-editable text, but if you specify any value for inputType, the default is true (single-line input field mode).

May be a boolean value, such as "`true`" or "`false`".

**Related methods:**

`setTransformationMethod(TransformationMethod)` (/reference/android/widget/TextView#setTransformationMethod(android.text.method.TransformationMethod))

## android:text

Text to display.

May be a string value, using '\\;' to escape characters such as '\\n' or '\\uxxxx' for a unicode character;

**Related methods:**

`setText(int,TextView.BufferType)` (/reference/android/widget/TextView#setText(int,%20android.widget.TextView.BufferType))

## android:textAllCaps

Present the text in ALL CAPS. This may use a small-caps form when available.

May be a boolean value, such as "`true`" or "`false`".

**Related methods:**

`setAllCaps(boolean)` (/reference/android/widget/TextView#setAllCaps(boolean))

## android:textAppearance

Base text color, typeface, size, and style.

May be a reference to another resource, in the form "`@[+][package:]type/name`" or a theme attribute in the form "`?[package:]type/name`".

**Related methods:**

`setTextAppearance(int)` (/reference/android/widget/TextView#setTextAppearance(int))

## android:textColor

Text color.

May be a reference to another resource, in the form "`@[+][package:]type/name`" or a theme attribute in the form "`?[package:]type/name`".

May be a color value, in the form of "`#rgb`", "`#argb`", "`#rrggbb`", or "`#aarrggbb`".

**Related methods:**

`setTextColor(ColorStateList)` (/reference/android/widget/TextView#setTextColor(android.content.res.ColorStateList))

## android:textColorHighlight

Color of the text selection highlight.

May be a reference to another resource, in the form "`@[+][package:]type/name`" or a theme attribute in the form "`?[package:]type/name`".

May be a color value, in the form of "`#rgb`", "`#argb`", "`#rrggbb`", or "`#aarrggbb`".

**Related methods:**

`setHighlightColor(int)` (/reference/android/widget/TextView#setHighlightColor(int))

## android:textColorHint

Color of the hint text.

May be a reference to another resource, in the form "`@[+][package:]type/name`" or a theme attribute in the form "`?[package:]type/name`".

May be a color value, in the form of "`#rgb`", "`#argb`", "`#rrggbb`", or "`#aarrggbb`".

**Related methods:**

`setHintTextColor(int)` (/reference/android/widget/TextView#setHintTextColor(int))

## android:textColorLink

Text color for links.

May be a reference to another resource, in the form "`@[+][package:]type/name`" or a theme attribute in the form "`?[package:]type/name`".

May be a color value, in the form of "*#rgb*", "*#argb*", "*#rrggbb*", or "*#aarrggbb*".

**Related methods:**

`setLinkTextColor(int)` (/reference/android/widget/TextView#setLinkTextColor(int))

## android:textCursorDrawable

Reference to a drawable that will be drawn under the insertion cursor.

May be a reference to another resource, in the form "`@[+][`*package*`:]`*type*`/`*name*" or a theme attribute in the form "`?[`*package*`:]`*type*`/`*name*".

**Related methods:**

`setTextCursorDrawable(int)` (/reference/android/widget/TextView#setTextCursorDrawable(int))

## android:textFontWeight

Weight for the font used in the TextView.

May be an integer value, such as "`100`".

## android:textIsSelectable

Indicates that the content of a non-editable text can be selected.

May be a boolean value, such as "`true`" or "`false`".

**Related methods:**

`isTextSelectable()` (/reference/android/widget/TextView#isTextSelectable())

## android:textScaleX

Sets the horizontal scaling factor for the text.

May be a floating point value, such as "`1.2`".

**Related methods:**

`setTextScaleX(float)` (/reference/android/widget/TextView#setTextScaleX(float))

## android:textSelectHandle

Reference to a drawable that will be used to display a text selection anchor for positioning the cursor within text.

May be a reference to another resource, in the form "`@[+][package:]type/name`" or a theme attribute in the form "`?[package:]type/name`".

**Related methods:**

`setTextSelectHandle(Drawable)` (/reference/android/widget/TextView#setTextSelectHandle(android.graphics.drawable.Drawable))

## android:textSelectHandleLeft

Reference to a drawable that will be used to display a text selection anchor on the left side of a selection region.

May be a reference to another resource, in the form "`@[+][package:]type/name`" or a theme attribute in the form "`?[package:]type/name`".

**Related methods:**

`setTextSelectHandleLeft(Drawable)` (/reference/android/widget/TextView#setTextSelectHandleLeft(android.graphics.drawable.Drawable))

## android:textSelectHandleRight

Reference to a drawable that will be used to display a text selection anchor on the right side of a selection region.

May be a reference to another resource, in the form "`@[+][package:]type/name`" or a theme attribute in the form "`?[package:]type/name`".

**Related methods:**

`setTextSelectHandleRight(int)` (/reference/android/widget/TextView#setTextSelectHandleRight(int))

## android:textSize

Size of the text. Recommended dimension type for text is "sp" for scaled-pixels (example: 15sp).

May be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), and mm (millimeters).

**Related methods:**

`setTextSize(float)` (/reference/android/widget/TextView#setTextSize(float))

## android:textStyle

Style (normal, bold, italic, bold|italic) for the text.

Must be one or more (separated by '|') of the following constant values.

| Constant | Value | Description |
|---|---|---|
| bold | 1 | |
| italic | 2 | |
| normal | 0 | |

**Related methods:**

`setTypeface(Typeface,int)` (/reference/android/widget/TextView#setTypeface(android.graphics.Typeface,%20int))

## android:typeface

Typeface (normal, sans, serif, monospace) for the text.

Must be one of the following constant values.

| Constant | Value | Description |
|---|---|---|
| monospace | 3 | |
| normal | 0 | |
| sans | 1 | |
| serif | 2 | |

**Related methods:**

`setTypeface(Typeface,int)` (/reference/android/widget/TextView#setTypeface(android.graphics.Typeface,%20int))

## android:width

Makes the TextView be exactly this wide. You could get the same effect by specifying this number in the layout parameters.

May be a dimension value, which is a floating point number appended with a unit such as "`14.5sp`". Available units are: px (pixels), dp (density-independent pixels), sp (scaled pixels based on preferred font size), in (inches), and mm (millimeters).

**Related methods:**

`setWidth(int)` (/reference/android/widget/TextView#setWidth(int))

## Constants

### AUTO_SIZE_TEXT_TYPE_NONE

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int AUTO_SIZE_TEXT_TYPE_NONE
```

The TextView does not auto-size text (default).

Constant Value: 0 (0x00000000)

### AUTO_SIZE_TEXT_TYPE_UNIFORM

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public static final int AUTO_SIZE_TEXT_TYPE_UNIFORM
```

The TextView scales text size both horizontally and vertically to fit within the container.

Constant Value: 1 (0x00000001)

# Public constructors

## TextView

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public TextView (Context (/reference/android/content/Context) context)
```

### Parameters

| | |
|---|---|
| **context** | **Context** |

## TextView

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public TextView (Context (/reference/android/content/Context) context,
                AttributeSet (/reference/android/util/AttributeSet) attrs)
```

### Parameters

| | |
|---|---|
| **context** | **Context** |
| **attrs** | **AttributeSet**: This value may be **null**. |

## TextView

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public TextView (Context (/reference/android/content/Context) context,
                AttributeSet (/reference/android/util/AttributeSet) attrs,
                int defStyleAttr)
```

### Parameters

| | |
|---|---|
| **context** | **Context** |
| **attrs** | **AttributeSet**: This value may be **null**. |
| **defStyleAttr** | **int** |

## TextView

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public TextView (Context (/reference/android/content/Context) context,
                AttributeSet (/reference/android/util/AttributeSet) attrs,
                int defStyleAttr,
                int defStyleRes)
```

### Parameters

| | |
|---|---|
| `context` | `Context` |
| `attrs` | `AttributeSet`: This value may be `null`. |
| `defStyleAttr` | `int` |
| `defStyleRes` | `int` |

## Public methods

### addExtraDataToAccessibilityNodeInfo

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void addExtraDataToAccessibilityNodeInfo (AccessibilityNodeInfo (/reference/android/view/accessibility/AccessibilityNodeInfo) info,
                String (/reference/java/lang/String) extraDataKey,
                Bundle (/reference/android/os/Bundle) arguments)
```

Adds extra data to an **`AccessibilityNodeInfo`** (/reference/android/view/accessibility/AccessibilityNodeInfo) based on an explicit request for the additional data.

This method only needs overloading if the node is marked as having extra data available.

### Parameters

| | |
|---|---|
| `info` | **`AccessibilityNodeInfo`**: The info to which to add the extra data. Never **`null`**. This value must never be **`null`**. |
| `extraDataKey` | **`String`**: A key specifying the type of extra data to add to the info. The extra data should be added to the **`Bundle`** (/reference/android/os/Bundle) returned by the info's **`AccessibilityNodeInfo#getExtras`** (/reference/android/view/accessibility/AccessibilityNodeInfo#getExtras()) method. Never **`null`**. This value must never be **`null`**. |
| `arguments` | **`Bundle`**: A **`Bundle`** (/reference/android/os/Bundle) holding any arguments relevant for this request. May be **`null`** if the service provided no arguments. This value may be **`null`**. |

## addTextChangedListener

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void addTextChangedListener (TextWatcher (/reference/android/text/TextWatcher) watcher)
```

Adds a TextWatcher to the list of those whose methods are called whenever this TextView's text changes.

In 1.0, the **`TextWatcher#afterTextChanged`** (/reference/android/text/TextWatcher#afterTextChanged(android.text.Editable)) method was erroneously not called after **`setText(char[], int, int)`** (/reference/android/widget/TextView#setText(char[],%20int,%20int)) calls. Now, doing **`setText(char[], int, int)`** (/reference/android/widget/TextView#setText(char[],%20int,%20int)) if there are any text changed listeners forces the buffer type to Editable if it would not otherwise be and does call this method.

### Parameters

| | |
|---|---|
| `watcher` | `TextWatcher` |

## append

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void append (CharSequence (/reference/java/lang/CharSequence) text)
```

Convenience method to append the specified text to the TextView's display buffer, upgrading it to **`TextView.BufferType.EDITABLE`** (/reference/android/widget/TextView.BufferType#EDITABLE) if it was not already editable.

### Parameters

| | |
|---|---|
| `text` | **`CharSequence`**: text to be appended to the already displayed text |

## append

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void append (CharSequence (/reference/java/lang/CharSequence) text,
                int start,
                int end)
```

Convenience method to append the specified text slice to the TextView's display buffer, upgrading it to **`TextView.BufferType.EDITABLE`**

(/reference/android/widget/TextView.BufferType#EDITABLE) if it was not already editable.

### Parameters

| | |
|---|---|
| `text` | **`CharSequence`**: text to be appended to the already displayed text |
| `start` | **`int`**: the index of the first character in the **`text`** |
| `end` | **`int`**: the index of the character following the last character in the **`text`** |

### See also:

`Appendable.append(CharSequence, int, int)` (/reference/java/lang/Appendable#append(java.lang.CharSequence,%20int,%20int))

## autofill

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void autofill (AutofillValue (/reference/android/view/autofill/AutofillValue) value)
```

Automatically fills the content of this view with the **`value`**.

Views support the Autofill Framework mainly by:

- Providing the metadata defining what the view means and how it can be autofilled.

- Implementing the methods that autofill the view.

**onProvideAutofillStructure(android.view.ViewStructure, int)** (/reference/android/view/View#onProvideAutofillStructure(android.view.ViewStructure,%20int)) is responsible for the former, this method is responsible for latter.

This method does nothing by default, but when overridden it typically:

1. Checks if the provided value matches the expected type (which is defined by **getAutofillType()** (/reference/android/view/View#getAutofillType())).
2. Checks if the view is editable - if it isn't, it should return right away.
3. Call the proper getter method on **AutofillValue** (/reference/android/view/autofill/AutofillValue) to fetch the actual value.
4. Pass the actual value to the equivalent setter in the view.

For example, a text-field view could implement the method this way:

```
@Override
public void autofill(AutofillValue value) {
  if (!value.isText() || !this.isEditable()) {
     return;
  }
  CharSequence text = value.getTextValue();
  if (text != null) {
    this.setText(text);
  }
}
```

If the value is updated asynchronously, the next call to **`AutofillManager#notifyValueChanged(View)`** (/reference/android/view/autofill/AutofillManager#notifyValueChanged(android.view.View)) must happen **after** the value was changed to the autofilled value. If not, the view will not be considered autofilled.

**Note:** After this method is called, the value returned by **`getAutofillValue()`** (/reference/android/view/View#getAutofillValue()) must be equal to the **`value`** passed to it, otherwise the view will not be highlighted as autofilled.

### Parameters

| | |
|---|---|
| **`value`** | **`AutofillValue`**: value to be autofilled. |

## beginBatchEdit

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void beginBatchEdit ()
```

## bringPointIntoView

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean bringPointIntoView (int offset)
```

Move the point, specified by the offset, into the view if it is needed. This has to be called after layout. Returns true if anything changed.

### Parameters

| | |
|---|---|
| `offset` | `int` |

### Returns

`boolean`

## cancelLongPress

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void cancelLongPress ()
```

Cancels a pending long press. Your subclass can use this if you want the context menu to come up if the user presses and holds at the same place, but you don't want it to come up if they press and then move around enough to cause scrolling.

## clearComposingText

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void clearComposingText ()
```

Use `BaseInputConnection#removeComposingSpans` (/reference/android/view/inputmethod/BaseInputConnection#removeComposingSpans(android.text.Spannable)) to remove any IME composing state from this text view.

## computeScroll

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void computeScroll ()
```

Called by a parent to request that a child update its values for mScrollX and mScrollY if necessary. This will typically be done if the child is animating a scroll using a `Scroller` (/reference/android/widget/Scroller) object.

## debug

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void debug (int depth)
```

### Parameters

| | |
|---|---|
| **depth** | **int** |

## didTouchFocusSelect

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean didTouchFocusSelect ()
```

Returns true, only while processing a touch gesture, if the initial touch down event caused focus to move to the text view and as a result its selection changed. Only valid

while processing the touch gesture of interest, in an editable text view.

### Returns

**boolean**

## drawableHotspotChanged

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void drawableHotspotChanged (float x,
                float y)
```

This function is called whenever the view hotspot changes and needs to be propagated to drawables or child views managed by the view.

Dispatching to child views is handled by **`dispatchDrawableHotspotChanged(float, float)`** (/reference/android/view/View#dispatchDrawableHotspotChanged(float,%20float)).

Be sure to call through to the superclass when overriding this function.
If you override this method you *must* call through to the superclass implementation.

### Parameters

| | |
|---|---|
| **x** | **`float`**: hotspot x coordinate |
| **y** | **`float`**: hotspot y coordinate |

## endBatchEdit

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void endBatchEdit ()
```

## extractText

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean extractText (ExtractedTextRequest (/reference/android/view/inputmethod/ExtractedTextRequest) request,
                ExtractedText (/reference/android/view/inputmethod/ExtractedText) outText)
```

If this TextView contains editable content, extract a portion of it based on the information in ***request*** in to ***outText***.

### Parameters

| | |
|---|---|
| **request** | **ExtractedTextRequest** |
| **outText** | **ExtractedText** |

### Returns

| | |
|---|---|
| **boolean** | Returns true if the text was successfully extracted, else false. |

## findViewsWithText

Added in API level 14 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void findViewsWithText (ArrayList<View> outViews,
                CharSequence searched,
                int flags)
```

Finds the Views that contain given text. The containment is case insensitive. The search is performed by either the text that the View renders or the content description that describes the view for accessibility purposes and the view does not render or both. Clients can specify how the search is to be performed via passing the **FIND_VIEWS_WITH_TEXT** (/reference/android/view/View#FIND_VIEWS_WITH_TEXT) and **FIND_VIEWS_WITH_CONTENT_DESCRIPTION** (/reference/android/view/View#FIND_VIEWS_WITH_CONTENT_DESCRIPTION) flags.

### Parameters

| | |
|---|---|
| **outViews** | **ArrayList**: The output list of matching Views. |
| **searched** | **CharSequence**: The text to match against. |
| **flags** | **int**: Value is either **0** or a combination of **View.FIND_VIEWS_WITH_TEXT** (/reference/android/view/View#FIND_VIEWS_WITH_TEXT), and **View.FIND_VIEWS_WITH_CONTENT_DESCRIPTION** (/reference/android/view/View#FIND_VIEWS_WITH_CONTENT_DESCRIPTION) |

## getAccessibilityClassName

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public CharSequence (/reference/java/lang/CharSequence) getAccessibilityClassName ()
```

Return the class name of this object to be used for accessibility purposes. Subclasses should only override this if they are implementing something that should be seen as a completely new class of view when used by accessibility, unrelated to the class it is deriving from. This is used to fill in `AccessibilityNodeInfo#setClassName` (/reference/android/view/accessibility/AccessibilityNodeInfo#setClassName(java.lang.CharSequence)).

### Returns

`CharSequence` (/reference/java/lang/CharSequence)

## getAutoLinkMask

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final int getAutoLinkMask ()
```

Gets the autolink mask of the text. See `Linkify#ALL` (/reference/android/text/util/Linkify#ALL) and peers for possible values.

### Related XML Attributes:

android:autoLink (/reference/android/widget/TextView#attr_android:autoLink)

### Returns

`int`

## getAutoSizeMaxTextSize

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getAutoSizeMaxTextSize ()
```

### Related XML Attributes:

android:autoSizeMaxTextSize (/reference/android/widget/TextView#attr_android:autoSizeMaxTextSize)

### Returns

| | |
|---|---|
| `int` | the current auto-size maximum text size in pixels (the default is 112sp). Note that if auto-size has not been configured this function returns `-1`. |

### See also:

`setAutoSizeTextTypeUniformWithConfiguration(int, int, int, int)` (/reference/android/widget/TextView#setAutoSizeTextTypeUniformWithConfiguration(int,%20int,%20int,%20int))

`setAutoSizeTextTypeUniformWithPresetSizes(int[], int)` (/reference/android/widget/TextView#setAutoSizeTextTypeUniformWithPresetSizes(int[],%20int))

## getAutoSizeMinTextSize

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getAutoSizeMinTextSize ()
```

### Related XML Attributes:

android:autoSizeMinTextSize (/reference/android/widget/TextView#attr_android:autoSizeMinTextSize)

### Returns

| | |
|---|---|
| `int` | the current auto-size minimum text size in pixels (the default is 12sp). Note that if auto-size has not been configured this function returns `-1`. |

### See also:

`setAutoSizeTextTypeUniformWithConfiguration(int, int, int, int)` (/reference/android/widget/TextView#setAutoSizeTextTypeUniformWithConfiguration(int,%20int,%20int,%20int))

`setAutoSizeTextTypeUniformWithPresetSizes(int[], int)` (/reference/android/widget/TextView#setAutoSizeTextTypeUniformWithPresetSizes(int[],%20int))

## getAutoSizeStepGranularity

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getAutoSizeStepGranularity ()
```

### Related XML Attributes:

android:autoSizeStepGranularity (/reference/android/widget/TextView#attr_android:autoSizeStepGranularity)

### Returns

| | |
|---|---|
| `int` | the current auto-size step granularity in pixels. |

### See also:

`setAutoSizeTextTypeUniformWithConfiguration(int, int, int, int)`
(/reference/android/widget/TextView#setAutoSizeTextTypeUniformWithConfiguration(int,%20int,%20int,%20int))

## getAutoSizeTextAvailableSizes

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int[] getAutoSizeTextAvailableSizes ()
```

### Returns

| | |
|---|---|
| `int[]` | the current auto-size `int` sizes array (in pixels). |

### See also:

`setAutoSizeTextTypeUniformWithConfiguration(int, int, int, int)`
(/reference/android/widget/TextView#setAutoSizeTextTypeUniformWithConfiguration(int,%20int,%20int,%20int))

`setAutoSizeTextTypeUniformWithPresetSizes(int[], int)` (/reference/android/widget/TextView#setAutoSizeTextTypeUniformWithPresetSizes(int[],%20int))

## getAutoSizeTextType

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getAutoSizeTextType ()
```

Returns the type of auto-size set for this widget.

**Related XML Attributes:**

android:autoSizeTextType (/reference/android/widget/TextView#attr_android:autoSizeTextType)

### Returns

| | |
|---|---|
| `int` | an `int` corresponding to one of the auto-size types: `TextView#AUTO_SIZE_TEXT_TYPE_NONE` (/reference/android/widget/TextView#AUTO_SIZE_TEXT_TYPE_NONE) or `TextView#AUTO_SIZE_TEXT_TYPE_UNIFORM` (/reference/android/widget/TextView#AUTO_SIZE_TEXT_TYPE_UNIFORM) Value is `AUTO_SIZE_TEXT_TYPE_NONE` (/reference/android/widget/TextView#AUTO_SIZE_TEXT_TYPE_NONE), or `AUTO_SIZE_TEXT_TYPE_UNIFORM` (/reference/android/widget/TextView#AUTO_SIZE_TEXT_TYPE_UNIFORM) |

### See also:

`setAutoSizeTextTypeWithDefaults(int)` (/reference/android/widget/TextView#setAutoSizeTextTypeWithDefaults(int))

`setAutoSizeTextTypeUniformWithConfiguration(int, int, int, int)`
(/reference/android/widget/TextView#setAutoSizeTextTypeUniformWithConfiguration(int,%20int,%20int,%20int))

`setAutoSizeTextTypeUniformWithPresetSizes(int[], int)` (/reference/android/widget/TextView#setAutoSizeTextTypeUniformWithPresetSizes(int[],%20int))

## getAutofillType

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getAutofillType ()
```

Describes the autofill type of this view, so an **`AutofillService`** (/reference/android/service/autofill/AutofillService) can create the proper **`AutofillValue`** (/reference/android/view/autofill/AutofillValue) when autofilling the view.

By default returns **`AUTOFILL_TYPE_NONE`** (/reference/android/view/View#AUTOFILL_TYPE_NONE), but views should override it to properly support the Autofill Framework.

### Returns

| | |
|---|---|
| **`int`** | Value is **`View.AUTOFILL_TYPE_NONE`** (/reference/android/view/View#AUTOFILL_TYPE_NONE), **`View.AUTOFILL_TYPE_TEXT`** (/reference/android/view/View#AUTOFILL_TYPE_TEXT), **`View.AUTOFILL_TYPE_TOGGLE`** (/reference/android/view/View#AUTOFILL_TYPE_TOGGLE), **`View.AUTOFILL_TYPE_LIST`** (/reference/android/view/View#AUTOFILL_TYPE_LIST), or **`View.AUTOFILL_TYPE_DATE`** (/reference/android/view/View#AUTOFILL_TYPE_DATE) |

## getAutofillValue

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public AutofillValue (/reference/android/view/autofill/AutofillValue) getAutofillValue ()
```

Gets the **`TextView`** (/reference/android/widget/TextView)'s current text for AutoFill. The value is trimmed to 100K `chars` if longer.

### Returns

| | |
|---|---|
| **`AutofillValue`** | current text, **`null`** if the text is not editable |

(/reference/android/view/autofill/AutofillValue)

**See also:**

`View.getAutofillValue()` (/reference/android/view/View#getAutofillValue())

## getBaseline

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getBaseline ()
```

Return the offset of the widget's text baseline from the widget's top boundary. If this widget does not support baseline alignment, this method returns -1.

### Returns

| | |
|---|---|
| `int` | the offset of the baseline within the widget's bounds or -1 if baseline alignment is not supported |

## getBreakStrategy

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getBreakStrategy ()
```

Gets the current strategy for breaking paragraphs into lines.

### Related XML Attributes:

android:breakStrategy (/reference/android/widget/TextView#attr_android:breakStrategy)

### Returns

| | |
|---|---|
| `int` | the current strategy for breaking paragraphs into lines. Value is `LineBreaker.BREAK_STRATEGY_SIMPLE` (/reference/android/graphics/text/LineBreaker#BREAK_STRATEGY_SIMPLE), `LineBreaker.BREAK_STRATEGY_HIGH_QUALITY` (/reference/android/graphics/text/LineBreaker#BREAK_STRATEGY_HIGH_QUALITY), or `LineBreaker.BREAK_STRATEGY_BALANCED` (/reference/android/graphics/text/LineBreaker#BREAK_STRATEGY_BALANCED) |

### See also:

`setBreakStrategy(int)` (/reference/android/widget/TextView#setBreakStrategy(int))

## getCompoundDrawablePadding

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getCompoundDrawablePadding ()
```

Returns the padding between the compound drawables and the text.

### Related XML Attributes:

android:drawablePadding (/reference/android/widget/TextView#attr_android:drawablePadding)

### Returns

`int`

## getCompoundDrawableTintBlendMode

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public BlendMode (/reference/android/graphics/BlendMode) getCompoundDrawableTintBlendMode ()`

Returns the blending mode used to apply the tint to the compound drawables, if specified.

### Related XML Attributes:

android:drawableTintMode (/reference/android/widget/TextView#attr_android:drawableTintMode)

### Returns

| | |
|---|---|
| **`BlendMode`** (/reference/android/graphics/BlendMode) | the blending mode used to apply the tint to the compound drawables This value may be **`null`**. |

### See also:

**`setCompoundDrawableTintBlendMode(BlendMode)`** (/reference/android/widget/TextView#setCompoundDrawableTintBlendMode(android.graphics.BlendMode))

## getCompoundDrawableTintList

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public ColorStateList (/reference/android/content/res/ColorStateList) getCompoundDrawableTintList ()`

**Related XML Attributes:**

android:drawableTint (/reference/android/widget/TextView#attr_android:drawableTint)

**Returns**

| | |
|---|---|
| `ColorStateList` (/reference/android/content/res/ColorStateList) | the tint applied to the compound drawables |

**See also:**

`setCompoundDrawableTintList(ColorStateList)` (/reference/android/widget/TextView#setCompoundDrawableTintList(android.content.res.ColorStateList))

## getCompoundDrawableTintMode

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public PorterDuff.Mode (/reference/android/graphics/PorterDuff.Mode) getCompoundDrawableTintMode ()`

Returns the blending mode used to apply the tint to the compound drawables, if specified.

**Related XML Attributes:**

android:drawableTintMode (/reference/android/widget/TextView#attr_android:drawableTintMode)

### Returns

| | |
|---|---|
| `PorterDuff.Mode` (/reference/android/graphics/PorterDuff.Mode) | the blending mode used to apply the tint to the compound drawables |

### See also:

`setCompoundDrawableTintMode(PorterDuff.Mode)` (/reference/android/widget/TextView#setCompoundDrawableTintMode(android.graphics.PorterDuff.Mode))

## getCompoundDrawables

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public Drawable[] (/reference/android/graphics/drawable/Drawable) getCompoundDrawables ()
```

Returns drawables for the left, top, right, and bottom borders.

### Related XML Attributes:

android:drawableLeft (/reference/android/widget/TextView#attr_android:drawableLeft)

android:drawableTop (/reference/android/widget/TextView#attr_android:drawableTop)

android:drawableRight (/reference/android/widget/TextView#attr_android:drawableRight)

android:drawableBottom (/reference/android/widget/TextView#attr_android:drawableBottom)

### Returns

| | |
|---|---|
| `Drawable[]` (/reference/android/graphics/drawable/Drawable) | This value will never be **`null`**. |

## getCompoundDrawablesRelative

Added in API level 17 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public Drawable[] (/reference/android/graphics/drawable/Drawable) getCompoundDrawablesRelative ()`

Returns drawables for the start, top, end, and bottom borders.

### Related XML Attributes:

android:drawableStart (/reference/android/widget/TextView#attr_android:drawableStart)

android:drawableTop (/reference/android/widget/TextView#attr_android:drawableTop)

android:drawableEnd (/reference/android/widget/TextView#attr_android:drawableEnd)

android:drawableBottom (/reference/android/widget/TextView#attr_android:drawableBottom)

### Returns

| | |
|---|---|
| `Drawable[]` (/reference/android/graphics/drawable/Drawable) | This value will never be `null`. |

## getCompoundPaddingBottom

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getCompoundPaddingBottom ()
```

Returns the bottom padding of the view, plus space for the bottom Drawable if any.

### Returns

`int`

## getCompoundPaddingEnd

Added in API level 17 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getCompoundPaddingEnd ()
```

Returns the end padding of the view, plus space for the end Drawable if any.

### Returns

`int`

## getCompoundPaddingLeft

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getCompoundPaddingLeft ()
```

Returns the left padding of the view, plus space for the left Drawable if any.

### Returns

`int`

## getCompoundPaddingRight

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getCompoundPaddingRight ()
```

Returns the right padding of the view, plus space for the right Drawable if any.

### Returns

`int`

## getCompoundPaddingStart

Added in API level 17 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getCompoundPaddingStart ()
```

Returns the start padding of the view, plus space for the start Drawable if any.

### Returns

**int**

## getCompoundPaddingTop

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getCompoundPaddingTop ()
```

Returns the top padding of the view, plus space for the top Drawable if any.

### Returns

**int**

## getCurrentHintTextColor

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final int getCurrentHintTextColor ()
```

Return the current color selected to paint the hint text.

### Returns

| | |
|---|---|
| **int** | Returns the current hint text color. |

## getCurrentTextColor

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final int getCurrentTextColor ()
```

Return the current color selected for normal text.

### Returns

| | |
|---|---|
| **int** | Returns the current text color. |

## getCustomInsertionActionModeCallback

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public ActionMode.Callback (/reference/android/view/ActionMode.Callback) getCustomInsertionActionModeCallback ()
```

Retrieves the value set in **`setCustomInsertionActionModeCallback(ActionMode.Callback)`** (/reference/android/widget/TextView#setCustomInsertionActionModeCallback(android.view.ActionMode.Callback)). Default is null.

### Returns

| | |
|---|---|
| **`ActionMode.Callback`** (/reference/android/view/ActionMode.Callback) | The current custom insertion callback. |

## getCustomSelectionActionModeCallback

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public ActionMode.Callback (/reference/android/view/ActionMode.Callback) getCustomSelectionActionModeCallback ()`

Retrieves the value set in **`setCustomSelectionActionModeCallback(ActionMode.Callback)`** (/reference/android/widget/TextView#setCustomSelectionActionModeCallback(android.view.ActionMode.Callback)). Default is null.

### Returns

| | |
|---|---|
| **`ActionMode.Callback`** (/reference/android/view/ActionMode.Callback) | The current custom selection callback. |

## getEditableText

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public Editable (/reference/android/text/Editable) getEditableText ()`

Return the text that TextView is displaying as an Editable object. If the text is not editable, null is returned.

### Returns

---

`Editable`
(/reference/android/text/Editable
)

### See also:

`getText()` (/reference/android/widget/TextView#getText())

## getEllipsize

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public TextUtils.TruncateAt (/reference/android/text/TextUtils.TruncateAt) getEllipsize ()`

Returns where, if anywhere, words that are longer than the view is wide should be ellipsized.

### Returns

---

**TextUtils.TruncateAt**
(/reference/android/text/TextUtils.TruncateAt)

## getError

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public CharSequence (/reference/java/lang/CharSequence) getError ()
```

Returns the error message that was set to be displayed with **setError(CharSequence)** (/reference/android/widget/TextView#setError(java.lang.CharSequence)), or **null** if no error was set or if it the error was cleared by the widget after user input.

### Returns

**CharSequence**
(/reference/java/lang/CharSequence)

## getExtendedPaddingBottom

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getExtendedPaddingBottom ()
```

Returns the extended bottom padding of the view, including both the bottom Drawable if any and any extra space to keep more than maxLines of text from showing. It is

only valid to call this after measuring.

### Returns

**int**

## getExtendedPaddingTop

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getExtendedPaddingTop ()
```

Returns the extended top padding of the view, including both the top Drawable if any and any extra space to keep more than maxLines of text from showing. It is only valid to call this after measuring.

### Returns

**int**

## getFilters

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

public InputFilter[] (/reference/android/text/InputFilter) getFilters ()

Returns the current list of input filters.

**Related XML Attributes:**

android:maxLength (/reference/android/widget/TextView#attr_android:maxLength)

### Returns

---

`InputFilter[]`
(/reference/android/text/InputFilt
er)

## getFirstBaselineToTopHeight

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getFirstBaselineToTopHeight ()
```

Returns the distance between the first text baseline and the top of this TextView.

**Related XML Attributes:**

android:firstBaselineToTopHeight (/reference/android/widget/TextView#attr_android:firstBaselineToTopHeight)

### Returns

---

`int`

**See also:**

`setFirstBaselineToTopHeight(int)` (/reference/android/widget/TextView#setFirstBaselineToTopHeight(int))

## getFocusedRect

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void getFocusedRect (Rect (/reference/android/graphics/Rect) r)
```

When a view has focus and the user navigates away from it, the next view is searched for starting from the rectangle filled in by this method. By default, the rectangle is the `getDrawingRect(android.graphics.Rect)` (/reference/android/view/View#getDrawingRect(android.graphics.Rect))) of the view. However, if your view maintains some idea of internal selection, such as a cursor, or a selected row or column, you should override this method and fill in a more specific rectangle.

### Parameters

| | |
|---|---|
| `r` | **`Rect`**: The rectangle to fill in, in this view's coordinates. |

## getFontFeatureSettings

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public String (/reference/java/lang/String) getFontFeatureSettings ()
```

Returns the font feature settings. The format is the same as the CSS font-feature-settings attribute: https://www.w3.org/TR/css-fonts-3/#font-feature-settings-prop

(https://www.w3.org/TR/css-fonts-3/#font-feature-settings-prop)

### Returns

| | |
|---|---|
| String (/reference/java/lang/String) | the currently set font feature settings. Default is null. |

### See also:

setFontFeatureSettings(String) (/reference/android/widget/TextView#setFontFeatureSettings(java.lang.String))

Paint.setFontFeatureSettings(String) (/reference/android/graphics/Paint#setFontFeatureSettings(java.lang.String))

## getFontVariationSettings

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public String (/reference/java/lang/String) getFontVariationSettings ()
```

Returns the font variation settings.

### Returns

| | |
|---|---|
| String (/reference/java/lang/String) | the currently set font variation settings. Returns null if no variation is specified. |

### See also:

`setFontVariationSettings(String)` (/reference/android/widget/TextView#setFontVariationSettings(java.lang.String))

`Paint.setFontVariationSettings(String)` (/reference/android/graphics/Paint#setFontVariationSettings(java.lang.String))

## getFreezesText

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean getFreezesText ()
```

Return whether this text view is including its entire text contents in frozen icicles. For `EditText` (/reference/android/widget/EditText) it always returns true.

### Returns

| | |
|---|---|
| **boolean** | Returns true if text is included, false if it isn't. |

### See also:

`setFreezesText(boolean)` (/reference/android/widget/TextView#setFreezesText(boolean))

## getGravity

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getGravity ()
```

Returns the horizontal and vertical alignment of this TextView.

### Related XML Attributes:

android:gravity (/reference/android/widget/TextView#attr_android:gravity)

### Returns

`int`

### See also:

`Gravity` (/reference/android/view/Gravity)

## getHighlightColor

Added in API level 16 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getHighlightColor ()
```

### Related XML Attributes:

android:textColorHighlight (/reference/android/widget/TextView#attr_android:textColorHighlight)

### Returns

| | |
|---|---|
| `int` | the color used to display the selection highlight |

**See also:**

`setHighlightColor(int)` (/reference/android/widget/TextView#setHighlightColor(int))

## getHint

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public CharSequence (/reference/java/lang/CharSequence) getHint ()`

Returns the hint that is displayed when the text of the TextView is empty.

**Related XML Attributes:**

android:hint (/reference/android/widget/TextView#attr_android:hint)

**Returns**

---

`CharSequence`
(/reference/java/lang/CharSeque
nce)

## getHintTextColors

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public final ColorStateList (/reference/android/content/res/ColorStateList) getHintTextColors ()`

### Related XML Attributes:

android:textColorHint (/reference/android/widget/TextView#attr_android:textColorHint)

### Returns

| | |
|---|---|
| `ColorStateList` (/reference/android/content/res/ColorStateList) | the color of the hint text, for the different states of this TextView. |

### See also:

`setHintTextColor(ColorStateList)` (/reference/android/widget/TextView#setHintTextColor(android.content.res.ColorStateList))

`setHintTextColor(int)` (/reference/android/widget/TextView#setHintTextColor(int))

`setTextColor(ColorStateList)` (/reference/android/widget/TextView#setTextColor(android.content.res.ColorStateList))

`setLinkTextColor(ColorStateList)` (/reference/android/widget/TextView#setLinkTextColor(android.content.res.ColorStateList))

## getHyphenationFrequency

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getHyphenationFrequency ()
```

Gets the current frequency of automatic hyphenation to be used when determining word breaks.

### Related XML Attributes:

android:hyphenationFrequency (/reference/android/widget/TextView#attr_android:hyphenationFrequency)

### Returns

| | |
|---|---|
| `int` | the current frequency of automatic hyphenation to be used when determining word breaks. Value is `Layout.HYPHENATION_FREQUENCY_NORMAL` (/reference/android/text/Layout#HYPHENATION_FREQUENCY_NORMAL), `Layout.HYPHENATION_FREQUENCY_FULL` (/reference/android/text/Layout#HYPHENATION_FREQUENCY_FULL), or `Layout.HYPHENATION_FREQUENCY_NONE` (/reference/android/text/Layout#HYPHENATION_FREQUENCY_NONE) |

### See also:

`setHyphenationFrequency(int)` (/reference/android/widget/TextView#setHyphenationFrequency(int))

## getImeActionId

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getImeActionId ()
```

Get the IME action ID previous set with `setImeActionLabel(CharSequence, int)` (/reference/android/widget/TextView#setImeActionLabel(java.lang.CharSequence,%20int)).

### Returns

| |
|---|
| `int` |

### See also:

`setImeActionLabel(CharSequence, int)` (/reference/android/widget/TextView#setImeActionLabel(java.lang.CharSequence,%20int))

`EditorInfo` (/reference/android/view/inputmethod/EditorInfo)

## getImeActionLabel

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public CharSequence (/reference/java/lang/CharSequence) getImeActionLabel ()
```

Get the IME action label previous set with `setImeActionLabel(CharSequence, int)` (/reference/android/widget/TextView#setImeActionLabel(java.lang.CharSequence,%20int)).

### Returns

---

`CharSequence`
(/reference/java/lang/CharSeque
nce)

### See also:

`setImeActionLabel(CharSequence, int)` (/reference/android/widget/TextView#setImeActionLabel(java.lang.CharSequence,%20int))

`EditorInfo` (/reference/android/view/inputmethod/EditorInfo)

## getImeHintLocales

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public LocaleList (/reference/android/os/LocaleList) getImeHintLocales ()
```

### Returns

| | |
|---|---|
| LocaleList (/reference/android/os/LocaleList) | The current languages list "hint". **null** when no "hint" is available. |

### See also:

setImeHintLocales(LocaleList) (/reference/android/widget/TextView#setImeHintLocales(android.os.LocaleList))

EditorInfo.hintLocales (/reference/android/view/inputmethod/EditorInfo#hintLocales)

## getImeOptions

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getImeOptions ()
```

Get the type of the Input Method Editor (IME).

### Returns

| | |
|---|---|
| `int` | the type of the IME |

**See also:**

`setImeOptions(int)` (/reference/android/widget/TextView#setImeOptions(int))

`EditorInfo` (/reference/android/view/inputmethod/EditorInfo)

## getIncludeFontPadding

Added in API level 16 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean getIncludeFontPadding ()
```

Gets whether the TextView includes extra top and bottom padding to make room for accents that go above the normal ascent and descent.

**Related XML Attributes:**

android:includeFontPadding (/reference/android/widget/TextView#attr_android:includeFontPadding)

**Returns**

| |
|---|
| `boolean` |

**See also:**

`setIncludeFontPadding(boolean)` (/reference/android/widget/TextView#setIncludeFontPadding(boolean))

# getInputExtras

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public Bundle (/reference/android/os/Bundle) getInputExtras (boolean create)`

Retrieve the input extras currently associated with the text view, which can be viewed as well as modified.

**Related XML Attributes:**

android:editorExtras (/reference/android/widget/TextView#attr_android:editorExtras)

## Parameters

| | |
|---|---|
| `create` | **`boolean`**: If true, the extras will be created if they don't already exist. Otherwise, null will be returned if none have been created. |

## Returns

`Bundle`
(/reference/android/os/Bundle)

## See also:

`setInputExtras(int)` (/reference/android/widget/TextView#setInputExtras(int))

`EditorInfo.extras` (/reference/android/view/inputmethod/EditorInfo#extras)

## getInputType

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getInputType ()
```

Get the type of the editable content.

### Returns

`int`

### See also:

`setInputType(int)` (/reference/android/widget/TextView#setInputType(int))

`InputType` (/reference/android/text/InputType)

## getJustificationMode

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getJustificationMode ()
```

### Returns

| | |
|---|---|
| `int` | true if currently paragraph justification mode. Value is `LineBreaker.JUSTIFICATION_MODE_NONE` (/reference/android/graphics/text/LineBreaker#JUSTIFICATION_MODE_NONE), or `LineBreaker.JUSTIFICATION_MODE_INTER_WORD` |

(/reference/android/graphics/text/LineBreaker#JUSTIFICATION_MODE_INTER_WORD)

**See also:**

`setJustificationMode(int)` (/reference/android/widget/TextView#setJustificationMode(int))

## getKeyListener

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final KeyListener (/reference/android/text/method/KeyListener) getKeyListener ()
```

Gets the current `KeyListener` (/reference/android/text/method/KeyListener) for the TextView. This will frequently be null for non-EditText TextViews.

**Related XML Attributes:**

android:numeric (/reference/android/widget/TextView#attr_android:numeric)

android:digits (/reference/android/widget/TextView#attr_android:digits)

android:phoneNumber (/reference/android/widget/TextView#attr_android:phoneNumber)

android:inputMethod (/reference/android/widget/TextView#attr_android:inputMethod)

android:capitalize (/reference/android/widget/TextView#attr_android:capitalize)

android:autoText (/reference/android/widget/TextView#attr_android:autoText)

**Returns**

| | |
|---|---|
| `KeyListener` (/reference/android/text/method/KeyListener) | the current key listener for this TextView. |

## getLastBaselineToBottomHeight

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getLastBaselineToBottomHeight ()
```

Returns the distance between the last text baseline and the bottom of this TextView.

**Related XML Attributes:**

android:lastBaselineToBottomHeight (/reference/android/widget/TextView#attr_android:lastBaselineToBottomHeight)

### Returns

`int`

### See also:

`setLastBaselineToBottomHeight(int)` (/reference/android/widget/TextView#setLastBaselineToBottomHeight(int))

## getLayout

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public final Layout (/reference/android/text/Layout) getLayout ()`

Gets the **`Layout`** (/reference/android/text/Layout) that is currently being used to display the text. This value can be null if the text or width has recently changed.

### Returns

| | |
|---|---|
| **`Layout`** (/reference/android/text/Layout) | The Layout that is currently being used to display the text. |

## getLetterSpacing

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public float getLetterSpacing ()`

Gets the text letter-space value, which determines the spacing between characters. The value returned is in ems. Normally, this value is 0.0.

### Returns

| | |
|---|---|
| **`float`** | The text letter-space value in ems. |

### See also:

**`setLetterSpacing(float)`** (/reference/android/widget/TextView#setLetterSpacing(float))

`Paint.setLetterSpacing(float)` (/reference/android/graphics/Paint#setLetterSpacing(float))

## getLineBounds

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getLineBounds (int line,
                Rect (/reference/android/graphics/Rect) bounds)
```

Return the baseline for the specified line (0...getLineCount() - 1) If bounds is not null, return the top, left, right, bottom extents of the specified line in it. If the internal Layout has not been built, return 0 and set bounds to (0, 0, 0, 0)

### Parameters

| | |
|---|---|
| **line** | **int**: which line to examine (0..getLineCount() - 1) |
| **bounds** | **Rect**: Optional. If not null, it returns the extent of the line |

### Returns

| | |
|---|---|
| **int** | the Y-coordinate of the baseline |

## getLineCount

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getLineCount ()
```

Return the number of lines of text, or 0 if the internal Layout has not been built.

### Returns

| | |
|---|---|
| **int** | |

## getLineHeight

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getLineHeight ()
```

Gets the vertical distance between lines of text, in pixels. Note that markup within the text can cause individual lines to be taller or shorter than this height, and the layout may contain additional first-or last-line padding.

### Returns

| | |
|---|---|
| **int** | The height of one standard line in pixels. |

## getLineSpacingExtra

Added in API level 16 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public float getLineSpacingExtra ()
```

Gets the line spacing extra space

**Related XML Attributes:**

android:lineSpacingExtra (/reference/android/widget/TextView#attr_android:lineSpacingExtra)

**Returns**

| | |
|---|---|
| `float` | the extra space that is added to the height of each lines of this TextView. |

**See also:**

`setLineSpacing(float, float)` (/reference/android/widget/TextView#setLineSpacing(float,%20float))

`getLineSpacingMultiplier()` (/reference/android/widget/TextView#getLineSpacingMultiplier())

## getLineSpacingMultiplier

Added in API level 16 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public float getLineSpacingMultiplier ()
```

Gets the line spacing multiplier

### Related XML Attributes:

android:lineSpacingMultiplier (/reference/android/widget/TextView#attr_android:lineSpacingMultiplier)

### Returns

| | |
|---|---|
| `float` | the value by which each line's height is multiplied to get its actual height. |

### See also:

`setLineSpacing(float, float)` (/reference/android/widget/TextView#setLineSpacing(float,%20float))

`getLineSpacingExtra()` (/reference/android/widget/TextView#getLineSpacingExtra())

## getLinkTextColors

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final ColorStateList (/reference/android/content/res/ColorStateList) getLinkTextColors ()
```

### Related XML Attributes:

android:textColorLink (/reference/android/widget/TextView#attr_android:textColorLink)

### Returns

| | |
|---|---|
| `ColorStateList` | the list of colors used to paint the links in the text, for the different states of this TextView |

(/reference/android/content/res/ColorStateList)

**See also:**

`setLinkTextColor(ColorStateList)` (/reference/android/widget/TextView#setLinkTextColor(android.content.res.ColorStateList))

`setLinkTextColor(int)` (/reference/android/widget/TextView#setLinkTextColor(int))

## getLinksClickable

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final boolean getLinksClickable ()
```

Returns whether the movement method will automatically be set to `LinkMovementMethod` (/reference/android/text/method/LinkMovementMethod) if `setAutoLinkMask(int)` (/reference/android/widget/TextView#setAutoLinkMask(int)) has been set to nonzero and links are detected in `setText(char[], int, int)` (/reference/android/widget/TextView#setText(char[],%20int,%20int)). The default is true.

**Related XML Attributes:**

android:linksClickable (/reference/android/widget/TextView#attr_android:linksClickable)

### Returns

`boolean`

## getMarqueeRepeatLimit

Added in API level 16 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getMarqueeRepeatLimit ()
```

Gets the number of times the marquee animation is repeated. Only meaningful if the TextView has marquee enabled.

**Related XML Attributes:**

android:marqueeRepeatLimit (/reference/android/widget/TextView#attr_android:marqueeRepeatLimit)

### Returns

| | |
|---|---|
| `int` | the number of times the marquee animation is repeated. -1 if the animation repeats indefinitely |

**See also:**

`setMarqueeRepeatLimit(int)` (/reference/android/widget/TextView#setMarqueeRepeatLimit(int))

## getMaxEms

Added in API level 16 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getMaxEms ()
```

Returns the maximum width of TextView in terms of ems or -1 if the maximum width was set using `setMaxWidth(int)` (/reference/android/widget/TextView#setMaxWidth(int)) or `setWidth(int)` (/reference/android/widget/TextView#setWidth(int)).

### Related XML Attributes:

android:maxEms (/reference/android/widget/TextView#attr_android:maxEms)

### Returns

| | |
|---|---|
| `int` | the maximum width of TextView in terms of ems or -1 if the maximum width is not defined in ems |

### See also:

`setMaxEms(int)` (/reference/android/widget/TextView#setMaxEms(int))

`setEms(int)` (/reference/android/widget/TextView#setEms(int))

## getMaxHeight

Added in API level 16 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getMaxHeight ()
```

Returns the maximum height of TextView in terms of pixels or -1 if the maximum height was set using `setMaxLines(int)` (/reference/android/widget/TextView#setMaxLines(int)) or `setLines(int)` (/reference/android/widget/TextView#setLines(int)).

### Related XML Attributes:

android:maxHeight (/reference/android/widget/TextView#attr_android:maxHeight)

### Returns

| | |
|---|---|
| `int` | the maximum height of TextView in terms of pixels or -1 if the maximum height is not defined in pixels |

**See also:**

`setMaxHeight(int)` (/reference/android/widget/TextView#setMaxHeight(int))

`setHeight(int)` (/reference/android/widget/TextView#setHeight(int))

## getMaxLines

Added in API level 16 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getMaxLines ()
```

Returns the maximum height of TextView in terms of number of lines or -1 if the maximum height was set using `setMaxHeight(int)` (/reference/android/widget/TextView#setMaxHeight(int)) or `setHeight(int)` (/reference/android/widget/TextView#setHeight(int)).

**Related XML Attributes:**

android:maxLines (/reference/android/widget/TextView#attr_android:maxLines)

**Returns**

| | |
|---|---|
| `int` | the maximum height of TextView in terms of number of lines. -1 if the maximum height is not defined in lines. |

**See also:**

**`setMaxLines(int)`** (/reference/android/widget/TextView#setMaxLines(int))

**`setLines(int)`** (/reference/android/widget/TextView#setLines(int))

## getMaxWidth

Added in API level 16 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getMaxWidth ()
```

Returns the maximum width of TextView in terms of pixels or -1 if the maximum width was set using **`setMaxEms(int)`** (/reference/android/widget/TextView#setMaxEms(int)) or **`setEms(int)`** (/reference/android/widget/TextView#setEms(int)).

### Related XML Attributes:

android:maxWidth (/reference/android/widget/TextView#attr_android:maxWidth)

### Returns

| | |
|---|---|
| `int` | the maximum width of TextView in terms of pixels. -1 if the maximum width is not defined in pixels |

### See also:

**`setMaxWidth(int)`** (/reference/android/widget/TextView#setMaxWidth(int))

**`setWidth(int)`** (/reference/android/widget/TextView#setWidth(int))

## getMinEms

Added in API level 16 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getMinEms ()
```

Returns the minimum width of TextView in terms of ems or -1 if the minimum width was set using **`setMinWidth(int)`** (/reference/android/widget/TextView#setMinWidth(int)) or **`setWidth(int)`** (/reference/android/widget/TextView#setWidth(int)).

**Related XML Attributes:**

android:minEms (/reference/android/widget/TextView#attr_android:minEms)

### Returns

| | |
|---|---|
| **`int`** | the minimum width of TextView in terms of ems. -1 if the minimum width is not defined in ems |

**See also:**

**`setMinEms(int)`** (/reference/android/widget/TextView#setMinEms(int))

**`setEms(int)`** (/reference/android/widget/TextView#setEms(int))

## getMinHeight

Added in API level 16 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getMinHeight ()
```

Returns the minimum height of TextView in terms of pixels or -1 if the minimum height was set using **`setMinLines(int)`** (/reference/android/widget/TextView#setMinLines(int)) or **`setLines(int)`** (/reference/android/widget/TextView#setLines(int)).

**Related XML Attributes:**

android:minHeight (/reference/android/widget/TextView#attr_android:minHeight)

**Returns**

| | |
|---|---|
| `int` | the minimum height of TextView in terms of pixels or -1 if the minimum height is not defined in pixels |

**See also:**

**`setMinHeight(int)`** (/reference/android/widget/TextView#setMinHeight(int))

**`setHeight(int)`** (/reference/android/widget/TextView#setHeight(int))

## getMinLines

Added in API level 16 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getMinLines ()
```

Returns the minimum height of TextView in terms of number of lines or -1 if the minimum height was set using **`setMinHeight(int)`** (/reference/android/widget/TextView#setMinHeight(int)) or **`setHeight(int)`** (/reference/android/widget/TextView#setHeight(int)).

**Related XML Attributes:**

android:minLines (/reference/android/widget/TextView#attr_android:minLines)

### Returns

| | |
|---|---|
| `int` | the minimum height of TextView in terms of number of lines or -1 if the minimum height is not defined in lines |

### See also:

`setMinLines(int)` (/reference/android/widget/TextView#setMinLines(int))

`setLines(int)` (/reference/android/widget/TextView#setLines(int))

## getMinWidth

Added in API level 16 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getMinWidth ()
```

Returns the minimum width of TextView in terms of pixels or -1 if the minimum width was set using `setMinEms(int)` (/reference/android/widget/TextView#setMinEms(int)) or `setEms(int)` (/reference/android/widget/TextView#setEms(int)).

### Related XML Attributes:

android:minWidth (/reference/android/widget/TextView#attr_android:minWidth)

### Returns

| | |
|---|---|
| `int` | the minimum width of TextView in terms of pixels or -1 if the minimum width is not defined in pixels |

### See also:

**`setMinWidth(int)`** (/reference/android/widget/TextView#setMinWidth(int))

**`setWidth(int)`** (/reference/android/widget/TextView#setWidth(int))

## getMovementMethod

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public final MovementMethod (/reference/android/text/method/MovementMethod) getMovementMethod ()`

Gets the **`MovementMethod`** (/reference/android/text/method/MovementMethod) being used for this TextView, which provides positioning, scrolling, and text selection functionality. This will frequently be null for non-EditText TextViews.

### Returns

| | |
|---|---|
| **`MovementMethod`** (/reference/android/text/method/MovementMethod) | the movement method being used for this TextView. |

### See also:

**`MovementMethod`** (/reference/android/text/method/MovementMethod)

## getOffsetForPosition

Added in API level 14 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getOffsetForPosition (float x,
                float y)
```

Get the character offset closest to the specified absolute position. A typical use case is to pass the result of **`MotionEvent#getX()`** (/reference/android/view/MotionEvent#getX()) and **`MotionEvent#getY()`** (/reference/android/view/MotionEvent#getY()) to this method.

### Parameters

| | |
|---|---|
| x | **`float`**: The horizontal absolute position of a point on screen |
| y | **`float`**: The vertical absolute position of a point on screen |

### Returns

| | |
|---|---|
| **`int`** | the character offset for the character whose position is closest to the specified position. Returns -1 if there is no layout. |

## getPaint

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public` TextPaint (/reference/android/text/TextPaint) `getPaint ()`

Gets the `TextPaint` (/reference/android/text/TextPaint) used for the text. Use this only to consult the Paint's properties and not to change them.

### Returns

| | |
|---|---|
| `TextPaint` (/reference/android/text/TextPaint) | The base paint used for the text. |

## getPaintFlags

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getPaintFlags ()
```

Gets the flags on the Paint being used to display the text.

### Returns

| | |
|---|---|
| `int` | The flags on the Paint being used to display the text. |

### See also:

`Paint.getFlags()` (/reference/android/graphics/Paint#getFlags())

## getPrivateImeOptions

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public String (/reference/java/lang/String) getPrivateImeOptions ()`

Get the private type of the content.

### Returns

---

**`String`**
(/reference/java/lang/String)

### See also:

**`setPrivateImeOptions(String)`** (/reference/android/widget/TextView#setPrivateImeOptions(java.lang.String))

**`EditorInfo.privateImeOptions`** (/reference/android/view/inputmethod/EditorInfo#privateImeOptions)

## getSelectionEnd

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public int getSelectionEnd ()`

Convenience for **`Selection#getSelectionEnd`** (/reference/android/text/Selection#getSelectionEnd(java.lang.CharSequence)).

### Returns

---

**`int`**

## getSelectionStart

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getSelectionStart ()
```

Convenience for **`Selection#getSelectionStart`** (/reference/android/text/Selection#getSelectionStart(java.lang.CharSequence)).

### Returns

---

**`int`**

## getShadowColor

Added in API level 16 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getShadowColor ()
```

Gets the color of the shadow layer.

### Related XML Attributes:

android:shadowColor (/reference/android/widget/TextView#attr_android:shadowColor)

### Returns

| | |
|---|---|
| `int` | the color of the shadow layer |

### See also:

`setShadowLayer(float, float, float, int)` (/reference/android/widget/TextView#setShadowLayer(float,%20float,%20float,%20int))

## getShadowDx

Added in API level 16 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public float getShadowDx ()
```

### Related XML Attributes:

android:shadowDx (/reference/android/widget/TextView#attr_android:shadowDx)

### Returns

| | |
|---|---|
| `float` | the horizontal offset of the shadow layer |

### See also:

`setShadowLayer(float, float, float, int)` (/reference/android/widget/TextView#setShadowLayer(float,%20float,%20float,%20int))

## getShadowDy

Added in API level 16 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public float getShadowDy ()
```

Gets the vertical offset of the shadow layer.

**Related XML Attributes:**

android:shadowDy (/reference/android/widget/TextView#attr_android:shadowDy)

**Returns**

| | |
|---|---|
| `float` | The vertical offset of the shadow layer. |

**See also:**

`setShadowLayer(float, float, float, int)` (/reference/android/widget/TextView#setShadowLayer(float,%20float,%20float,%20int))

## getShadowRadius

Added in API level 16 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public float getShadowRadius ()
```

Gets the radius of the shadow layer.

**Related XML Attributes:**

android:shadowRadius (/reference/android/widget/TextView#attr_android:shadowRadius)

### Returns

| | |
|---|---|
| `float` | the radius of the shadow layer. If 0, the shadow layer is not visible |

### See also:

`setShadowLayer(float, float, float, int)` (/reference/android/widget/TextView#setShadowLayer(float,%20float,%20float,%20int))

## getShowSoftInputOnFocus

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final boolean getShowSoftInputOnFocus ()
```

Returns whether the soft input method will be made visible when this TextView gets focused. The default is true.

### Returns

| |
|---|
| `boolean` |

## getText

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public CharSequence (/reference/java/lang/CharSequence) getText ()`

Return the text that TextView is displaying. If **`setText(java.lang.CharSequence)`** (/reference/android/widget/TextView#setText(java.lang.CharSequence)) was called with an argument of **`BufferType.SPANNABLE`** (/reference/android/widget/TextView.BufferType#SPANNABLE) or **`BufferType.EDITABLE`** (/reference/android/widget/TextView.BufferType#EDITABLE), you can cast the return value from this method to Spannable or Editable, respectively.

The content of the return value should not be modified. If you want a modifiable one, you should make your own copy first.

**Related XML Attributes:**

android:text (/reference/android/widget/TextView#attr_android:text)

**Returns**

| | |
|---|---|
| **`CharSequence`** (/reference/java/lang/CharSequence) | The text displayed by the text view. |

## getTextClassifier

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public TextClassifier (/reference/android/view/textclassifier/TextClassifier) getTextClassifier ()`

Returns the **`TextClassifier`** (/reference/android/view/textclassifier/TextClassifier) used by this TextView. If no TextClassifier has been set, this TextView uses the default set by the **`TextClassificationManager`** (/reference/android/view/textclassifier/TextClassificationManager).

### Returns

| | |
|---|---|
| `TextClassifier` (/reference/android/view/textclassifier/TextClassifier) | This value will never be **`null`**. |

## getTextColors

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final ColorStateList (/reference/android/content/res/ColorStateList) getTextColors ()
```

Gets the text colors for the different states (normal, selected, focused) of the TextView.

### Related XML Attributes:

android:textColor (/reference/android/widget/TextView#attr_android:textColor)

### Returns

`ColorStateList` (/reference/android/content/res/ColorStateList)

### See also:

`setTextColor(ColorStateList)` (/reference/android/widget/TextView#setTextColor(android.content.res.ColorStateList))

`setTextColor(int)` (/reference/android/widget/TextView#setTextColor(int))

## getTextCursorDrawable

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public Drawable (/reference/android/graphics/drawable/Drawable) getTextCursorDrawable ()`

Returns the Drawable corresponding to the text cursor. Note that any change applied to the cursor Drawable will not be visible until the cursor is hidden and then drawn again.

**Related XML Attributes:**

android:textCursorDrawable (/reference/android/widget/TextView#attr_android:textCursorDrawable)

### Returns

| | |
|---|---|
| `Drawable` (/reference/android/graphics/drawable/Drawable) | the text cursor drawable This value may be **`null`**. |

### See also:

`setTextCursorDrawable(Drawable)` (/reference/android/widget/TextView#setTextCursorDrawable(android.graphics.drawable.Drawable))

`setTextCursorDrawable(int)` (/reference/android/widget/TextView#setTextCursorDrawable(int))

## getTextDirectionHeuristic

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public TextDirectionHeuristic (/reference/android/text/TextDirectionHeuristic) getTextDirectionHeuristic ()`

Returns resolved **`TextDirectionHeuristic`** (/reference/android/text/TextDirectionHeuristic) that will be used for text layout. The **`TextDirectionHeuristic`** (/reference/android/text/TextDirectionHeuristic) that is used by TextView is only available after **`View.getTextDirection()`** (/reference/android/view/View#getTextDirection()) and **`View.getLayoutDirection()`** (/reference/android/view/View#getLayoutDirection()) is resolved. Therefore the return value may not be the same as the one TextView uses if the View's layout direction is not resolved or detached from parent root view.

### Returns

| | |
|---|---|
| **`TextDirectionHeuristic`** (/reference/android/text/TextDirectionHeuristic) | This value will never be **`null`**. |

## getTextLocale

Added in API level 17 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public Locale (/reference/java/util/Locale) getTextLocale ()`

Get the default primary **`Locale`** (/reference/java/util/Locale) of the text in this TextView. This will always be the first member of **`getTextLocales()`** (/reference/android/widget/TextView#getTextLocales()).

### Returns

| | |
|---|---|
| **Locale** (/reference/java/util/Locale) | the default primary **Locale** (/reference/java/util/Locale) of the text in this TextView. This value will never be **null**. |

## getTextLocales

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public LocaleList (/reference/android/os/LocaleList) getTextLocales ()`

Get the default **LocaleList** (/reference/android/os/LocaleList) of the text in this TextView.

### Returns

| | |
|---|---|
| **LocaleList** (/reference/android/os/LocaleList) | the default **LocaleList** (/reference/android/os/LocaleList) of the text in this TextView. This value will never be **null**. |

## getTextMetricsParams

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public PrecomputedText.Params (/reference/android/text/PrecomputedText.Params) getTextMetricsParams ()`

Gets the parameters for text layout precomputation, for use with **PrecomputedText** (/reference/android/text/PrecomputedText).

### Returns

| | |
|---|---|
| **PrecomputedText.Params** (/reference/android/text/PrecomputedText.Params) | a current **PrecomputedText.Params** (/reference/android/text/PrecomputedText.Params) This value will never be **null**. |

### See also:

**PrecomputedText** (/reference/android/text/PrecomputedText)

## getTextScaleX

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public float getTextScaleX ()
```

Gets the extent by which text should be stretched horizontally. This will usually be 1.0.

### Returns

| | |
|---|---|
| **float** | The horizontal scale factor. |

## getTextSelectHandle

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public Drawable (/reference/android/graphics/drawable/Drawable) getTextSelectHandle ()
```

Returns the Drawable corresponding to the selection handle used for positioning the cursor within text. Note that any change applied to the handle Drawable will not be visible until the handle is hidden and then drawn again.

**Related XML Attributes:**

android:textSelectHandle (/reference/android/widget/TextView#attr_android:textSelectHandle)

**Returns**

| | |
|---|---|
| `Drawable` (/reference/android/graphics/drawable/Drawable) | the text select handle drawable This value may be **`null`**. |

**See also:**

`setTextSelectHandle(Drawable)` (/reference/android/widget/TextView#setTextSelectHandle(android.graphics.drawable.Drawable))

`setTextSelectHandle(int)` (/reference/android/widget/TextView#setTextSelectHandle(int))

## getTextSelectHandleLeft

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public Drawable (/reference/android/graphics/drawable/Drawable) getTextSelectHandleLeft ()
```

Returns the Drawable corresponding to the left handle used for selecting text. Note that any change applied to the handle Drawable will not be visible until the handle is hidden and then drawn again.

**Related XML Attributes:**

android:textSelectHandleLeft (/reference/android/widget/TextView#attr_android:textSelectHandleLeft)

**Returns**

| | |
|---|---|
| Drawable (/reference/android/graphics/drawable/Drawable) | the left text selection handle drawable This value may be **null**. |

**See also:**

setTextSelectHandleLeft(Drawable) (/reference/android/widget/TextView#setTextSelectHandleLeft(android.graphics.drawable.Drawable))

setTextSelectHandleLeft(int) (/reference/android/widget/TextView#setTextSelectHandleLeft(int))

## getTextSelectHandleRight

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public Drawable (/reference/android/graphics/drawable/Drawable) getTextSelectHandleRight ()
```

Returns the Drawable corresponding to the right handle used for selecting text. Note that any change applied to the handle Drawable will not be visible until the handle is hidden and then drawn again.

**Related XML Attributes:**

android:textSelectHandleRight (/reference/android/widget/TextView#attr_android:textSelectHandleRight)

### Returns

| | |
|---|---|
| `Drawable` (/reference/android/graphics/drawable/Drawable) | the right text selection handle drawable This value may be `null`. |

### See also:

`setTextSelectHandleRight(Drawable)` (/reference/android/widget/TextView#setTextSelectHandleRight(android.graphics.drawable.Drawable))

`setTextSelectHandleRight(int)` (/reference/android/widget/TextView#setTextSelectHandleRight(int))

## getTextSize

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public float getTextSize ()
```

### Returns

| | |
|---|---|
| `float` | the size (in pixels) of the default text size in this TextView. |

## getTotalPaddingBottom

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getTotalPaddingBottom ()
```

Returns the total bottom padding of the view, including the bottom Drawable if any, the extra space to keep more than maxLines from showing, and the vertical offset for gravity, if any.

### Returns

---

**`int`**

## getTotalPaddingEnd

Added in API level 17 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getTotalPaddingEnd ()
```

Returns the total end padding of the view, including the end Drawable if any.

### Returns

---

**`int`**

## getTotalPaddingLeft

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getTotalPaddingLeft ()
```

Returns the total left padding of the view, including the left Drawable if any.

### Returns

---

**int**

## getTotalPaddingRight

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getTotalPaddingRight ()
```

Returns the total right padding of the view, including the right Drawable if any.

### Returns

---

**int**

## getTotalPaddingStart

Added in API level 17 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getTotalPaddingStart ()
```

Returns the total start padding of the view, including the start Drawable if any.

### Returns

`int`

## getTotalPaddingTop

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int getTotalPaddingTop ()
```

Returns the total top padding of the view, including the top Drawable if any, the extra space to keep more than maxLines from showing, and the vertical offset for gravity, if any.

### Returns

`int`

## getTransformationMethod

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final TransformationMethod (/reference/android/text/method/TransformationMethod) getTransformationMethod ()
```

Gets the current **`TransformationMethod`** (/reference/android/text/method/TransformationMethod) for the TextView. This is frequently null, except for single-line and password fields.

**Related XML Attributes:**

android:password (/reference/android/widget/TextView#attr_android:password)

android:singleLine (/reference/android/widget/TextView#attr_android:singleLine)

**Returns**

| | |
|---|---|
| **`TransformationMethod`** (/reference/android/text/method/TransformationMethod) | the current transformation method for this TextView. |

## getTypeface

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public Typeface (/reference/android/graphics/Typeface) getTypeface ()
```

Gets the current **`Typeface`** (/reference/android/graphics/Typeface) that is used to style the text.

**Related XML Attributes:**

android:fontFamily (/reference/android/widget/TextView#attr_android:fontFamily)

android:typeface (/reference/android/widget/TextView#attr_android:typeface)

android:textStyle (/reference/android/widget/TextView#attr_android:textStyle)

### Returns

| | |
|---|---|
| Typeface (/reference/android/graphics/Typeface) | The current Typeface. |

### See also:

setTypeface(Typeface) (/reference/android/widget/TextView#setTypeface(android.graphics.Typeface))

## getUrls

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public URLSpan[] (/reference/android/text/style/URLSpan) getUrls ()
```

Returns the list of URLSpans (/reference/android/text/style/URLSpan) attached to the text (by Linkify (/reference/android/text/util/Linkify) or otherwise) if any. You can call URLSpan#getURL (/reference/android/text/style/URLSpan#getURL()) on them to find where they link to or use Spanned#getSpanStart (/reference/android/text/Spanned#getSpanStart(java.lang.Object)) and Spanned#getSpanEnd (/reference/android/text/Spanned#getSpanEnd(java.lang.Object)) to find the region of the text they are attached to.

### Returns

| |
|---|
| URLSpan[] |

(/reference/android/text/style/URLSpan)

## hasOverlappingRendering

Added in API level 16 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean hasOverlappingRendering ()
```

Returns whether this View has content which overlaps.

This function, intended to be overridden by specific View types, is an optimization when alpha is set on a view. If rendering overlaps in a view with alpha < 1, that view is drawn to an offscreen buffer and then composited into place, which can be expensive. If the view has no overlapping rendering, the view can draw each primitive with the appropriate alpha value directly. An example of overlapping rendering is a TextView with a background image, such as a Button. An example of non-overlapping rendering is a TextView with no background, or an ImageView with only the foreground image. The default implementation returns true; subclasses should override if they have cases which can be optimized.

**Note:** The return value of this method is ignored if **`forceHasOverlappingRendering(boolean)`** (/reference/android/view/View#forceHasOverlappingRendering(boolean)) has been called on this view.

### Returns

| | |
|---|---|
| **`boolean`** | true if the content in this view might overlap, false otherwise. |

## hasSelection

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean hasSelection ()
```

Return true iff there is a selection of nonzero length inside this text view.

### Returns

**boolean**

## invalidateDrawable

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void invalidateDrawable (Drawable (/reference/android/graphics/drawable/Drawable) drawable)
```

Invalidates the specified Drawable.

### Parameters

| | |
|---|---|
| **drawable** | **Drawable**: This value must never be **null**. |

## isAllCaps

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isAllCaps ()
```

Checks whether the transformation method applied to this TextView is set to ALL CAPS.

### Returns

| | |
|---|---|
| **boolean** | Whether the current transformation method is for ALL CAPS. |

### See also:

setAllCaps(boolean) (/reference/android/widget/TextView#setAllCaps(boolean))

setTransformationMethod(TransformationMethod) (/reference/android/widget/TextView#setTransformationMethod(android.text.method.TransformationMethod))

## isCursorVisible

Added in API level 16 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isCursorVisible ()
```

### Related XML Attributes:

android:cursorVisible (/reference/android/widget/TextView#attr_android:cursorVisible)

### Returns

| | |
|---|---|
| `boolean` | whether or not the cursor is visible (assuming this TextView is editable) |

### See also:

`setCursorVisible(boolean)` (/reference/android/widget/TextView#setCursorVisible(boolean))

## isElegantTextHeight

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isElegantTextHeight ()
```

Get the value of the TextView's elegant height metrics flag. This setting selects font variants that have not been compacted to fit Latin-based vertical metrics, and also increases top and bottom bounds to provide more space.

### Returns

| | |
|---|---|
| `boolean` | `true` if the elegant height metrics flag is set. |

### See also:

`setElegantTextHeight(boolean)` (/reference/android/widget/TextView#setElegantTextHeight(boolean))

`Paint.setElegantTextHeight(boolean)` (/reference/android/graphics/Paint#setElegantTextHeight(boolean))

## isFallbackLineSpacing

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isFallbackLineSpacing ()
```

**Related XML Attributes:**

android:fallbackLineSpacing (/reference/android/widget/TextView#attr_android:fallbackLineSpacing)

**Returns**

| | |
|---|---|
| **boolean** | whether fallback line spacing is enabled, **true** by default |

**See also:**

**setFallbackLineSpacing(boolean)** (/reference/android/widget/TextView#setFallbackLineSpacing(boolean))

## isHorizontallyScrollable

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final boolean isHorizontallyScrollable ()
```

Returns whether the text is allowed to be wider than the View. If false, the text will be wrapped to the width of the View.

**Related XML Attributes:**

android:scrollHorizontally (/reference/android/widget/TextView#attr_android:scrollHorizontally)

### Returns

**boolean**

### See also:

**setHorizontallyScrolling(boolean)** (/reference/android/widget/TextView#setHorizontallyScrolling(boolean))

## isInputMethodTarget

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isInputMethodTarget ()
```

Returns whether this text view is a current input method target. The default implementation just checks with **InputMethodManager** (/reference/android/view/inputmethod/InputMethodManager).

### Returns

| | |
|---|---|
| **boolean** | True if the TextView is a current input method target; false otherwise. |

## isSingleLine

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isSingleLine ()
```

Returns if the text is constrained to a single horizontally scrolling line ignoring new line characters instead of letting it wrap onto multiple lines.

**Related XML Attributes:**

android:singleLine (/reference/android/widget/TextView#attr_android:singleLine)

### Returns

---

**boolean**

## isSuggestionsEnabled

Added in API level 14 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isSuggestionsEnabled ()
```

Return whether or not suggestions are enabled on this TextView. The suggestions are generated by the IME or by the spell checker as the user types. This is done by adding SuggestionSpan (/reference/android/text/style/SuggestionSpan)s to the text. When suggestions are enabled (default), this list of suggestions will be displayed when the user asks for them on these parts of the text. This value depends on the inputType of this TextView. The class of the input type must be InputType#TYPE_CLASS_TEXT (/reference/android/text/InputType#TYPE_CLASS_TEXT). In addition, the type variation must be one of InputType#TYPE_TEXT_VARIATION_NORMAL (/reference/android/text/InputType#TYPE_TEXT_VARIATION_NORMAL), InputType#TYPE_TEXT_VARIATION_EMAIL_SUBJECT (/reference/android/text/InputType#TYPE_TEXT_VARIATION_EMAIL_SUBJECT), InputType#TYPE_TEXT_VARIATION_LONG_MESSAGE (/reference/android/text/InputType#TYPE_TEXT_VARIATION_LONG_MESSAGE), InputType#TYPE_TEXT_VARIATION_SHORT_MESSAGE (/reference/android/text/InputType#TYPE_TEXT_VARIATION_SHORT_MESSAGE) or InputType#TYPE_TEXT_VARIATION_WEB_EDIT_TEXT

(/reference/android/text/InputType#TYPE_TEXT_VARIATION_WEB_EDIT_TEXT). And finally, the `InputType#TYPE_TEXT_FLAG_NO_SUGGESTIONS` (/reference/android/text/InputType#TYPE_TEXT_FLAG_NO_SUGGESTIONS) flag must *not* be set.

### Returns

| | |
|---|---|
| **boolean** | true if the suggestions popup window is enabled, based on the inputType. |

## isTextSelectable

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean isTextSelectable ()
```

Returns the state of the **`textIsSelectable`** flag (See `setTextIsSelectable()` (/reference/android/widget/TextView#setTextIsSelectable(boolean))). Although you have to set this flag to allow users to select and copy text in a non-editable TextView, the content of an `EditText` (/reference/android/widget/EditText) can always be selected, independently of the value of this flag.

**Related XML Attributes:**

android:textIsSelectable (/reference/android/widget/TextView#attr_android:textIsSelectable)

### Returns

| | |
|---|---|
| **boolean** | True if the text displayed in this TextView can be selected by the user. |

## jumpDrawablesToCurrentState

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void jumpDrawablesToCurrentState ()
```

Call **`Drawable#jumpToCurrentState()`** (/reference/android/graphics/drawable/Drawable#jumpToCurrentState()) on all Drawable objects associated with this view.

Also calls **`StateListAnimator#jumpToCurrentState()`** (/reference/android/animation/StateListAnimator#jumpToCurrentState()) if there is a StateListAnimator attached to this view.
If you override this method you *must* call through to the superclass implementation.

## length

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public int length ()
```

Returns the length, in characters, of the text managed by this TextView

### Returns

| | |
|---|---|
| **`int`** | The length of the text managed by the TextView in characters. |

## moveCursorToVisibleOffset

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean moveCursorToVisibleOffset ()
```

Move the cursor, if needed, so that it is at an offset that is visible to the user. This will not move the cursor if it represents more than one character (a selection range). This will only work if the TextView contains spannable text; otherwise it will do nothing.

### Returns

| | |
|---|---|
| **boolean** | True if the cursor was actually moved, false otherwise. |

## onBeginBatchEdit

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void onBeginBatchEdit ()
```

Called by the framework in response to a request to begin a batch of edit operations through a call to link **`beginBatchEdit()`** (/reference/android/widget/TextView#beginBatchEdit()).

## onCheckIsTextEditor

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean onCheckIsTextEditor ()
```

Check whether the called view is a text editor, in which case it would make sense to automatically display a soft input window for it. Subclasses should override this if they

implement **`onCreateInputConnection(android.view.inputmethod.EditorInfo)`** (/reference/android/view/View#onCreateInputConnection(android.view.inputmethod.EditorInfo)) to return true if a call on that method would return a non-null InputConnection, and they are really a first-class editor that the user would normally start typing on when the go into a window containing your view.

The default implementation always returns false. This does *not* mean that its **`onCreateInputConnection(android.view.inputmethod.EditorInfo)`** (/reference/android/view/View#onCreateInputConnection(android.view.inputmethod.EditorInfo)) will not be called or the user can not otherwise perform edits on your view; it is just a hint to the system that this is not the primary purpose of this view.

### Returns

| | |
|---|---|
| **`boolean`** | Returns true if this view is a text editor, else false. |

## onCommitCompletion

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void onCommitCompletion (CompletionInfo (/reference/android/view/inputmethod/CompletionInfo) text)
```

Called by the framework in response to a text completion from the current input method, provided by it calling **`InputConnection#commitCompletion`** (/reference/android/view/inputmethod/InputConnection#commitCompletion(android.view.inputmethod.CompletionInfo)). The default implementation does nothing; text views that are supporting auto-completion should override this to do their desired behavior.

### Parameters

| | |
|---|---|
| **`text`** | **`CompletionInfo`**: The auto complete text the user has selected. |

## onCommitCorrection

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void onCommitCorrection (CorrectionInfo (/reference/android/view/inputmethod/CorrectionInfo) info)
```

Called by the framework in response to a text auto-correction (such as fixing a typo using a dictionary) from the current input method, provided by it calling **InputConnection#commitCorrection(CorrectionInfo)** (/reference/android/view/inputmethod/InputConnection#commitCorrection(android.view.inputmethod.CorrectionInfo)). The default implementation flashes the background of the corrected word to provide feedback to the user.

### Parameters

| | |
|---|---|
| **info** | **CorrectionInfo**: The auto correct info about the text that was corrected. |

## onCreateInputConnection

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public InputConnection (/reference/android/view/inputmethod/InputConnection) onCreateInputConnection (EditorInfo (/reference/android/view/inputmethod/EditorInfo) outAtt
```

Create a new InputConnection for an InputMethod to interact with the view. The default implementation returns null, since it doesn't support input methods. You can override this to implement such support. This is only needed for views that take focus and text input.

When implementing this, you probably also want to implement **onCheckIsTextEditor()** (/reference/android/view/View#onCheckIsTextEditor()) to indicate you will return a non-null InputConnection.

Also, take good care to fill in the **EditorInfo** (/reference/android/view/inputmethod/EditorInfo) object correctly and in its entirety, so that the connected IME can rely on its

values. For example, **EditorInfo.initialSelStart** (/reference/android/view/inputmethod/EditorInfo#initialSelStart) and **EditorInfo.initialSelEnd** (/reference/android/view/inputmethod/EditorInfo#initialSelEnd) members must be filled in with the correct cursor position for IMEs to work correctly with your application.

### Parameters

| | |
|---|---|
| **outAttrs** | **EditorInfo**: Fill in with attribute information about the connection. |

### Returns

**InputConnection** (/reference/android/view/inputmethod/InputConnection)

## onDragEvent

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean onDragEvent (DragEvent (/reference/android/view/DragEvent) event)
```

Handles drag events sent by the system following a call to **startDragAndDrop()** (/reference/android/view/View#startDragAndDrop(android.content.ClipData,%20android.view.View.DragShadowBuilder,%20java.lang.Object,%20int)).

When the system calls this method, it passes a **DragEvent** (/reference/android/view/DragEvent) object. A call to **DragEvent.getAction()** (/reference/android/view/DragEvent#getAction()) returns one of the action type constants defined in DragEvent. The method uses these to determine what is happening in the drag and drop operation.

### Parameters

| | |
|---|---|
| `event` | **`DragEvent`**: The **`DragEvent`** (/reference/android/view/DragEvent) sent by the system. The **`DragEvent.getAction()`** (/reference/android/view/DragEvent#getAction()) method returns an action type constant defined in DragEvent, indicating the type of drag event represented by this object. |

### Returns

| | |
|---|---|
| `boolean` | **`true`** if the method was successful, otherwise **`false`**.<br><br>The method should return **`true`** in response to an action type of **`DragEvent.ACTION_DRAG_STARTED`** (/reference/android/view/DragEvent#ACTION_DRAG_STARTED) to receive drag events for the current operation.<br><br>The method should also return **`true`** in response to an action type of **`DragEvent.ACTION_DROP`** (/reference/android/view/DragEvent#ACTION_DROP) if it consumed the drop, or **`false`** if it didn't.<br><br>For all other events, the return value is ignored. |

## onEditorAction

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void onEditorAction (int actionCode)
```

Called when an attached input method calls **`InputConnection#performEditorAction(int)`** (/reference/android/view/inputmethod/InputConnection#performEditorAction(int)) for this text view. The default implementation will call your action listener supplied to **`setOnEditorActionListener(TextView.OnEditorActionListener)`**

(/reference/android/widget/TextView#setOnEditorActionListener(android.widget.TextView.OnEditorActionListener)), or perform a standard operation for **`EditorInfo#IME_ACTION_NEXT`** (/reference/android/view/inputmethod/EditorInfo#IME_ACTION_NEXT), **`EditorInfo#IME_ACTION_PREVIOUS`** (/reference/android/view/inputmethod/EditorInfo#IME_ACTION_PREVIOUS), or **`EditorInfo#IME_ACTION_DONE`** (/reference/android/view/inputmethod/EditorInfo#IME_ACTION_DONE).

For backwards compatibility, if no IME options have been set and the text view would not normally advance focus on enter, then the NEXT and DONE actions received here will be turned into an enter key down/up pair to go through the normal key handling.

### Parameters

| | |
|---|---|
| **`actionCode`** | **`int`**: The code of the action being performed. |

### See also:

**`setOnEditorActionListener(TextView.OnEditorActionListener)`** (/reference/android/widget/TextView#setOnEditorActionListener(android.widget.TextView.OnEditorActionListener))

## onEndBatchEdit

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void onEndBatchEdit ()
```

Called by the framework in response to a request to end a batch of edit operations through a call to link **`endBatchEdit()`** (/reference/android/widget/TextView#endBatchEdit()).

## onGenericMotionEvent

Added in API level 12 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public boolean onGenericMotionEvent (MotionEvent (/reference/android/view/MotionEvent) event)`

Implement this method to handle generic motion events.

Generic motion events describe joystick movements, mouse hovers, track pad touches, scroll wheel movements and other input events. The **`MotionEvent#getSource()`** (/reference/android/view/MotionEvent#getSource()) of the motion event specifies the class of input that was received. Implementations of this method must examine the bits in the source before processing the event. The following code example shows how this is done.

Generic motion events with source class **`InputDevice#SOURCE_CLASS_POINTER`** (/reference/android/view/InputDevice#SOURCE_CLASS_POINTER) are delivered to the view under the pointer. All other generic motion events are delivered to the focused view.

```
public boolean onGenericMotionEvent(MotionEvent event) {
    if (event.isFromSource(InputDevice.SOURCE_CLASS_JOYSTICK)) {
        if (event.getAction() == MotionEvent.ACTION_MOVE) {
            // process the joystick movement...
            return true;
        }
    }
    if (event.isFromSource(InputDevice.SOURCE_CLASS_POINTER)) {
        switch (event.getAction()) {
            case MotionEvent.ACTION_HOVER_MOVE:
                // process the mouse hover movement...
                return true;
            case MotionEvent.ACTION_SCROLL:
                // process the scroll wheel movement...
                return true;
        }
    }
```

```
    return super.onGenericMotionEvent(event);
}
```

### Parameters

| | |
|---|---|
| **`event`** | **`MotionEvent`**: The generic motion event being processed. |

### Returns

| | |
|---|---|
| **`boolean`** | True if the event was handled, false otherwise. |

## onKeyDown

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean onKeyDown (int keyCode,
                KeyEvent (/reference/android/view/KeyEvent) event)
```

Default implementation of **`KeyEvent.Callback#onKeyDown(int, KeyEvent)`** (/reference/android/view/KeyEvent.Callback#onKeyDown(int,%20android.view.KeyEvent)): perform press of the view when **`KeyEvent#KEYCODE_DPAD_CENTER`** (/reference/android/view/KeyEvent#KEYCODE_DPAD_CENTER) or **`KeyEvent#KEYCODE_ENTER`** (/reference/android/view/KeyEvent#KEYCODE_ENTER) is released, if the view is enabled and clickable.

Key presses in software keyboards will generally NOT trigger this listener, although some may elect to do so in some situations. Do not rely on this to catch software key presses.

### Parameters

| | |
|---|---|
| `keyCode` | `int`: a key code that represents the button pressed, from `KeyEvent` (/reference/android/view/KeyEvent) |
| `event` | `KeyEvent`: the KeyEvent object that defines the button action |

### Returns

| | |
|---|---|
| `boolean` | If you handled the event, return true. If you want to allow the event to be handled by the next receiver, return false. |

## onKeyMultiple

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean onKeyMultiple (int keyCode,
                int repeatCount,
                KeyEvent (/reference/android/view/KeyEvent) event)
```

Default implementation of `KeyEvent.Callback#onKeyMultiple(int, int, KeyEvent)` (/reference/android/view/KeyEvent.Callback#onKeyMultiple(int,%20int,%20android.view.KeyEvent)): always returns false (doesn't handle the event).

Key presses in software keyboards will generally NOT trigger this listener, although some may elect to do so in some situations. Do not rely on this to catch software key presses.

### Parameters

| | |
|---|---|
| **keyCode** | **int**: A key code that represents the button pressed, from **KeyEvent** (/reference/android/view/KeyEvent). |
| **repeatCount** | **int**: The number of times the action was made. |
| **event** | **KeyEvent**: The KeyEvent object that defines the button action. |

### Returns

| | |
|---|---|
| **boolean** | If you handled the event, return true. If you want to allow the event to be handled by the next receiver, return false. |

## onKeyPreIme

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean onKeyPreIme (int keyCode,
                KeyEvent (/reference/android/view/KeyEvent) event)
```

Handle a key event before it is processed by any input method associated with the view hierarchy. This can be used to intercept key events in special situations before the IME consumes them; a typical example would be handling the BACK key to update the application's UI instead of allowing the IME to see it and close itself.

### Parameters

| | |
|---|---|
| **keyCode** | **int**: The value in event.getKeyCode(). |

| | |
|---|---|
| **event** | **KeyEvent**: Description of the key event. |

### Returns

| | |
|---|---|
| **boolean** | If you handled the event, return true. If you want to allow the event to be handled by the next receiver, return false. |

## onKeyShortcut

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean onKeyShortcut (int keyCode,
                KeyEvent (/reference/android/view/KeyEvent) event)
```

Called on the focused view when a key shortcut event is not handled. Override this method to implement local key shortcuts for the View. Key shortcuts can also be implemented by setting the **`MenuItem#setShortcut(char, char)`** (/reference/android/view/MenuItem#setShortcut(char,%20char)) property of menu items.

### Parameters

| | |
|---|---|
| **keyCode** | **int**: The value in event.getKeyCode(). |
| **event** | **KeyEvent**: Description of the key event. |

### Returns

| | |
|---|---|
| **boolean** | If you handled the event, return true. If you want to allow the event to be handled by the next receiver, return false. |

## onKeyUp

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean onKeyUp (int keyCode,
                KeyEvent (/reference/android/view/KeyEvent) event)
```

Default implementation of **KeyEvent.Callback#onKeyUp(int, KeyEvent)** (/reference/android/view/KeyEvent.Callback#onKeyUp(int,%20android.view.KeyEvent)): perform clicking of the view when **KeyEvent#KEYCODE_DPAD_CENTER** (/reference/android/view/KeyEvent#KEYCODE_DPAD_CENTER), **KeyEvent#KEYCODE_ENTER** (/reference/android/view/KeyEvent#KEYCODE_ENTER) or **KeyEvent#KEYCODE_SPACE** (/reference/android/view/KeyEvent#KEYCODE_SPACE) is released.

Key presses in software keyboards will generally NOT trigger this listener, although some may elect to do so in some situations. Do not rely on this to catch software key presses.

### Parameters

| | |
|---|---|
| **keyCode** | **int**: A key code that represents the button pressed, from **KeyEvent** (/reference/android/view/KeyEvent). |
| **event** | **KeyEvent**: The KeyEvent object that defines the button action. |

### Returns

| | |
|---|---|
| **boolean** | If you handled the event, return true. If you want to allow the event to be handled by the next receiver, return false. |

## onPreDraw

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean onPreDraw ()
```

Callback method to be invoked when the view tree is about to be drawn. At this point, all views in the tree have been measured and given a frame. Clients can use this to adjust their scroll bounds or even to request a new layout before drawing occurs.

### Returns

| | |
|---|---|
| **boolean** | Return true to proceed with the current drawing pass, or false to cancel. |

## onPrivateIMECommand

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean onPrivateIMECommand (String (/reference/java/lang/String) action,
                Bundle (/reference/android/os/Bundle) data)
```

Called by the framework in response to a private command from the current method, provided by it calling **`InputConnection#performPrivateCommand`** (/reference/android/view/inputmethod/InputConnection#performPrivateCommand(java.lang.String,%20android.os.Bundle)).

### Parameters

| | |
|---|---|
| **action** | **`String`**: The action name of the command. |

| | |
|---|---|
| **data** | **Bundle**: Any additional data for the command. This may be null. |

### Returns

| | |
|---|---|
| **boolean** | Return true if you handled the command, else false. |

## onResolvePointerIcon

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public PointerIcon (/reference/android/view/PointerIcon) onResolvePointerIcon (MotionEvent (/reference/android/view/MotionEvent) event,
                int pointerIndex)
```

Returns the pointer icon for the motion event, or null if it doesn't specify the icon. The default implementation does not care the location or event types, but some subclasses may use it (such as WebViews).

### Parameters

| | |
|---|---|
| **event** | **MotionEvent**: The MotionEvent from a mouse |
| **pointerIndex** | **int**: The index of the pointer for which to retrieve the **PointerIcon** (/reference/android/view/PointerIcon). This will be between 0 and **MotionEvent#getPointerCount()** (/reference/android/view/MotionEvent#getPointerCount()). |

### Returns

**PointerIcon**
(/reference/android/view/PointerIcon)

## onRestoreInstanceState

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void onRestoreInstanceState (Parcelable (/reference/android/os/Parcelable) state)
```

Hook allowing a view to re-apply a representation of its internal state that had previously been generated by **onSaveInstanceState()** (/reference/android/view/View#onSaveInstanceState()). This function will never be called with a null state.
If you override this method you *must* call through to the superclass implementation.

### Parameters

| | |
|---|---|
| **state** | **Parcelable**: The frozen state that had previously been returned by **onSaveInstanceState()** (/reference/android/widget/TextView#onSaveInstanceState()). |

## onRtlPropertiesChanged

Added in API level 17 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void onRtlPropertiesChanged (int layoutDirection)
```

Called when any RTL property (layout direction or text direction or text alignment) has been changed. Subclasses need to override this method to take care of cached information that depends on the resolved layout direction, or to inform child views that inherit their layout direction. The default implementation does nothing.

### Parameters

| | |
|---|---|
| **`layoutDirection`** | **`int`**: the direction of the layout Value is **`View.LAYOUT_DIRECTION_LTR`** (/reference/android/view/View#LAYOUT_DIRECTION_LTR), or **`View.LAYOUT_DIRECTION_RTL`** (/reference/android/view/View#LAYOUT_DIRECTION_RTL) |

## onSaveInstanceState

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public Parcelable (/reference/android/os/Parcelable) onSaveInstanceState ()`

Hook allowing a view to generate a representation of its internal state that can later be used to create a new instance with that same state. This state should only contain information that is not persistent or can not be reconstructed later. For example, you will never store your current position on screen because that will be computed again when a new instance of the view is placed in its view hierarchy.

Some examples of things you may store here: the current cursor position in a text view (but usually not the text itself since that is stored in a content provider or other persistent storage), the currently selected item in a list view.
If you override this method you *must* call through to the superclass implementation.

### Returns

| | |
|---|---|
| **`Parcelable`** | Returns a Parcelable object containing the view's current dynamic state, or null if there is nothing interesting to save. |

(/reference/android/os/Parcelable)

## onScreenStateChanged

Added in API level 16 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void onScreenStateChanged (int screenState)
```

This method is called whenever the state of the screen this view is attached to changes. A state change will usually occurs when the screen turns on or off (whether it happens automatically or the user does it manually.)

### Parameters

| | |
|---|---|
| **screenState** | **int**: The new state of the screen. Can be either **View.SCREEN_STATE_ON** (/reference/android/view/View#SCREEN_STATE_ON) or **View.SCREEN_STATE_OFF** (/reference/android/view/View#SCREEN_STATE_OFF) |

## onTextContextMenuItem

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean onTextContextMenuItem (int id)
```

Called when a context menu option for the text view is selected. Currently this will be one of **R.id.selectAll** (/reference/android/R.id#selectAll), **R.id.cut** (/reference/android/R.id#cut), **R.id.copy** (/reference/android/R.id#copy), **R.id.paste** (/reference/android/R.id#paste) or **R.id.shareText** (/reference/android/R.id#shareText).

### Parameters

| | |
|---|---|
| `id` | `int` |

### Returns

| | |
|---|---|
| `boolean` | true if the context menu item action was performed. |

## onTouchEvent

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean onTouchEvent (MotionEvent (/reference/android/view/MotionEvent) event)
```

Implement this method to handle touch screen motion events.

If this method is used to detect click actions, it is recommended that the actions be performed by implementing and calling **`performClick()`** (/reference/android/view/View#performClick()). This will ensure consistent system behavior, including:

- obeying click sound preferences
- dispatching OnClickListener calls
- handling **`AccessibilityNodeInfo#ACTION_CLICK`** (/reference/android/view/accessibility/AccessibilityNodeInfo#ACTION_CLICK) when accessibility features are enabled

### Parameters

| | |
|---|---|
| **event** | **MotionEvent**: The motion event. |

### Returns

| | |
|---|---|
| **boolean** | True if the event was handled, false otherwise. |

## onTrackballEvent

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean onTrackballEvent (MotionEvent (/reference/android/view/MotionEvent) event)
```

Implement this method to handle trackball motion events. The *relative* movement of the trackball since the last event can be retrieve with **MotionEvent#getX** (/reference/android/view/MotionEvent#getX()) and **MotionEvent#getY** (/reference/android/view/MotionEvent#getY()). These are normalized so that a movement of 1 corresponds to the user pressing one DPAD key (so they will often be fractional values, representing the more fine-grained movement information available from a trackball).

### Parameters

| | |
|---|---|
| **event** | **MotionEvent**: The motion event. |

### Returns

| | |
|---|---|
| **boolean** | True if the event was handled, false otherwise. |

## onWindowFocusChanged

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void onWindowFocusChanged (boolean hasWindowFocus)
```

Called when the window containing this view gains or loses focus. Note that this is separate from view focus: to receive key events, both your view and its window must have focus. If a window is displayed on top of yours that takes input focus, then your own window will lose focus but the view focus will remain unchanged.

### Parameters

| | |
|---|---|
| **`hasWindowFocus`** | **`boolean`**: True if the window containing this view now has focus, false otherwise. |

## performLongClick

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean performLongClick ()
```

Calls this view's OnLongClickListener, if it is defined. Invokes the context menu if the OnLongClickListener did not consume the event.

### Returns

| | |
|---|---|
| **`boolean`** | **`true`** if one of the above receivers consumed the event, **`false`** otherwise |

## removeTextChangedListener

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void removeTextChangedListener (TextWatcher (/reference/android/text/TextWatcher) watcher)
```

Removes the specified TextWatcher from the list of those whose methods are called whenever this TextView's text changes.

### Parameters

| | |
|---|---|
| `watcher` | `TextWatcher` |

## sendAccessibilityEventUnchecked

Added in API level 4 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void sendAccessibilityEventUnchecked (AccessibilityEvent (/reference/android/view/accessibility/AccessibilityEvent) event)
```

This method behaves exactly as `sendAccessibilityEvent(int)` (/reference/android/view/View#sendAccessibilityEvent(int)) but takes as an argument an empty `AccessibilityEvent` (/reference/android/view/accessibility/AccessibilityEvent) and does not perform a check whether accessibility is enabled.

If an `AccessibilityDelegate` (/reference/android/view/View.AccessibilityDelegate) has been specified via calling `setAccessibilityDelegate(android.view.View.AccessibilityDelegate)` (/reference/android/view/View#setAccessibilityDelegate(android.view.View.AccessibilityDelegate)) its `AccessibilityDelegate#sendAccessibilityEventUnchecked(View, AccessibilityEvent)` (/reference/android/view/View.AccessibilityDelegate#sendAccessibilityEventUnchecked(android.view.View,%20android.view.accessibility.AccessibilityEvent)) is responsible for handling this call.

### Parameters

| | |
|---|---|
| `event` | **`AccessibilityEvent`**: The event to send. |

## setAllCaps

Added in API level 14 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setAllCaps (boolean allCaps)
```

Sets the properties of this field to transform input to ALL CAPS display. This may use a "small caps" formatting if available. This setting will be ignored if this field is editable or selectable. This call replaces the current transformation method. Disabling this will not necessarily restore the previous behavior from before this was enabled.

**Related XML Attributes:**

android:textAllCaps (/reference/android/widget/TextView#attr_android:textAllCaps)

### Parameters

| | |
|---|---|
| `allCaps` | **`boolean`** |

**See also:**

`setTransformationMethod(TransformationMethod)` (/reference/android/widget/TextView#setTransformationMethod(android.text.method.TransformationMethod))

## setAutoLinkMask

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void setAutoLinkMask (int mask)
```

Sets the autolink mask of the text. See **`Linkify.ALL`** (/reference/android/text/util/Linkify#ALL) and peers for possible values.

**Note:** **`Linkify.MAP_ADDRESSES`** (/reference/android/text/util/Linkify#MAP_ADDRESSES) is deprecated and should be avoided; see its documentation.

### Related XML Attributes:

android:autoLink (/reference/android/widget/TextView#attr_android:autoLink)

### Parameters

| | |
|---|---|
| **`mask`** | **`int`** |

## setAutoSizeTextTypeUniformWithConfiguration

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setAutoSizeTextTypeUniformWithConfiguration (int autoSizeMinTextSize,
                int autoSizeMaxTextSize,
                int autoSizeStepGranularity,
                int unit)
```

Specify whether this widget should automatically scale the text to try to perfectly fit within the layout bounds. If all the configuration params are valid the type of auto-size is set to `AUTO_SIZE_TEXT_TYPE_UNIFORM` (/reference/android/widget/TextView#AUTO_SIZE_TEXT_TYPE_UNIFORM).

**Related XML Attributes:**

android:autoSizeTextType (/reference/android/widget/TextView#attr_android:autoSizeTextType)

android:autoSizeMinTextSize (/reference/android/widget/TextView#attr_android:autoSizeMinTextSize)

android:autoSizeMaxTextSize (/reference/android/widget/TextView#attr_android:autoSizeMaxTextSize)

android:autoSizeStepGranularity (/reference/android/widget/TextView#attr_android:autoSizeStepGranularity)

### Parameters

| | |
|---|---|
| `autoSizeMinTextSize` | `int`: the minimum text size available for auto-size |
| `autoSizeMaxTextSize` | `int`: the maximum text size available for auto-size |
| `autoSizeStepGranularity` | `int`: the auto-size step granularity. It is used in conjunction with the minimum and maximum text size in order to build the set of text sizes the system uses to choose from when auto-sizing |
| `unit` | `int`: the desired dimension unit for all sizes above. See `TypedValue` (/reference/android/util/TypedValue) for the possible dimension units |

### Throws

| | |
|---|---|
| `IllegalArgumentException` | if any of the configuration params are invalid. |

(/reference/java/lang/IllegalArgumentException)

**See also:**

`setAutoSizeTextTypeWithDefaults(int)` (/reference/android/widget/TextView#setAutoSizeTextTypeWithDefaults(int))

`setAutoSizeTextTypeUniformWithPresetSizes(int[], int)` (/reference/android/widget/TextView#setAutoSizeTextTypeUniformWithPresetSizes(int[],%20int))

`getAutoSizeMinTextSize()` (/reference/android/widget/TextView#getAutoSizeMinTextSize())

`getAutoSizeMaxTextSize()` (/reference/android/widget/TextView#getAutoSizeMaxTextSize())

`getAutoSizeStepGranularity()` (/reference/android/widget/TextView#getAutoSizeStepGranularity())

`getAutoSizeTextAvailableSizes()` (/reference/android/widget/TextView#getAutoSizeTextAvailableSizes())

## setAutoSizeTextTypeUniformWithPresetSizes

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setAutoSizeTextTypeUniformWithPresetSizes (int[] presetSizes,
                int unit)
```

Specify whether this widget should automatically scale the text to try to perfectly fit within the layout bounds. If at least one value from the `presetSizes` is valid then the type of auto-size is set to `AUTO_SIZE_TEXT_TYPE_UNIFORM` (/reference/android/widget/TextView#AUTO_SIZE_TEXT_TYPE_UNIFORM).

**Related XML Attributes:**

android:autoSizeTextType (/reference/android/widget/TextView#attr_android:autoSizeTextType)

android:autoSizePresetSizes (/reference/android/widget/TextView#attr_android:autoSizePresetSizes)

### Parameters

| | |
|---|---|
| `presetSizes` | `int`: an `int` array of sizes in pixels This value must never be `null`. |
| `unit` | `int`: the desired dimension unit for the preset sizes above. See `TypedValue` (/reference/android/util/TypedValue) for the possible dimension units |

### Throws

| | |
|---|---|
| `IllegalArgumentException` (/reference/java/lang/IllegalArgumentException) | if all of the `presetSizes` are invalid. |

### See also:

`setAutoSizeTextTypeWithDefaults(int)` (/reference/android/widget/TextView#setAutoSizeTextTypeWithDefaults(int))

`setAutoSizeTextTypeUniformWithConfiguration(int, int, int, int)` (/reference/android/widget/TextView#setAutoSizeTextTypeUniformWithConfiguration(int,%20int,%20int,%20int))

`getAutoSizeMinTextSize()` (/reference/android/widget/TextView#getAutoSizeMinTextSize())

`getAutoSizeMaxTextSize()` (/reference/android/widget/TextView#getAutoSizeMaxTextSize())

`getAutoSizeTextAvailableSizes()` (/reference/android/widget/TextView#getAutoSizeTextAvailableSizes())

# setAutoSizeTextTypeWithDefaults

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setAutoSizeTextTypeWithDefaults (int autoSizeTextType)
```

Specify whether this widget should automatically scale the text to try to perfectly fit within the layout bounds by using the default auto-size configuration.

**Related XML Attributes:**

android:autoSizeTextType (/reference/android/widget/TextView#attr_android:autoSizeTextType)

## Parameters

| | |
|---|---|
| **autoSizeTextType** | **int**: the type of auto-size. Must be one of **TextView#AUTO_SIZE_TEXT_TYPE_NONE** (/reference/android/widget/TextView#AUTO_SIZE_TEXT_TYPE_NONE) or **TextView#AUTO_SIZE_TEXT_TYPE_UNIFORM** (/reference/android/widget/TextView#AUTO_SIZE_TEXT_TYPE_UNIFORM) Value is **AUTO_SIZE_TEXT_TYPE_NONE** (/reference/android/widget/TextView#AUTO_SIZE_TEXT_TYPE_NONE), or **AUTO_SIZE_TEXT_TYPE_UNIFORM** (/reference/android/widget/TextView#AUTO_SIZE_TEXT_TYPE_UNIFORM) |

## Throws

| | |
|---|---|
| **IllegalArgumentException** (/reference/java/lang/IllegalArgumentException) | if **autoSizeTextType** is none of the types above. |

## See also:

**getAutoSizeTextType()** (/reference/android/widget/TextView#getAutoSizeTextType())

# setBreakStrategy

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setBreakStrategy (int breakStrategy)
```

Sets the break strategy for breaking paragraphs into lines. The default value for TextView is `Layout#BREAK_STRATEGY_HIGH_QUALITY` (/reference/android/text/Layout#BREAK_STRATEGY_HIGH_QUALITY), and the default value for EditText is `Layout#BREAK_STRATEGY_SIMPLE` (/reference/android/text/Layout#BREAK_STRATEGY_SIMPLE), the latter to avoid the text "dancing" when being edited.

Enabling hyphenation with either using `Layout#HYPHENATION_FREQUENCY_NORMAL` (/reference/android/text/Layout#HYPHENATION_FREQUENCY_NORMAL) or `Layout#HYPHENATION_FREQUENCY_FULL` (/reference/android/text/Layout#HYPHENATION_FREQUENCY_FULL) while line breaking is set to one of `Layout#BREAK_STRATEGY_BALANCED` (/reference/android/text/Layout#BREAK_STRATEGY_BALANCED), `Layout#BREAK_STRATEGY_HIGH_QUALITY` (/reference/android/text/Layout#BREAK_STRATEGY_HIGH_QUALITY) improves the structure of text layout however has performance impact and requires more time to do the text layout.

**Related XML Attributes:**

android:breakStrategy (/reference/android/widget/TextView#attr_android:breakStrategy)

**Parameters**

| | |
|---|---|
| `breakStrategy` | `int`: Value is `LineBreaker.BREAK_STRATEGY_SIMPLE` (/reference/android/graphics/text/LineBreaker#BREAK_STRATEGY_SIMPLE), `LineBreaker.BREAK_STRATEGY_HIGH_QUALITY` (/reference/android/graphics/text/LineBreaker#BREAK_STRATEGY_HIGH_QUALITY), or `LineBreaker.BREAK_STRATEGY_BALANCED` (/reference/android/graphics/text/LineBreaker#BREAK_STRATEGY_BALANCED) |

**See also:**

`getBreakStrategy()` (/reference/android/widget/TextView#getBreakStrategy())

`setHyphenationFrequency(int)` (/reference/android/widget/TextView#setHyphenationFrequency(int))

## setCompoundDrawablePadding

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setCompoundDrawablePadding (int pad)
```

Sets the size of the padding between the compound drawables and the text.

**Related XML Attributes:**

android:drawablePadding (/reference/android/widget/TextView#attr_android:drawablePadding)

**Parameters**

| | |
|---|---|
| `pad` | `int` |

## setCompoundDrawableTintBlendMode

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setCompoundDrawableTintBlendMode (BlendMode (/reference/android/graphics/BlendMode) blendMode)
```

Specifies the blending mode used to apply the tint specified by `setCompoundDrawableTintList(android.content.res.ColorStateList)`

(/reference/android/widget/TextView#setCompoundDrawableTintList(android.content.res.ColorStateList)) to the compound drawables. The default mode is **`PorterDuff.Mode#SRC_IN`** (/reference/android/graphics/PorterDuff.Mode#SRC_IN).

**Related XML Attributes:**

android:drawableTintMode (/reference/android/widget/TextView#attr_android:drawableTintMode)

### Parameters

| | |
|---|---|
| **`blendMode`** | **`BlendMode`**: the blending mode used to apply the tint, may be **`null`** to clear tint This value may be **`null`**. |

**See also:**

**`setCompoundDrawableTintList(ColorStateList)`** (/reference/android/widget/TextView#setCompoundDrawableTintList(android.content.res.ColorStateList))

**`Drawable.setTintBlendMode(BlendMode)`** (/reference/android/graphics/drawable/Drawable#setTintBlendMode(android.graphics.BlendMode))

## setCompoundDrawableTintList

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setCompoundDrawableTintList (ColorStateList (/reference/android/content/res/ColorStateList) tint)
```

Applies a tint to the compound drawables. Does not modify the current tint mode, which is **`PorterDuff.Mode#SRC_IN`** (/reference/android/graphics/PorterDuff.Mode#SRC_IN) by default.

Subsequent calls to **`setCompoundDrawables(android.graphics.drawable.Drawable, android.graphics.drawable.Drawable,`**

`android.graphics.drawable.Drawable, android.graphics.drawable.Drawable)`
(/reference/android/widget/TextView#setCompoundDrawables(android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable))
and related methods will automatically mutate the drawables and apply the specified tint and tint mode using `Drawable#setTintList(ColorStateList)`
(/reference/android/graphics/drawable/Drawable#setTintList(android.content.res.ColorStateList)).

**Related XML Attributes:**

android:drawableTint (/reference/android/widget/TextView#attr_android:drawableTint)

### Parameters

| | |
|---|---|
| `tint` | `ColorStateList`: the tint to apply, may be `null` to clear tint This value may be `null`. |

**See also:**

`getCompoundDrawableTintList()` (/reference/android/widget/TextView#getCompoundDrawableTintList())

`Drawable.setTintList(ColorStateList)` (/reference/android/graphics/drawable/Drawable#setTintList(android.content.res.ColorStateList))

## setCompoundDrawableTintMode

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setCompoundDrawableTintMode (PorterDuff.Mode (/reference/android/graphics/PorterDuff.Mode) tintMode)
```

Specifies the blending mode used to apply the tint specified by `setCompoundDrawableTintList(android.content.res.ColorStateList)`

(/reference/android/widget/TextView#setCompoundDrawableTintList(android.content.res.ColorStateList)) to the compound drawables. The default mode is `PorterDuff.Mode#SRC_IN` (/reference/android/graphics/PorterDuff.Mode#SRC_IN).

**Related XML Attributes:**

android:drawableTintMode (/reference/android/widget/TextView#attr_android:drawableTintMode)

### Parameters

| | |
|---|---|
| `tintMode` | `PorterDuff.Mode`: the blending mode used to apply the tint, may be `null` to clear tint This value may be `null`. |

**See also:**

`setCompoundDrawableTintList(ColorStateList)` (/reference/android/widget/TextView#setCompoundDrawableTintList(android.content.res.ColorStateList))

`Drawable.setTintMode(PorterDuff.Mode)` (/reference/android/graphics/drawable/Drawable#setTintMode(android.graphics.PorterDuff.Mode))

## setCompoundDrawables

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setCompoundDrawables (Drawable (/reference/android/graphics/drawable/Drawable) left,
                Drawable (/reference/android/graphics/drawable/Drawable) top,
                Drawable (/reference/android/graphics/drawable/Drawable) right,
                Drawable (/reference/android/graphics/drawable/Drawable) bottom)
```

Sets the Drawables (if any) to appear to the left of, above, to the right of, and below the text. Use `null` if you do not want a Drawable there. The Drawables must already

have had `Drawable#setBounds` (/reference/android/graphics/drawable/Drawable#setBounds(android.graphics.Rect)) called.

Calling this method will overwrite any Drawables previously set using `setCompoundDrawablesRelative(Drawable, Drawable, Drawable, Drawable)` (/reference/android/widget/TextView#setCompoundDrawablesRelative(android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable)) or related methods.

**Related XML Attributes:**

android:drawableLeft (/reference/android/widget/TextView#attr_android:drawableLeft)

android:drawableTop (/reference/android/widget/TextView#attr_android:drawableTop)

android:drawableRight (/reference/android/widget/TextView#attr_android:drawableRight)

android:drawableBottom (/reference/android/widget/TextView#attr_android:drawableBottom)

### Parameters

| | |
|---|---|
| `left` | `Drawable`: This value may be `null`. |
| `top` | `Drawable`: This value may be `null`. |
| `right` | `Drawable`: This value may be `null`. |
| `bottom` | `Drawable`: This value may be `null`. |

# setCompoundDrawablesRelative

Added in API level 17 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setCompoundDrawablesRelative (Drawable (/reference/android/graphics/drawable/Drawable) start,
                Drawable (/reference/android/graphics/drawable/Drawable) top,
                Drawable (/reference/android/graphics/drawable/Drawable) end,
                Drawable (/reference/android/graphics/drawable/Drawable) bottom)
```

Sets the Drawables (if any) to appear to the start of, above, to the end of, and below the text. Use `null` if you do not want a Drawable there. The Drawables must already have had `Drawable#setBounds` (/reference/android/graphics/drawable/Drawable#setBounds(android.graphics.Rect)) called.

Calling this method will overwrite any Drawables previously set using `setCompoundDrawables(Drawable, Drawable, Drawable, Drawable)` (/reference/android/widget/TextView#setCompoundDrawables(android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable)) or related methods.

**Related XML Attributes:**

android:drawableStart (/reference/android/widget/TextView#attr_android:drawableStart)

android:drawableTop (/reference/android/widget/TextView#attr_android:drawableTop)

android:drawableEnd (/reference/android/widget/TextView#attr_android:drawableEnd)

android:drawableBottom (/reference/android/widget/TextView#attr_android:drawableBottom)

### Parameters

| | |
|---|---|
| `start` | `Drawable`: This value may be `null`. |

| | |
|---|---|
| **top** | **Drawable**: This value may be **null**. |
| **end** | **Drawable**: This value may be **null**. |
| **bottom** | **Drawable**: This value may be **null**. |

## setCompoundDrawablesRelativeWithIntrinsicBounds

Added in API level 17 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setCompoundDrawablesRelativeWithIntrinsicBounds (Drawable (/reference/android/graphics/drawable/Drawable) start,
                Drawable (/reference/android/graphics/drawable/Drawable) top,
                Drawable (/reference/android/graphics/drawable/Drawable) end,
                Drawable (/reference/android/graphics/drawable/Drawable) bottom)
```

Sets the Drawables (if any) to appear to the start of, above, to the end of, and below the text. Use **null** if you do not want a Drawable there. The Drawables' bounds will be set to their intrinsic bounds.

Calling this method will overwrite any Drawables previously set using **setCompoundDrawables(Drawable, Drawable, Drawable, Drawable)** (/reference/android/widget/TextView#setCompoundDrawables(android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable))
or related methods.

**Related XML Attributes:**

android:drawableStart (/reference/android/widget/TextView#attr_android:drawableStart)

android:drawableTop (/reference/android/widget/TextView#attr_android:drawableTop)

android:drawableEnd (/reference/android/widget/TextView#attr_android:drawableEnd)

android:drawableBottom (/reference/android/widget/TextView#attr_android:drawableBottom)

### Parameters

| | |
|---|---|
| `start` | **`Drawable`**: This value may be **`null`**. |
| `top` | **`Drawable`**: This value may be **`null`**. |
| `end` | **`Drawable`**: This value may be **`null`**. |
| `bottom` | **`Drawable`**: This value may be **`null`**. |

## setCompoundDrawablesRelativeWithIntrinsicBounds

Added in API level 17 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setCompoundDrawablesRelativeWithIntrinsicBounds (int start,
                int top,
                int end,
                int bottom)
```

Sets the Drawables (if any) to appear to the start of, above, to the end of, and below the text. Use 0 if you do not want a Drawable there. The Drawables' bounds will be set to

their intrinsic bounds.

Calling this method will overwrite any Drawables previously set using `setCompoundDrawables(Drawable, Drawable, Drawable, Drawable)` (/reference/android/widget/TextView#setCompoundDrawables(android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable)) or related methods.

**Related XML Attributes:**

android:drawableStart (/reference/android/widget/TextView#attr_android:drawableStart)

android:drawableTop (/reference/android/widget/TextView#attr_android:drawableTop)

android:drawableEnd (/reference/android/widget/TextView#attr_android:drawableEnd)

android:drawableBottom (/reference/android/widget/TextView#attr_android:drawableBottom)

### Parameters

| | |
|---|---|
| `start` | `int`: Resource identifier of the start Drawable. |
| `top` | `int`: Resource identifier of the top Drawable. |
| `end` | `int`: Resource identifier of the end Drawable. |
| `bottom` | `int`: Resource identifier of the bottom Drawable. |

# setCompoundDrawablesWithIntrinsicBounds

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setCompoundDrawablesWithIntrinsicBounds (Drawable (/reference/android/graphics/drawable/Drawable) left,
                Drawable (/reference/android/graphics/drawable/Drawable) top,
                Drawable (/reference/android/graphics/drawable/Drawable) right,
                Drawable (/reference/android/graphics/drawable/Drawable) bottom)
```

Sets the Drawables (if any) to appear to the left of, above, to the right of, and below the text. Use **`null`** if you do not want a Drawable there. The Drawables' bounds will be set to their intrinsic bounds.

Calling this method will overwrite any Drawables previously set using **`setCompoundDrawablesRelative(Drawable, Drawable, Drawable, Drawable)`** (/reference/android/widget/TextView#setCompoundDrawablesRelative(android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable))
or related methods.

**Related XML Attributes:**

android:drawableLeft (/reference/android/widget/TextView#attr_android:drawableLeft)

android:drawableTop (/reference/android/widget/TextView#attr_android:drawableTop)

android:drawableRight (/reference/android/widget/TextView#attr_android:drawableRight)

android:drawableBottom (/reference/android/widget/TextView#attr_android:drawableBottom)

### Parameters

| | |
|---|---|
| **`left`** | **`Drawable`**: This value may be **`null`**. |

| | |
|---|---|
| **top** | **Drawable**: This value may be **null**. |
| **right** | **Drawable**: This value may be **null**. |
| **bottom** | **Drawable**: This value may be **null**. |

## setCompoundDrawablesWithIntrinsicBounds

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setCompoundDrawablesWithIntrinsicBounds (int left,
                int top,
                int right,
                int bottom)
```

Sets the Drawables (if any) to appear to the left of, above, to the right of, and below the text. Use 0 if you do not want a Drawable there. The Drawables' bounds will be set to their intrinsic bounds.

Calling this method will overwrite any Drawables previously set using **`setCompoundDrawablesRelative(Drawable, Drawable, Drawable, Drawable)`** (/reference/android/widget/TextView#setCompoundDrawablesRelative(android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable,%20android.graphics.drawable.Drawable))
or related methods.

**Related XML Attributes:**

android:drawableLeft (/reference/android/widget/TextView#attr_android:drawableLeft)

android:drawableTop (/reference/android/widget/TextView#attr_android:drawableTop)

android:drawableRight (/reference/android/widget/TextView#attr_android:drawableRight)

android:drawableBottom (/reference/android/widget/TextView#attr_android:drawableBottom)

### Parameters

| | |
|---|---|
| **`left`** | **`int`**: Resource identifier of the left Drawable. |
| **`top`** | **`int`**: Resource identifier of the top Drawable. |
| **`right`** | **`int`**: Resource identifier of the right Drawable. |
| **`bottom`** | **`int`**: Resource identifier of the bottom Drawable. |

## setCursorVisible

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setCursorVisible (boolean visible)
```

Set whether the cursor is visible. The default is true. Note that this property only makes sense for editable TextView.

**Related XML Attributes:**

android:cursorVisible (/reference/android/widget/TextView#attr_android:cursorVisible)

### Parameters

| | |
|---|---|
| `visible` | `boolean` |

### See also:

`isCursorVisible()` (/reference/android/widget/TextView#isCursorVisible())

## setCustomInsertionActionModeCallback

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setCustomInsertionActionModeCallback (ActionMode.Callback (/reference/android/view/ActionMode.Callback) actionModeCallback)
```

If provided, this ActionMode.Callback will be used to create the ActionMode when text insertion is initiated in this View. The standard implementation populates the menu with a subset of Select All, Paste and Replace actions, depending on what this View supports.

A custom implementation can add new entries in the default menu in its `ActionMode.Callback.onPrepareActionMode(android.view.ActionMode, android.view.Menu)` (/reference/android/view/ActionMode.Callback#onPrepareActionMode(android.view.ActionMode,%20android.view.Menu)) method. The default actions can also be removed from the menu using `Menu.removeItem(int)` (/reference/android/view/Menu#removeItem(int)) and passing `R.id.selectAll` (/reference/android/R.id#selectAll), `R.id.paste` (/reference/android/R.id#paste) or `R.id.replaceText` (/reference/android/R.id#replaceText) ids as parameters.

Returning false from `ActionMode.Callback.onCreateActionMode(android.view.ActionMode, android.view.Menu)` (/reference/android/view/ActionMode.Callback#onCreateActionMode(android.view.ActionMode,%20android.view.Menu)) will prevent the action mode from being started.

Action click events should be handled by the custom implementation of `ActionMode.Callback.onActionItemClicked(android.view.ActionMode, android.view.MenuItem)` (/reference/android/view/ActionMode.Callback#onActionItemClicked(android.view.ActionMode,%20android.view.MenuItem)).

Note that text insertion mode is not started when a TextView receives focus and the `R.attr.selectAllOnFocus` (/reference/android/R.attr#selectAllOnFocus) flag has been set.

### Parameters

| | |
|---|---|
| `actionModeCallback` | `ActionMode.Callback` |

## setCustomSelectionActionModeCallback

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setCustomSelectionActionModeCallback (ActionMode.Callback (/reference/android/view/ActionMode.Callback) actionModeCallback)
```

If provided, this ActionMode.Callback will be used to create the ActionMode when text selection is initiated in this View.

The standard implementation populates the menu with a subset of Select All, Cut, Copy, Paste, Replace and Share actions, depending on what this View supports.

A custom implementation can add new entries in the default menu in its `ActionMode.Callback.onPrepareActionMode(ActionMode, android.view.Menu)` (/reference/android/view/ActionMode.Callback#onPrepareActionMode(android.view.ActionMode,%20android.view.Menu)) method. The default actions can also be removed from the menu using `Menu.removeItem(int)` (/reference/android/view/Menu#removeItem(int)) and passing `R.id.selectAll` (/reference/android/R.id#selectAll), `R.id.cut` (/reference/android/R.id#cut), `R.id.copy` (/reference/android/R.id#copy), `R.id.paste` (/reference/android/R.id#paste), `R.id.replaceText` (/reference/android/R.id#replaceText) or `R.id.shareText` (/reference/android/R.id#shareText) ids as parameters.

Returning false from `ActionMode.Callback.onCreateActionMode(ActionMode, android.view.Menu)`

(/reference/android/view/ActionMode.Callback#onCreateActionMode(android.view.ActionMode,%20android.view.Menu)) will prevent the action mode from being started.

Action click events should be handled by the custom implementation of **`ActionMode.Callback.onActionItemClicked(ActionMode, android.view.MenuItem)`** (/reference/android/view/ActionMode.Callback#onActionItemClicked(android.view.ActionMode,%20android.view.MenuItem)).

Note that text selection mode is not started when a TextView receives focus and the **`R.attr.selectAllOnFocus`** (/reference/android/R.attr#selectAllOnFocus) flag has been set. The content is highlighted in that case, to allow for quick replacement.

### Parameters

| | |
|---|---|
| **`actionModeCallback`** | **`ActionMode.Callback`** |

## setEditableFactory

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void setEditableFactory (Editable.Factory (/reference/android/text/Editable.Factory) factory)
```

Sets the Factory used to create new **`Editable`** (/reference/android/text/Editable).

### Parameters

| | |
|---|---|
| **`factory`** | **`Editable.Factory`**: **`Editable.Factory`** (/reference/android/text/Editable.Factory) to be used |

### See also:

`Editable.Factory` (/reference/android/text/Editable.Factory)

`TextView.BufferType.EDITABLE` (/reference/android/widget/TextView.BufferType#EDITABLE)

## setElegantTextHeight

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setElegantTextHeight (boolean elegant)
```

Set the TextView's elegant height metrics flag. This setting selects font variants that have not been compacted to fit Latin-based vertical metrics, and also increases top and bottom bounds to provide more space.

**Related XML Attributes:**

android:elegantTextHeight (/reference/android/widget/TextView#attr_android:elegantTextHeight)

### Parameters

| | |
|---|---|
| `elegant` | **`boolean`**: set the paint's elegant metrics flag. |

### See also:

`isElegantTextHeight()` (/reference/android/widget/TextView#isElegantTextHeight())

`Paint.isElegantTextHeight()` (/reference/android/graphics/Paint#isElegantTextHeight())

## setEllipsize

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setEllipsize (TextUtils.TruncateAt (/reference/android/text/TextUtils.TruncateAt) where)
```

Causes words in the text that are longer than the view's width to be ellipsized instead of broken in the middle. You may also want to **`setSingleLine()`** (/reference/android/widget/TextView#setSingleLine()) or **`setHorizontallyScrolling(boolean)`** (/reference/android/widget/TextView#setHorizontallyScrolling(boolean)) to constrain the text to a single line. Use **`null`** to turn off ellipsizing. If **`setMaxLines(int)`** (/reference/android/widget/TextView#setMaxLines(int)) has been used to set two or more lines, only **`TextUtils.TruncateAt.END`** (/reference/android/text/TextUtils.TruncateAt#END) and **`TextUtils.TruncateAt.MARQUEE`** (/reference/android/text/TextUtils.TruncateAt#MARQUEE) are supported (other ellipsizing types will not do anything).

**Related XML Attributes:**

android:ellipsize (/reference/android/widget/TextView#attr_android:ellipsize)

### Parameters

| | |
|---|---|
| **`where`** | **`TextUtils.TruncateAt`** |

## setEms

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setEms (int ems)
```

Sets the width of the TextView to be exactly **`ems`** wide. This value is used for width calculation if LayoutParams does not force TextView to have an exact width. Setting this

value overrides previous minimum/maximum configurations such as **`setMinEms(int)`** (/reference/android/widget/TextView#setMinEms(int)) or **`setMaxEms(int)`** (/reference/android/widget/TextView#setMaxEms(int)).

**Related XML Attributes:**

android:ems (/reference/android/widget/TextView#attr_android:ems)

### Parameters

| | |
|---|---|
| **`ems`** | **`int`**: the exact width of the TextView in terms of ems |

**See also:**

**`setWidth(int)`** (/reference/android/widget/TextView#setWidth(int))

## setEnabled

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setEnabled (boolean enabled)
```

Set the enabled state of this view. The interpretation of the enabled state varies by subclass.

### Parameters

| | |
|---|---|
| **`enabled`** | **`boolean`**: True if this view is enabled, false otherwise. |

## setError

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setError (CharSequence (/reference/java/lang/CharSequence) error)
```

Sets the right-hand compound drawable of the TextView to the "error" icon and sets an error message that will be displayed in a popup when the TextView has focus. The icon and error message will be reset to null when any key events cause changes to the TextView's text. If the **`error`** is **`null`**, the error message and icon will be cleared.

### Parameters

| | |
|---|---|
| **`error`** | **`CharSequence`** |

## setError

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setError (CharSequence (/reference/java/lang/CharSequence) error,
                Drawable (/reference/android/graphics/drawable/Drawable) icon)
```

Sets the right-hand compound drawable of the TextView to the specified icon and sets an error message that will be displayed in a popup when the TextView has focus. The icon and error message will be reset to null when any key events cause changes to the TextView's text. The drawable must already have had **`Drawable#setBounds`** (/reference/android/graphics/drawable/Drawable#setBounds(android.graphics.Rect)) set on it. If the **`error`** is **`null`**, the error message will be cleared (and you should provide a **`null`** icon as well).

### Parameters

| | |
|---|---|
| `error` | `CharSequence` |
| `icon` | `Drawable` |

## setExtractedText

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public void setExtractedText (ExtractedText (/reference/android/view/inputmethod/ExtractedText) text)`

Apply to this text view the given extracted text, as previously returned by **`extractText(android.view.inputmethod.ExtractedTextRequest, android.view.inputmethod.ExtractedText)`** (/reference/android/widget/TextView#extractText(android.view.inputmethod.ExtractedTextRequest,%20android.view.inputmethod.ExtractedText)).

### Parameters

| | |
|---|---|
| `text` | `ExtractedText` |

## setFallbackLineSpacing

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setFallbackLineSpacing (boolean enabled)
```

Set whether to respect the ascent and descent of the fallback fonts that are used in displaying the text (which is needed to avoid text from consecutive lines running into each other). If set, fallback fonts that end up getting used can increase the ascent and descent of the lines that they are used on.

It is required to be true if text could be in languages like Burmese or Tibetan where text is typically much taller or deeper than Latin text.

**Related XML Attributes:**

android:fallbackLineSpacing (/reference/android/widget/TextView#attr_android:fallbackLineSpacing)

### Parameters

| | |
|---|---|
| **`enabled`** | **`boolean`**: whether to expand linespacing based on fallback fonts, **`true`** by default |

**See also:**

**`StaticLayout.Builder.setUseLineSpacingFromFallbacks(boolean)`** (/reference/android/text/StaticLayout.Builder#setUseLineSpacingFromFallbacks(boolean))

## setFilters

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setFilters (InputFilter[] (/reference/android/text/InputFilter) filters)
```

Sets the list of input filters that will be used if the buffer is Editable. Has no effect otherwise.

**Related XML Attributes:**

android:maxLength (/reference/android/widget/TextView#attr_android:maxLength)

### Parameters

| | |
|---|---|
| `filters` | `InputFilter` |

## setFirstBaselineToTopHeight

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setFirstBaselineToTopHeight (int firstBaselineToTopHeight)
```

Updates the top padding of the TextView so that **`firstBaselineToTopHeight`** is the distance between the top of the TextView and first line's baseline.


First and last baseline metrics for a TextView.

**Note** that if `FontMetrics.top` or `FontMetrics.ascent` was already greater than `firstBaselineToTopHeight`, the top padding is not updated. Moreover since this function sets the top padding, if the height of the TextView is less than the sum of top padding, line height and bottom padding, top of the line will be pushed down and bottom will be clipped.

**Related XML Attributes:**

android:firstBaselineToTopHeight (/reference/android/widget/TextView#attr_android:firstBaselineToTopHeight)

### Parameters

`firstBaselineToTopHeight` `int`: distance between first baseline to top of the container in pixels This units of this value are pixels. Value is 0 or greater

### See also:

`getFirstBaselineToTopHeight()` (/reference/android/widget/TextView#getFirstBaselineToTopHeight())

`setLastBaselineToBottomHeight(int)` (/reference/android/widget/TextView#setLastBaselineToBottomHeight(int))

`setPadding(int, int, int, int)` (/reference/android/widget/TextView#setPadding(int,%20int,%20int,%20int))

`setPaddingRelative(int, int, int, int)` (/reference/android/widget/TextView#setPaddingRelative(int,%20int,%20int,%20int))

## setFontFeatureSettings

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setFontFeatureSettings (String (/reference/java/lang/String) fontFeatureSettings)
```

Sets font feature settings. The format is the same as the CSS font-feature-settings attribute: https://www.w3.org/TR/css-fonts-3/#font-feature-settings-prop (https://www.w3.org/TR/css-fonts-3/#font-feature-settings-prop)

**Related XML Attributes:**

android:fontFeatureSettings (/reference/android/widget/TextView#attr_android:fontFeatureSettings)

### Parameters

| | |
|---|---|
| `fontFeatureSettings` | `String`: font feature settings represented as CSS compatible string This value may be `null`. |

### See also:

`getFontFeatureSettings()` (/reference/android/widget/TextView#getFontFeatureSettings())

`Paint.getFontFeatureSettings()` (/reference/android/graphics/Paint#getFontFeatureSettings())

## setFontVariationSettings

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean setFontVariationSettings (String (/reference/java/lang/String) fontVariationSettings)
```

Sets TrueType or OpenType font variation settings. The settings string is constructed from multiple pairs of axis tag and style values. The axis tag must contain four ASCII characters and must be wrapped with single quotes (U+0027) or double quotes (U+0022). Axis strings that are longer or shorter than four characters, or contain characters outside of U+0020..U+007E are invalid. If a specified axis name is not defined in the font, the settings will be ignored.

Examples,

- Set font width to 150.

```
TextView textView = (TextView) findViewById(R.id.textView);
textView.setFontVariationSettings("'wdth' 150");
```

- Set the font slant to 20 degrees and ask for italic style.

```
TextView textView = (TextView) findViewById(R.id.textView);
```

```
textView.setFontVariationSettings("'slnt' 20, 'ital' 1");
```

### Related XML Attributes:

android:fontVariationSettings (/reference/android/widget/TextView#attr_android:fontVariationSettings)

### Parameters

| | |
|---|---|
| `fontVariationSettings` | `String`: font variation settings. You can pass null or empty string as no variation settings. This value may be `null`. |

### Returns

| | |
|---|---|
| `boolean` | true if the given settings is effective to at least one font file underlying this TextView. This function also returns true for empty settings string. Otherwise returns false. |

### Throws

| | |
|---|---|
| `IllegalArgumentException` (/reference/java/lang/IllegalArgumentException) | If given string is not a valid font variation settings format. |

### See also:

`getFontVariationSettings()` (/reference/android/widget/TextView#getFontVariationSettings())

**FontVariationAxis** (/reference/android/graphics/fonts/FontVariationAxis)

## setFreezesText

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setFreezesText (boolean freezesText)
```

Control whether this text view saves its entire text contents when freezing to an icicle, in addition to dynamic state such as cursor position. By default this is false, not saving the text. Set to true if the text in the text view is not being saved somewhere else in persistent storage (such as in a content provider) so that if the view is later thawed the user will not lose their data. For **EditText** (/reference/android/widget/EditText) it is always enabled, regardless of the value of the attribute.

**Related XML Attributes:**

android:freezesText (/reference/android/widget/TextView#attr_android:freezesText)

### Parameters

| | |
|---|---|
| `freezesText` | **boolean**: Controls whether a frozen icicle should include the entire text data: true to include it, false to not. |

## setGravity

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setGravity (int gravity)
```

Sets the horizontal alignment of the text and the vertical gravity that will be used when there is extra space in the TextView beyond what is required for the text itself.

**Related XML Attributes:**

android:gravity (/reference/android/widget/TextView#attr_android:gravity)

### Parameters

| | |
|---|---|
| `gravity` | `int` |

**See also:**

`Gravity` (/reference/android/view/Gravity)

## setHeight

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setHeight (int pixels)
```

Sets the height of the TextView to be exactly `pixels` tall.

This value is used for height calculation if LayoutParams does not force TextView to have an exact height. Setting this value overrides previous minimum/maximum height configurations such as `setMinHeight(int)` (/reference/android/widget/TextView#setMinHeight(int)) or `setMaxHeight(int)` (/reference/android/widget/TextView#setMaxHeight(int)).

**Related XML Attributes:**

android:height (/reference/android/widget/TextView#attr_android:height)

### Parameters

| | |
|---|---|
| `pixels` | `int`: the exact height of the TextView in terms of pixels |

### See also:

`setLines(int)` (/reference/android/widget/TextView#setLines(int))

## setHighlightColor

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setHighlightColor (int color)
```

Sets the color used to display the selection highlight.

### Related XML Attributes:

android:textColorHighlight (/reference/android/widget/TextView#attr_android:textColorHighlight)

### Parameters

| | |
|---|---|
| `color` | `int` |

## setHint

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void setHint (CharSequence (/reference/java/lang/CharSequence) hint)
```

Sets the text to be displayed when the text of the TextView is empty. Null means to use the normal empty text. The hint does not currently participate in determining the size of the view.

**Related XML Attributes:**

android:hint (/reference/android/widget/TextView#attr_android:hint)

### Parameters

| | |
|---|---|
| `hint` | `CharSequence` |

## setHint

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void setHint (int resid)
```

Sets the text to be displayed when the text of the TextView is empty, from a resource.

**Related XML Attributes:**

android:hint (/reference/android/widget/TextView#attr_android:hint)

### Parameters

| | |
|---|---|
| `resid` | `int` |

## setHintTextColor

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public final void setHintTextColor (ColorStateList (/reference/android/content/res/ColorStateList) colors)`

Sets the color of the hint text.

**Related XML Attributes:**

android:textColorHint (/reference/android/widget/TextView#attr_android:textColorHint)

### Parameters

| | |
|---|---|
| `colors` | `ColorStateList` |

**See also:**

`getHintTextColors()` (/reference/android/widget/TextView#getHintTextColors())

`setHintTextColor(int)` (/reference/android/widget/TextView#setHintTextColor(int))

`setTextColor(ColorStateList)` (/reference/android/widget/TextView#setTextColor(android.content.res.ColorStateList))

`setLinkTextColor(ColorStateList)` (/reference/android/widget/TextView#setLinkTextColor(android.content.res.ColorStateList))

## setHintTextColor

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void setHintTextColor (int color)
```

Sets the color of the hint text for all the states (disabled, focussed, selected...) of this TextView.

**Related XML Attributes:**

android:textColorHint (/reference/android/widget/TextView#attr_android:textColorHint)

### Parameters

| | |
|---|---|
| `color` | `int` |

### See also:

`setHintTextColor(ColorStateList)` (/reference/android/widget/TextView#setHintTextColor(android.content.res.ColorStateList))

`getHintTextColors()` (/reference/android/widget/TextView#getHintTextColors())

`setTextColor(int)` (/reference/android/widget/TextView#setTextColor(int))

## setHorizontallyScrolling

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setHorizontallyScrolling (boolean whether)
```

Sets whether the text should be allowed to be wider than the View is. If false, it will be wrapped to the width of the View.

**Related XML Attributes:**

android:scrollHorizontally (/reference/android/widget/TextView#attr_android:scrollHorizontally)

**Parameters**

| | |
|---|---|
| **whether** | **boolean** |

## setHyphenationFrequency

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setHyphenationFrequency (int hyphenationFrequency)
```

Sets the frequency of automatic hyphenation to use when determining word breaks. The default value for both TextView and **`EditText`** (/reference/android/widget/EditText) is **`Layout#HYPHENATION_FREQUENCY_NONE`** (/reference/android/text/Layout#HYPHENATION_FREQUENCY_NONE). Note that the default hyphenation frequency value is set from the theme.

Enabling hyphenation with either using **`Layout#HYPHENATION_FREQUENCY_NORMAL`** (/reference/android/text/Layout#HYPHENATION_FREQUENCY_NORMAL) or

`Layout#HYPHENATION_FREQUENCY_FULL` (/reference/android/text/Layout#HYPHENATION_FREQUENCY_FULL) while line breaking is set to one of `Layout#BREAK_STRATEGY_BALANCED` (/reference/android/text/Layout#BREAK_STRATEGY_BALANCED), `Layout#BREAK_STRATEGY_HIGH_QUALITY` (/reference/android/text/Layout#BREAK_STRATEGY_HIGH_QUALITY) improves the structure of text layout however has performance impact and requires more time to do the text layout.

Note: Before Android Q, in the theme hyphenation frequency is set to `Layout#HYPHENATION_FREQUENCY_NORMAL` (/reference/android/text/Layout#HYPHENATION_FREQUENCY_NORMAL). The default value is changed into `Layout#HYPHENATION_FREQUENCY_NONE` (/reference/android/text/Layout#HYPHENATION_FREQUENCY_NONE) on Q.

**Related XML Attributes:**

android:hyphenationFrequency (/reference/android/widget/TextView#attr_android:hyphenationFrequency)

### Parameters

| | |
|---|---|
| `hyphenationFrequency` | `int`: the hyphenation frequency to use, one of `Layout#HYPHENATION_FREQUENCY_NONE` (/reference/android/text/Layout#HYPHENATION_FREQUENCY_NONE), `Layout#HYPHENATION_FREQUENCY_NORMAL` (/reference/android/text/Layout#HYPHENATION_FREQUENCY_NORMAL), `Layout#HYPHENATION_FREQUENCY_FULL` (/reference/android/text/Layout#HYPHENATION_FREQUENCY_FULL) Value is `Layout.HYPHENATION_FREQUENCY_NORMAL` (/reference/android/text/Layout#HYPHENATION_FREQUENCY_NORMAL), `Layout.HYPHENATION_FREQUENCY_FULL` (/reference/android/text/Layout#HYPHENATION_FREQUENCY_FULL), or `Layout.HYPHENATION_FREQUENCY_NONE` (/reference/android/text/Layout#HYPHENATION_FREQUENCY_NONE) |

**See also:**

`getHyphenationFrequency()` (/reference/android/widget/TextView#getHyphenationFrequency())

`getBreakStrategy()` (/reference/android/widget/TextView#getBreakStrategy())

# setImeActionLabel

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setImeActionLabel (CharSequence (/reference/java/lang/CharSequence) label,
                int actionId)
```

Change the custom IME action associated with the text view, which will be reported to an IME with **`EditorInfo#actionLabel`** (/reference/android/view/inputmethod/EditorInfo#actionLabel) and **`EditorInfo#actionId`** (/reference/android/view/inputmethod/EditorInfo#actionId) when it has focus.

**Related XML Attributes:**

android:imeActionLabel (/reference/android/widget/TextView#attr_android:imeActionLabel)

android:imeActionId (/reference/android/widget/TextView#attr_android:imeActionId)

## Parameters

| | |
|---|---|
| `label` | `CharSequence` |
| `actionId` | `int` |

## See also:

**`getImeActionLabel()`** (/reference/android/widget/TextView#getImeActionLabel())

**`getImeActionId()`** (/reference/android/widget/TextView#getImeActionId())

**`EditorInfo`** (/reference/android/view/inputmethod/EditorInfo)

## setImeHintLocales

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setImeHintLocales (LocaleList (/reference/android/os/LocaleList) hintLocales)
```

Change "hint" locales associated with the text view, which will be reported to an IME with **`EditorInfo#hintLocales`** (/reference/android/view/inputmethod/EditorInfo#hintLocales) when it has focus. Starting with Android O, this also causes internationalized listeners to be created (or change locale) based on the first locale in the input locale list.

**Note:** If you want new "hint" to take effect immediately you need to call **`InputMethodManager#restartInput(View)`** (/reference/android/view/inputmethod/InputMethodManager#restartInput(android.view.View)).

### Parameters

| | |
|---|---|
| **`hintLocales`** | **`LocaleList`**: List of the languages that the user is supposed to switch to no matter what input method subtype is currently used. Set **`null`** to clear the current "hint". This value may be **`null`**. |

### See also:

**`getImeHintLocales()`** (/reference/android/widget/TextView#getImeHintLocales())

**`EditorInfo.hintLocales`** (/reference/android/view/inputmethod/EditorInfo#hintLocales)

## setImeOptions

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setImeOptions (int imeOptions)
```

Change the editor type integer associated with the text view, which is reported to an Input Method Editor (IME) with `EditorInfo#imeOptions` (/reference/android/view/inputmethod/EditorInfo#imeOptions) when it has focus.

**Related XML Attributes:**

android:imeOptions (/reference/android/widget/TextView#attr_android:imeOptions)

### Parameters

| | |
|---|---|
| `imeOptions` | `int` |

**See also:**

`getImeOptions()` (/reference/android/widget/TextView#getImeOptions())

`EditorInfo` (/reference/android/view/inputmethod/EditorInfo)

## setIncludeFontPadding

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setIncludeFontPadding (boolean includepad)
```

Set whether the TextView includes extra top and bottom padding to make room for accents that go above the normal ascent and descent. The default is true.

**Related XML Attributes:**

android:includeFontPadding (/reference/android/widget/TextView#attr_android:includeFontPadding)

### Parameters

| | |
|---|---|
| `includepad` | `boolean` |

**See also:**

`getIncludeFontPadding()` (/reference/android/widget/TextView#getIncludeFontPadding())

## setInputExtras

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setInputExtras (int xmlResId)
```

Set the extra input data of the text, which is the `EditorInfo#extras` (/reference/android/view/inputmethod/EditorInfo#extras) Bundle that will be filled in when creating an input connection. The given integer is the resource identifier of an XML resource holding an `<input-extras>` (/reference/android/R.styleable#InputExtras) XML tree.

**Related XML Attributes:**

android:editorExtras (/reference/android/widget/TextView#attr_android:editorExtras)

### Parameters

| | |
|---|---|
| `xmlResId` | `int` |

### Throws

`IOException`
(/reference/java/io/IOException)

`XmlPullParserException`
(/reference/org/xmlpull/v1/XmlPullParserException)

### See also:

`getInputExtras(boolean)` (/reference/android/widget/TextView#getInputExtras(boolean))

`EditorInfo.extras` (/reference/android/view/inputmethod/EditorInfo#extras)

## setInputType

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setInputType (int type)
```

Set the type of the content with a constant as defined for `EditorInfo#inputType` (/reference/android/view/inputmethod/EditorInfo#inputType). This will take care of changing the key listener, by calling `setKeyListener(android.text.method.KeyListener)` (/reference/android/widget/TextView#setKeyListener(android.text.method.KeyListener)), to match the given content type. If the given content type is `EditorInfo#TYPE_NULL` (/reference/android/text/InputType#TYPE_NULL) then a soft keyboard will not be displayed for this text view. Note that the maximum number of displayed lines (see `setMaxLines(int)` (/reference/android/widget/TextView#setMaxLines(int))) will be modified if you change the `EditorInfo#TYPE_TEXT_FLAG_MULTI_LINE` (/reference/android/text/InputType#TYPE_TEXT_FLAG_MULTI_LINE) flag of the input type.

### Related XML Attributes:

android:inputType (/reference/android/widget/TextView#attr_android:inputType)

### Parameters

| | |
|---|---|
| `type` | `int` |

### See also:

`getInputType()` (/reference/android/widget/TextView#getInputType())

`setRawInputType(int)` (/reference/android/widget/TextView#setRawInputType(int))

`InputType` (/reference/android/text/InputType)

## setJustificationMode

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setJustificationMode (int justificationMode)
```

Set justification mode. The default value is `Layout#JUSTIFICATION_MODE_NONE` (/reference/android/text/Layout#JUSTIFICATION_MODE_NONE). If the last line is too short for justification, the last line will be displayed with the alignment set by `View.setTextAlignment(int)` (/reference/android/view/View#setTextAlignment(int)).

### Parameters

| | |
|---|---|
| `justificationMode` | `int`: Value is `LineBreaker.JUSTIFICATION_MODE_NONE` (/reference/android/graphics/text/LineBreaker#JUSTIFICATION_MODE_NONE), or `LineBreaker.JUSTIFICATION_MODE_INTER_WORD` (/reference/android/graphics/text/LineBreaker#JUSTIFICATION_MODE_INTER_WORD) |

### Returns

| | |
|---|---|
| `void` | Value is `LineBreaker.JUSTIFICATION_MODE_NONE` (/reference/android/graphics/text/LineBreaker#JUSTIFICATION_MODE_NONE), or `LineBreaker.JUSTIFICATION_MODE_INTER_WORD` (/reference/android/graphics/text/LineBreaker#JUSTIFICATION_MODE_INTER_WORD) |

### See also:

`getJustificationMode()` (/reference/android/widget/TextView#getJustificationMode())

## setKeyListener

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setKeyListener (KeyListener (/reference/android/text/method/KeyListener) input)
```

Sets the key listener to be used with this TextView. This can be null to disallow user input. Note that this method has significant and subtle interactions with soft keyboards and other input method: see `KeyListener#getInputType()` (/reference/android/text/method/KeyListener#getInputType()) for important details. Calling this method will replace the current content type of the text view with the content type returned by the key listener.

Be warned that if you want a TextView with a key listener or movement method not to be focusable, or if you want a TextView without a key listener or movement method to be focusable, you must call `View.setFocusable(boolean)` (/reference/android/view/View#setFocusable(boolean)) again after calling this to get the focusability back the way you want it.

**Related XML Attributes:**

android:numeric (/reference/android/widget/TextView#attr_android:numeric)

android:digits (/reference/android/widget/TextView#attr_android:digits)

android:phoneNumber (/reference/android/widget/TextView#attr_android:phoneNumber)

android:inputMethod (/reference/android/widget/TextView#attr_android:inputMethod)

android:capitalize (/reference/android/widget/TextView#attr_android:capitalize)

android:autoText (/reference/android/widget/TextView#attr_android:autoText)

### Parameters

| | |
|---|---|
| `input` | `KeyListener` |

## setLastBaselineToBottomHeight

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setLastBaselineToBottomHeight (int lastBaselineToBottomHeight)
```

Updates the bottom padding of the TextView so that `lastBaselineToBottomHeight` is the distance between the bottom of the TextView and the last line's baseline.


First and last baseline metrics for a TextView.

**Note** that if `FontMetrics.bottom` or `FontMetrics.descent` was already greater than `lastBaselineToBottomHeight`, the bottom padding is not updated. Moreover since this function sets the bottom padding, if the height of the TextView is less than the sum of top padding, line height and bottom padding, bottom of the text will be clipped.

**Related XML Attributes:**

android:lastBaselineToBottomHeight (/reference/android/widget/TextView#attr_android:lastBaselineToBottomHeight)

### Parameters

`lastBaselineToBottomHeigh t` `int`: distance between last baseline to bottom of the container in pixels This units of this value are pixels. Value is 0 or greater

**See also:**

`getLastBaselineToBottomHeight()` (/reference/android/widget/TextView#getLastBaselineToBottomHeight())

`setFirstBaselineToTopHeight(int)` (/reference/android/widget/TextView#setFirstBaselineToTopHeight(int))

`setPadding(int, int, int, int)` (/reference/android/widget/TextView#setPadding(int,%20int,%20int,%20int))

`setPaddingRelative(int, int, int, int)` (/reference/android/widget/TextView#setPaddingRelative(int,%20int,%20int,%20int))

## setLetterSpacing

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setLetterSpacing (float letterSpacing)
```

Sets text letter-spacing in em units. Typical values for slight expansion will be around 0.05. Negative values tighten text.

**Related XML Attributes:**

android:letterSpacing (/reference/android/widget/TextView#attr_android:letterSpacing)

**Parameters**

| | |
|---|---|
| `letterSpacing` | `float`: A text letter-space value in ems. |

**See also:**

`getLetterSpacing()` (/reference/android/widget/TextView#getLetterSpacing())

`Paint.getLetterSpacing()` (/reference/android/graphics/Paint#getLetterSpacing())

## setLineHeight

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setLineHeight (int lineHeight)
```

Sets an explicit line height for this TextView. This is equivalent to the vertical distance between subsequent baselines in the TextView.

**Related XML Attributes:**

android:lineHeight (/reference/android/widget/TextView#attr_android:lineHeight)

**Parameters**

| | |
|---|---|
| `lineHeight` | `int`: the line height in pixels This units of this value are pixels. Value is 0 or greater |

**See also:**

`setLineSpacing(float, float)` (/reference/android/widget/TextView#setLineSpacing(float,%20float))

`getLineSpacingExtra()` (/reference/android/widget/TextView#getLineSpacingExtra())

## setLineSpacing

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setLineSpacing (float add,
                float mult)
```

Sets line spacing for this TextView. Each line other than the last line will have its height multiplied by **`mult`** and have **`add`** added to it.

**Related XML Attributes:**

android:lineSpacingExtra (/reference/android/widget/TextView#attr_android:lineSpacingExtra)

android:lineSpacingMultiplier (/reference/android/widget/TextView#attr_android:lineSpacingMultiplier)

### Parameters

| | |
|---|---|
| **`add`** | **`float`**: The value in pixels that should be added to each line other than the last line. This will be applied after the multiplier |
| **`mult`** | **`float`**: The value by which each line height other than the last line will be multiplied by |

## setLines

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setLines (int lines)
```

Sets the height of the TextView to be exactly **`lines`** tall.

This value is used for height calculation if LayoutParams does not force TextView to have an exact height. Setting this value overrides previous minimum/maximum height configurations such as **`setMinLines(int)`** (/reference/android/widget/TextView#setMinLines(int)) or **`setMaxLines(int)`** (/reference/android/widget/TextView#setMaxLines(int)). **`setSingleLine()`** (/reference/android/widget/TextView#setSingleLine()) will set this value to 1.

**Related XML Attributes:**

android:lines (/reference/android/widget/TextView#attr_android:lines)

### Parameters

| | |
|---|---|
| `lines` | **`int`**: the exact height of the TextView in terms of lines |

### See also:

`setHeight(int)` (/reference/android/widget/TextView#setHeight(int))

## setLinkTextColor

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void setLinkTextColor (ColorStateList (/reference/android/content/res/ColorStateList) colors)
```

Sets the color of links in the text.

### Related XML Attributes:

android:textColorLink (/reference/android/widget/TextView#attr_android:textColorLink)

### Parameters

| | |
|---|---|
| `colors` | **`ColorStateList`** |

**See also:**

`setLinkTextColor(int)` (/reference/android/widget/TextView#setLinkTextColor(int))

`getLinkTextColors()` (/reference/android/widget/TextView#getLinkTextColors())

`setTextColor(ColorStateList)` (/reference/android/widget/TextView#setTextColor(android.content.res.ColorStateList))

`setHintTextColor(ColorStateList)` (/reference/android/widget/TextView#setHintTextColor(android.content.res.ColorStateList))

## setLinkTextColor

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void setLinkTextColor (int color)
```

Sets the color of links in the text.

**Related XML Attributes:**

android:textColorLink (/reference/android/widget/TextView#attr_android:textColorLink)

### Parameters

| | |
|---|---|
| `color` | `int` |

**See also:**

`setLinkTextColor(ColorStateList)` (/reference/android/widget/TextView#setLinkTextColor(android.content.res.ColorStateList))

`getLinkTextColors()` (/reference/android/widget/TextView#getLinkTextColors())

## setLinksClickable

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void setLinksClickable (boolean whether)
```

Sets whether the movement method will automatically be set to `LinkMovementMethod` (/reference/android/text/method/LinkMovementMethod) if `setAutoLinkMask(int)` (/reference/android/widget/TextView#setAutoLinkMask(int)) has been set to nonzero and links are detected in `setText(char[], int, int)` (/reference/android/widget/TextView#setText(char[],%20int,%20int)). The default is true.

**Related XML Attributes:**

android:linksClickable (/reference/android/widget/TextView#attr_android:linksClickable)

### Parameters

| | |
|---|---|
| `whether` | `boolean` |

## setMarqueeRepeatLimit

Added in API level 2 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setMarqueeRepeatLimit (int marqueeLimit)
```

Sets how many times to repeat the marquee animation. Only applied if the TextView has marquee enabled. Set to -1 to repeat indefinitely.

**Related XML Attributes:**

android:marqueeRepeatLimit (/reference/android/widget/TextView#attr_android:marqueeRepeatLimit)

### Parameters

| | |
|---|---|
| **`marqueeLimit`** | **`int`** |

**See also:**

**`getMarqueeRepeatLimit()`** (/reference/android/widget/TextView#getMarqueeRepeatLimit())

## setMaxEms

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setMaxEms (int maxEms)
```

Sets the width of the TextView to be at most **`maxEms`** wide.

This value is used for width calculation if LayoutParams does not force TextView to have an exact width. Setting this value overrides previous maximum width configurations such as **`setMaxWidth(int)`** (/reference/android/widget/TextView#setMaxWidth(int)) or **`setWidth(int)`** (/reference/android/widget/TextView#setWidth(int)).

### Related XML Attributes:

android:maxEms (/reference/android/widget/TextView#attr_android:maxEms)

### Parameters

| | |
|---|---|
| `maxEms` | `int`: the maximum width of TextView in terms of ems |

### See also:

`getMaxEms()` (/reference/android/widget/TextView#getMaxEms())

`setEms(int)` (/reference/android/widget/TextView#setEms(int))

## setMaxHeight

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setMaxHeight (int maxPixels)
```

Sets the height of the TextView to be at most `maxPixels` tall.

This value is used for height calculation if LayoutParams does not force TextView to have an exact height. Setting this value overrides previous maximum height configurations such as `setMaxLines(int)` (/reference/android/widget/TextView#setMaxLines(int)) or `setLines(int)` (/reference/android/widget/TextView#setLines(int)).

### Related XML Attributes:

android:maxHeight (/reference/android/widget/TextView#attr_android:maxHeight)

### Parameters

| | |
|---|---|
| **`maxPixels`** | **`int`**: the maximum height of TextView in terms of pixels |

### See also:

**`getMaxHeight()`** (/reference/android/widget/TextView#getMaxHeight())

**`setHeight(int)`** (/reference/android/widget/TextView#setHeight(int))

## setMaxLines

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setMaxLines (int maxLines)
```

Sets the height of the TextView to be at most **`maxLines`** tall.

This value is used for height calculation if LayoutParams does not force TextView to have an exact height. Setting this value overrides previous maximum height configurations such as **`setMaxHeight(int)`** (/reference/android/widget/TextView#setMaxHeight(int)) or **`setLines(int)`** (/reference/android/widget/TextView#setLines(int)).

### Related XML Attributes:

android:maxLines (/reference/android/widget/TextView#attr_android:maxLines)

### Parameters

| | |
|---|---|
| `maxLines` | `int`: the maximum height of TextView in terms of number of lines |

### See also:

`getMaxLines()` (/reference/android/widget/TextView#getMaxLines())

`setLines(int)` (/reference/android/widget/TextView#setLines(int))

## setMaxWidth

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setMaxWidth (int maxPixels)
```

Sets the width of the TextView to be at most `maxPixels` wide.

This value is used for width calculation if LayoutParams does not force TextView to have an exact width. Setting this value overrides previous maximum width configurations such as `setMaxEms(int)` (/reference/android/widget/TextView#setMaxEms(int)) or `setEms(int)` (/reference/android/widget/TextView#setEms(int)).

### Related XML Attributes:

android:maxWidth (/reference/android/widget/TextView#attr_android:maxWidth)

### Parameters

| | |
|---|---|
| `maxPixels` | `int`: the maximum width of TextView in terms of pixels |

**See also:**

**`getMaxWidth()`** (/reference/android/widget/TextView#getMaxWidth())

**`setWidth(int)`** (/reference/android/widget/TextView#setWidth(int))

## setMinEms

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setMinEms (int minEms)
```

Sets the width of the TextView to be at least **`minEms`** wide.

This value is used for width calculation if LayoutParams does not force TextView to have an exact width. Setting this value overrides previous minimum width configurations such as **`setMinWidth(int)`** (/reference/android/widget/TextView#setMinWidth(int)) or **`setWidth(int)`** (/reference/android/widget/TextView#setWidth(int)).

**Related XML Attributes:**

android:minEms (/reference/android/widget/TextView#attr_android:minEms)

### Parameters

| | |
|---|---|
| **`minEms`** | **`int`**: the minimum width of TextView in terms of ems |

**See also:**

`getMinEms()` (/reference/android/widget/TextView#getMinEms())

`setEms(int)` (/reference/android/widget/TextView#setEms(int))

## setMinHeight

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setMinHeight (int minPixels)
```

Sets the height of the TextView to be at least **`minPixels`** tall.

This value is used for height calculation if LayoutParams does not force TextView to have an exact height. Setting this value overrides previous minimum height configurations such as `setMinLines(int)` (/reference/android/widget/TextView#setMinLines(int)) or `setLines(int)` (/reference/android/widget/TextView#setLines(int)).

The value given here is different than `View.setMinimumHeight(int)` (/reference/android/view/View#setMinimumHeight(int)). Between **`minHeight`** and the value set in `View.setMinimumHeight(int)` (/reference/android/view/View#setMinimumHeight(int)), the greater one is used to decide the final height.

**Related XML Attributes:**

android:minHeight (/reference/android/widget/TextView#attr_android:minHeight)

### Parameters

| | |
|---|---|
| **`minPixels`** | **`int`**: the minimum height of TextView in terms of pixels |

**See also:**

`getMinHeight()` (/reference/android/widget/TextView#getMinHeight())

`setHeight(int)` (/reference/android/widget/TextView#setHeight(int))

## setMinLines

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setMinLines (int minLines)
```

Sets the height of the TextView to be at least `minLines` tall.

This value is used for height calculation if LayoutParams does not force TextView to have an exact height. Setting this value overrides other previous minimum height configurations such as `setMinHeight(int)` (/reference/android/widget/TextView#setMinHeight(int)) or `setHeight(int)` (/reference/android/widget/TextView#setHeight(int)). `setSingleLine()` (/reference/android/widget/TextView#setSingleLine()) will set this value to 1.

**Related XML Attributes:**

android:minLines (/reference/android/widget/TextView#attr_android:minLines)

### Parameters

| | |
|---|---|
| `minLines` | `int`: the minimum height of TextView in terms of number of lines |

**See also:**

`getMinLines()` (/reference/android/widget/TextView#getMinLines())

`setLines(int)` (/reference/android/widget/TextView#setLines(int))

## setMinWidth

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setMinWidth (int minPixels)
```

Sets the width of the TextView to be at least **`minPixels`** wide.

This value is used for width calculation if LayoutParams does not force TextView to have an exact width. Setting this value overrides previous minimum width configurations such as `setMinEms(int)` (/reference/android/widget/TextView#setMinEms(int)) or `setEms(int)` (/reference/android/widget/TextView#setEms(int)).

The value given here is different than `View.setMinimumWidth(int)` (/reference/android/view/View#setMinimumWidth(int)). Between **`minWidth`** and the value set in `View.setMinimumWidth(int)` (/reference/android/view/View#setMinimumWidth(int)), the greater one is used to decide the final width.

**Related XML Attributes:**

android:minWidth (/reference/android/widget/TextView#attr_android:minWidth)

### Parameters

| | |
|---|---|
| **`minPixels`** | **`int`**: the minimum width of TextView in terms of pixels |

### See also:

**getMinWidth()** (/reference/android/widget/TextView#getMinWidth())

**setWidth(int)** (/reference/android/widget/TextView#setWidth(int))

## setMovementMethod

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void setMovementMethod (MovementMethod (/reference/android/text/method/MovementMethod) movement)
```

Sets the **MovementMethod** (/reference/android/text/method/MovementMethod) for handling arrow key movement for this TextView. This can be null to disallow using the arrow keys to move the cursor or scroll the view.

Be warned that if you want a TextView with a key listener or movement method not to be focusable, or if you want a TextView without a key listener or movement method to be focusable, you must call **View.setFocusable(boolean)** (/reference/android/view/View#setFocusable(boolean)) again after calling this to get the focusability back the way you want it.

### Parameters

| | |
|---|---|
| **movement** | **MovementMethod** |

## setOnEditorActionListener

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setOnEditorActionListener (TextView.OnEditorActionListener (/reference/android/widget/TextView.OnEditorActionListener) l)
```

Set a special listener to be called when an action is performed on the text view. This will be called when the enter key is pressed, or when an action supplied to the IME is selected by the user. Setting this means that the normal hard key event will not insert a newline into the text view, even if it is multi-line; holding down the ALT modifier will, however, allow the user to insert a newline character.

### Parameters

| | |
|---|---|
| `l` | `TextView.OnEditorActionListener` |

## setPadding

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setPadding (int left,
                int top,
                int right,
                int bottom)
```

Sets the padding. The view may add on the space required to display the scrollbars, depending on the style and visibility of the scrollbars. So the values returned from **`getPaddingLeft()`** (/reference/android/view/View#getPaddingLeft()), **`getPaddingTop()`** (/reference/android/view/View#getPaddingTop()), **`getPaddingRight()`** (/reference/android/view/View#getPaddingRight()) and **`getPaddingBottom()`** (/reference/android/view/View#getPaddingBottom()) may be different from the values set in this call.

### Parameters

| | |
|---|---|
| `left` | **`int`**: the left padding in pixels |

| | |
|---|---|
| `top` | `int`: the top padding in pixels |
| `right` | `int`: the right padding in pixels |
| `bottom` | `int`: the bottom padding in pixels |

### See also:

`setFirstBaselineToTopHeight(int)` (/reference/android/widget/TextView#setFirstBaselineToTopHeight(int))

`setLastBaselineToBottomHeight(int)` (/reference/android/widget/TextView#setLastBaselineToBottomHeight(int))

## setPaddingRelative

Added in API level 16 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setPaddingRelative (int start,
                int top,
                int end,
                int bottom)
```

Sets the relative padding. The view may add on the space required to display the scrollbars, depending on the style and visibility of the scrollbars. So the values returned from `getPaddingStart()` (/reference/android/view/View#getPaddingStart()), `getPaddingTop()` (/reference/android/view/View#getPaddingTop()), `getPaddingEnd()` (/reference/android/view/View#getPaddingEnd()) and `getPaddingBottom()` (/reference/android/view/View#getPaddingBottom()) may be different from the values set in this call.

### Parameters

| | |
|---|---|
| `start` | `int`: the start padding in pixels |
| `top` | `int`: the top padding in pixels |
| `end` | `int`: the end padding in pixels |
| `bottom` | `int`: the bottom padding in pixels |

### See also:

`setFirstBaselineToTopHeight(int)` (/reference/android/widget/TextView#setFirstBaselineToTopHeight(int))

`setLastBaselineToBottomHeight(int)` (/reference/android/widget/TextView#setLastBaselineToBottomHeight(int))

## setPaintFlags

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setPaintFlags (int flags)
```

Sets flags on the Paint being used to display the text and reflows the text if they are different from the old flags.

### Parameters

| | |
|---|---|
| `flags` | `int` |

**See also:**

`Paint.setFlags(int)` (/reference/android/graphics/Paint#setFlags(int))

## setPrivateImeOptions

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setPrivateImeOptions (String type)
```

Set the private content type of the text, which is the `EditorInfo#privateImeOptions` (/reference/android/view/inputmethod/EditorInfo#privateImeOptions) field that will be filled in when creating an input connection.

**Related XML Attributes:**

android:privateImeOptions (/reference/android/widget/TextView#attr_android:privateImeOptions)

**Parameters**

| | |
|---|---|
| `type` | `String` |

**See also:**

`getPrivateImeOptions()` (/reference/android/widget/TextView#getPrivateImeOptions())

`EditorInfo.privateImeOptions` (/reference/android/view/inputmethod/EditorInfo#privateImeOptions)

## setRawInputType

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setRawInputType (int type)
```

Directly change the content type integer of the text view, without modifying any other state.

**Related XML Attributes:**

android:inputType (/reference/android/widget/TextView#attr_android:inputType)

### Parameters

| | |
|---|---|
| `type` | `int` |

### See also:

`setInputType(int)` (/reference/android/widget/TextView#setInputType(int))

`InputType` (/reference/android/text/InputType)

## setScroller

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setScroller (Scroller (/reference/android/widget/Scroller) s)
```

Sets the Scroller used for producing a scrolling animation

### Parameters

| | |
|---|---|
| `s` | `Scroller`: A Scroller instance |

## setSelectAllOnFocus

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setSelectAllOnFocus (boolean selectAllOnFocus)
```

Set the TextView so that when it takes focus, all the text is selected.

**Related XML Attributes:**

android:selectAllOnFocus (/reference/android/widget/TextView#attr_android:selectAllOnFocus)

### Parameters

| | |
|---|---|
| `selectAllOnFocus` | `boolean` |

## setSelected

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setSelected (boolean selected)
```

Changes the selection state of this view. A view can be selected or not. Note that selection is not the same as focus. Views are typically selected in the context of an AdapterView like ListView or GridView; the selected view is the view that is highlighted.

### Parameters

| | |
|---|---|
| **`selected`** | **`boolean`**: true if the view must be selected, false otherwise |

## setShadowLayer

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setShadowLayer (float radius,
                float dx,
                float dy,
                int color)
```

Gives the text a shadow of the specified blur radius and color, the specified distance from its drawn position.

The text shadow produced does not interact with the properties on view that are responsible for real time shadows, **`View#getElevation()`** (/reference/android/view/View#getElevation()) and **`View#getTranslationZ()`** (/reference/android/view/View#getTranslationZ()).

**Related XML Attributes:**

android:shadowColor (/reference/android/widget/TextView#attr_android:shadowColor)

android:shadowDx (/reference/android/widget/TextView#attr_android:shadowDx)

android:shadowDy (/reference/android/widget/TextView#attr_android:shadowDy)

android:shadowRadius (/reference/android/widget/TextView#attr_android:shadowRadius)

### Parameters

| | |
|---|---|
| `radius` | `float` |
| `dx` | `float` |
| `dy` | `float` |
| `color` | `int` |

### See also:

`Paint.setShadowLayer(float, float, float, int)` (/reference/android/graphics/Paint#setShadowLayer(float,%20float,%20float,%20int))

## setShowSoftInputOnFocus

Added in API level 21 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void setShowSoftInputOnFocus (boolean show)
```

Sets whether the soft input method will be made visible when this TextView gets focused. The default is true.

### Parameters

| | |
|---|---|
| **show** | **boolean** |

## setSingleLine

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setSingleLine (boolean singleLine)
```

If true, sets the properties of this field (number of lines, horizontally scrolling, transformation method) to be for a single-line input; if false, restores these to the default conditions. Note that the default conditions are not necessarily those that were in effect prior this method, and you may want to reset these properties to your custom values.

**Related XML Attributes:**

android:singleLine (/reference/android/widget/TextView#attr_android:singleLine)

### Parameters

| | |
|---|---|
| **singleLine** | **boolean** |

## setSingleLine

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setSingleLine ()
```

Sets the properties of this field (lines, horizontally scrolling, transformation method) to be for a single-line input.

**Related XML Attributes:**

android:singleLine (/reference/android/widget/TextView#attr_android:singleLine)

## setSpannableFactory

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void setSpannableFactory (Spannable.Factory (/reference/android/text/Spannable.Factory) factory)
```

Sets the Factory used to create new **`Spannable`** (/reference/android/text/Spannable).

**Parameters**

| | |
|---|---|
| `factory` | **`Spannable.Factory`**: **`Spannable.Factory`** (/reference/android/text/Spannable.Factory) to be used |

**See also:**

**`Spannable.Factory`** (/reference/android/text/Spannable.Factory)

`TextView.BufferType.SPANNABLE` (/reference/android/widget/TextView.BufferType#SPANNABLE)

## setText

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void setText (int resid)
```

Sets the text to be displayed using a string resource identifier.

**Related XML Attributes:**

android:text (/reference/android/widget/TextView#attr_android:text)

### Parameters

| | |
|---|---|
| `resid` | `int`: the resource identifier of the string resource to be displayed |

### See also:

`setText(CharSequence)` (/reference/android/widget/TextView#setText(java.lang.CharSequence))

## setText

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public final void setText (CharSequence (/reference/java/lang/CharSequence) text)`

Sets the text to be displayed. TextView *does not* accept HTML-like formatting, which you can do with text strings in XML resource files. To style your strings, attach android.text.style.* objects to a **`SpannableString`** (/reference/android/text/SpannableString), or see the Available Resource Types (/guide/topics/resources/available-resources#stringresources) documentation for an example of setting formatted text in the XML resource file.

When required, TextView will use **`Spannable.Factory`** (/reference/android/text/Spannable.Factory) to create final or intermediate **`Spannable`** (/reference/android/text/Spannable). Likewise it will use **`Editable.Factory`** (/reference/android/text/Editable.Factory) to create final or intermediate **`Editable`** (/reference/android/text/Editable). If the passed text is a **`PrecomputedText`** (/reference/android/text/PrecomputedText) but the parameters used to create the PrecomputedText mismatches with this TextView, IllegalArgumentException is thrown. To ensure the parameters match, you can call **`TextView#setTextMetricsParams`** (/reference/android/widget/TextView#setTextMetricsParams(android.text.PrecomputedText.Params)) before calling this.

**Related XML Attributes:**

android:text (/reference/android/widget/TextView#attr_android:text)

### Parameters

| | |
|---|---|
| `text` | **`CharSequence`**: text to be displayed |

### Throws

| | |
|---|---|
| **`IllegalArgumentException`** (/reference/java/lang/IllegalArgumentException) | if the passed text is a **`PrecomputedText`** (/reference/android/text/PrecomputedText) but the parameters used to create the PrecomputedText mismatches with this TextView. |

# setText

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setText (CharSequence (/reference/java/lang/CharSequence) text,
                TextView.BufferType (/reference/android/widget/TextView.BufferType) type)
```

Sets the text to be displayed and the **`TextView.BufferType`** (/reference/android/widget/TextView.BufferType).

When required, TextView will use **`Spannable.Factory`** (/reference/android/text/Spannable.Factory) to create final or intermediate **`Spannable`** (/reference/android/text/Spannable). Likewise it will use **`Editable.Factory`** (/reference/android/text/Editable.Factory) to create final or intermediate **`Editable`** (/reference/android/text/Editable). Subclasses overriding this method should ensure that the following post condition holds, in order to guarantee the safety of the view's measurement and layout operations: regardless of the input, after calling #setText both **`mText`** and **`mTransformed`** will be different from **`null`**.

**Related XML Attributes:**

android:text (/reference/android/widget/TextView#attr_android:text)

android:bufferType (/reference/android/widget/TextView#attr_android:bufferType)

### Parameters

| | |
|---|---|
| **`text`** | **`CharSequence`**: text to be displayed |
| **`type`** | **`TextView.BufferType`**: a **`TextView.BufferType`** (/reference/android/widget/TextView.BufferType) which defines whether the text is stored as a static text, styleable/spannable text, or editable text |

### See also:

`setText(CharSequence)` (/reference/android/widget/TextView#setText(java.lang.CharSequence))

`TextView.BufferType` (/reference/android/widget/TextView.BufferType)

`setSpannableFactory(Spannable.Factory)` (/reference/android/widget/TextView#setSpannableFactory(android.text.Spannable.Factory))

`setEditableFactory(Editable.Factory)` (/reference/android/widget/TextView#setEditableFactory(android.text.Editable.Factory))

## setText

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void setText (int resid,
                TextView.BufferType (/reference/android/widget/TextView.BufferType) type)
```

Sets the text to be displayed using a string resource identifier and the `TextView.BufferType` (/reference/android/widget/TextView.BufferType).

When required, TextView will use `Spannable.Factory` (/reference/android/text/Spannable.Factory) to create final or intermediate `Spannable` (/reference/android/text/Spannable). Likewise it will use `Editable.Factory` (/reference/android/text/Editable.Factory) to create final or intermediate `Editable` (/reference/android/text/Editable).

**Related XML Attributes:**

android:text (/reference/android/widget/TextView#attr_android:text)

android:bufferType (/reference/android/widget/TextView#attr_android:bufferType)

### Parameters

| | |
|---|---|
| `resid` | `int`: the resource identifier of the string resource to be displayed |

| | |
|---|---|
| `type` | **`TextView.BufferType`**: a **`TextView.BufferType`** (/reference/android/widget/TextView.BufferType) which defines whether the text is stored as a static text, styleable/spannable text, or editable text |

### See also:

**`setText(int)`** (/reference/android/widget/TextView#setText(int))

**`setText(CharSequence)`** (/reference/android/widget/TextView#setText(java.lang.CharSequence))

**`TextView.BufferType`** (/reference/android/widget/TextView.BufferType)

**`setSpannableFactory(Spannable.Factory)`** (/reference/android/widget/TextView#setSpannableFactory(android.text.Spannable.Factory))

**`setEditableFactory(Editable.Factory)`** (/reference/android/widget/TextView#setEditableFactory(android.text.Editable.Factory))

## setText

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void setText (char[] text,
                int start,
                int len)
```

Sets the TextView to display the specified slice of the specified char array. You must promise that you will not change the contents of the array except for right before another call to setText(), since the TextView has no way to know that the text has changed and that it needs to invalidate and re-layout.

### Parameters

| | |
|---|---|
| **text** | **char**: char array to be displayed |
| **start** | **int**: start index in the char array |
| **len** | **int**: length of char count after **start** |

## setTextAppearance

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)
Deprecated in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setTextAppearance (Context (/reference/android/content/Context) context,
                int resId)
```

**This method was deprecated in API level 23.**

Use **setTextAppearance(int)** (/reference/android/widget/TextView#setTextAppearance(int)) instead.

Sets the text color, size, style, hint color, and highlight color from the specified TextAppearance resource.

### Parameters

| | |
|---|---|
| **context** | **Context** |
| **resId** | **int** |

## setTextAppearance

Added in API level 23 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setTextAppearance (int resId)
```

Sets the text appearance from the specified style resource.

Use a framework-defined `TextAppearance` style like `@android:style/TextAppearance.Material.Body1` (/reference/android/R.style#TextAppearance_Material_Body1) or see `TextAppearance` (/reference/android/R.styleable#TextAppearance) for the set of attributes that can be used in a custom style.

**Related XML Attributes:**

android:textAppearance (/reference/android/widget/TextView#attr_android:textAppearance)

### Parameters

| | |
|---|---|
| `resId` | `int`: the resource identifier of the style to apply |

## setTextClassifier

Added in API level 26 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setTextClassifier (TextClassifier (/reference/android/view/textclassifier/TextClassifier) textClassifier)
```

Sets the `TextClassifier` (/reference/android/view/textclassifier/TextClassifier) for this TextView.

### Parameters

| | |
|---|---|
| `textClassifier` | `TextClassifier`: This value may be `null`. |

## setTextColor

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setTextColor (int color)
```

Sets the text color for all the states (normal, selected, focused) to be this color.

### Related XML Attributes:

android:textColor (/reference/android/widget/TextView#attr_android:textColor)

### Parameters

| | |
|---|---|
| `color` | `int`: A color value in the form 0xAARRGGBB. Do not pass a resource ID. To get a color value from a resource ID, call `getColor` (https://developer.android.com/reference/android/support/v4/content/ContextCompat.html#getColor(android.content.Context,%20int)). |

### See also:

`setTextColor(ColorStateList)` (/reference/android/widget/TextView#setTextColor(android.content.res.ColorStateList))

`getTextColors()` (/reference/android/widget/TextView#getTextColors())

## setTextColor

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setTextColor (ColorStateList (/reference/android/content/res/ColorStateList) colors)
```

Sets the text color.

**Related XML Attributes:**

android:textColor (/reference/android/widget/TextView#attr_android:textColor)

### Parameters

| | |
|---|---|
| `colors` | `ColorStateList` |

### See also:

`setTextColor(int)` (/reference/android/widget/TextView#setTextColor(int))

`getTextColors()` (/reference/android/widget/TextView#getTextColors())

`setHintTextColor(ColorStateList)` (/reference/android/widget/TextView#setHintTextColor(android.content.res.ColorStateList))

`setLinkTextColor(ColorStateList)` (/reference/android/widget/TextView#setLinkTextColor(android.content.res.ColorStateList))

## setTextCursorDrawable

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setTextCursorDrawable (Drawable (/reference/android/graphics/drawable/Drawable) textCursorDrawable)
```

Sets the Drawable corresponding to the text cursor. The Drawable defaults to the value of the textCursorDrawable attribute. Note that any change applied to the cursor Drawable will not be visible until the cursor is hidden and then drawn again.

**Related XML Attributes:**

android:textCursorDrawable (/reference/android/widget/TextView#attr_android:textCursorDrawable)

### Parameters

| | |
|---|---|
| `textCursorDrawable` | **`Drawable`**: This value may be **`null`**. |

### See also:

**`setTextCursorDrawable(int)`** (/reference/android/widget/TextView#setTextCursorDrawable(int))

## setTextCursorDrawable

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setTextCursorDrawable (int textCursorDrawable)
```

Sets the Drawable corresponding to the text cursor. The Drawable defaults to the value of the textCursorDrawable attribute. Note that any change applied to the cursor

Drawable will not be visible until the cursor is hidden and then drawn again.

**Related XML Attributes:**

android:textCursorDrawable (/reference/android/widget/TextView#attr_android:textCursorDrawable)

**Parameters**

| | |
|---|---|
| `textCursorDrawable` | `int` |

**See also:**

`setTextCursorDrawable(Drawable)` (/reference/android/widget/TextView#setTextCursorDrawable(android.graphics.drawable.Drawable))

## setTextIsSelectable

Added in API level 11 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setTextIsSelectable (boolean selectable)
```

Sets whether the content of this view is selectable by the user. The default is `false`, meaning that the content is not selectable.

When you use a TextView to display a useful piece of information to the user (such as a contact's address), make it selectable, so that the user can select and copy its content. You can also use set the XML attribute `R.styleable.TextView_textIsSelectable` (/reference/android/R.styleable#TextView_textIsSelectable) to "true".

When you call this method to set the value of `textIsSelectable`, it sets the flags `focusable`, `focusableInTouchMode`, `clickable`, and `longClickable` to the same value. These flags correspond to the attributes `android:focusable` (/reference/android/R.styleable#View_focusable), `android:focusableInTouchMode`

(/reference/android/R.styleable#View_focusableInTouchMode), **android:clickable** (/reference/android/R.styleable#View_clickable), and **android:longClickable** (/reference/android/R.styleable#View_longClickable). To restore any of these flags to a state you had set previously, call one or more of the following methods: **setFocusable()** (/reference/android/view/View#setFocusable(boolean)), **setFocusableInTouchMode()** (/reference/android/view/View#setFocusableInTouchMode(boolean)), **setClickable()** (/reference/android/view/View#setClickable(boolean)) or **setLongClickable()** (/reference/android/view/View#setLongClickable(boolean)).

### Parameters

| | |
|---|---|
| **selectable** | **boolean**: Whether the content of this TextView should be selectable. |

## setTextKeepState

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void setTextKeepState (CharSequence (/reference/java/lang/CharSequence) text)
```

Sets the text to be displayed but retains the cursor position. Same as **setText(java.lang.CharSequence)** (/reference/android/widget/TextView#setText(java.lang.CharSequence)) except that the cursor position (if any) is retained in the new text.

When required, TextView will use **Spannable.Factory** (/reference/android/text/Spannable.Factory) to create final or intermediate **Spannable** (/reference/android/text/Spannable). Likewise it will use **Editable.Factory** (/reference/android/text/Editable.Factory) to create final or intermediate **Editable** (/reference/android/text/Editable).

### Parameters

| | |
|---|---|
| **text** | **CharSequence**: text to be displayed |

### See also:

`setText(CharSequence)` (/reference/android/widget/TextView#setText(java.lang.CharSequence))

## setTextKeepState

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void setTextKeepState (CharSequence (/reference/java/lang/CharSequence) text,
                TextView.BufferType (/reference/android/widget/TextView.BufferType) type)
```

Sets the text to be displayed and the `TextView.BufferType` (/reference/android/widget/TextView.BufferType) but retains the cursor position. Same as `setText(java.lang.CharSequence, android.widget.TextView.BufferType)` (/reference/android/widget/TextView#setText(java.lang.CharSequence,%20android.widget.TextView.BufferType)) except that the cursor position (if any) is retained in the new text.

When required, TextView will use `Spannable.Factory` (/reference/android/text/Spannable.Factory) to create final or intermediate `Spannable` (/reference/android/text/Spannable). Likewise it will use `Editable.Factory` (/reference/android/text/Editable.Factory) to create final or intermediate `Editable` (/reference/android/text/Editable).

### Parameters

| | |
|---|---|
| `text` | **`CharSequence`**: text to be displayed |
| `type` | **`TextView.BufferType`**: a `TextView.BufferType` (/reference/android/widget/TextView.BufferType) which defines whether the text is stored as a static text, styleable/spannable text, or editable text |

### See also:

**setText(CharSequence, android.widget.TextView.BufferType)** (/reference/android/widget/TextView#setText(java.lang.CharSequence,%20android.widget.TextView.BufferType))

## setTextLocale

Added in API level 17 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setTextLocale (Locale (/reference/java/util/Locale) locale)
```

Set the default **Locale** (/reference/java/util/Locale) of the text in this TextView to a one-member **LocaleList** (/reference/android/os/LocaleList) containing just the given Locale.

### Parameters

| | |
|---|---|
| **locale** | **Locale**: the **Locale** (/reference/java/util/Locale) for drawing text, must not be null. This value must never be **null**. |

### See also:

**setTextLocales(LocaleList)** (/reference/android/widget/TextView#setTextLocales(android.os.LocaleList))

## setTextLocales

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setTextLocales (LocaleList (/reference/android/os/LocaleList) locales)
```

Set the default **LocaleList** (/reference/android/os/LocaleList) of the text in this TextView to the given value. This value is used to choose appropriate typefaces for ambiguous

characters (typically used for CJK locales to disambiguate Hanzi/Kanji/Hanja characters). It also affects other aspects of text display, including line breaking.

### Parameters

| | |
|---|---|
| **`locales`** | **`LocaleList`**: the **`LocaleList`** (/reference/android/os/LocaleList) for drawing text, must not be null or empty. This value must never be **`null`**. |

### See also:

**`Paint.setTextLocales(LocaleList)`** (/reference/android/graphics/Paint#setTextLocales(android.os.LocaleList))

## setTextMetricsParams

Added in API level 28 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setTextMetricsParams (PrecomputedText.Params (/reference/android/text/PrecomputedText.Params) params)
```

Apply the text layout parameter. Update the TextView parameters to be compatible with **`PrecomputedText.Params`** (/reference/android/text/PrecomputedText.Params).

### Parameters

| | |
|---|---|
| **`params`** | **`PrecomputedText.Params`**: This value must never be **`null`**. |

### See also:

**`PrecomputedText`** (/reference/android/text/PrecomputedText)

## setTextScaleX

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setTextScaleX (float size)
```

Sets the horizontal scale factor for text. The default value is 1.0. Values greater than 1.0 stretch the text wider. Values less than 1.0 make the text narrower. By default, this value is 1.0.

**Related XML Attributes:**

android:textScaleX (/reference/android/widget/TextView#attr_android:textScaleX)

### Parameters

| | |
|---|---|
| `size` | `float`: The horizontal scale factor. |

## setTextSelectHandle

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setTextSelectHandle (int textSelectHandle)
```

Sets the Drawable corresponding to the selection handle used for positioning the cursor within text. The Drawable defaults to the value of the textSelectHandle attribute. Note that any change applied to the handle Drawable will not be visible until the handle is hidden and then drawn again.

**Related XML Attributes:**

android:textSelectHandle (/reference/android/widget/TextView#attr_android:textSelectHandle)

### Parameters

| | |
|---|---|
| `textSelectHandle` | `int` |

**See also:**

`setTextSelectHandle(Drawable)` (/reference/android/widget/TextView#setTextSelectHandle(android.graphics.drawable.Drawable))

## setTextSelectHandle

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setTextSelectHandle (Drawable (/reference/android/graphics/drawable/Drawable) textSelectHandle)
```

Sets the Drawable corresponding to the selection handle used for positioning the cursor within text. The Drawable defaults to the value of the textSelectHandle attribute. Note that any change applied to the handle Drawable will not be visible until the handle is hidden and then drawn again.

**Related XML Attributes:**

android:textSelectHandle (/reference/android/widget/TextView#attr_android:textSelectHandle)

### Parameters

| | |
|---|---|
| `textSelectHandle` | `Drawable`: This value must never be `null`. |

**See also:**

`setTextSelectHandle(int)` (/reference/android/widget/TextView#setTextSelectHandle(int))

## setTextSelectHandleLeft

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setTextSelectHandleLeft (int textSelectHandleLeft)
```

Sets the Drawable corresponding to the left handle used for selecting text. The Drawable defaults to the value of the textSelectHandleLeft attribute. Note that any change applied to the handle Drawable will not be visible until the handle is hidden and then drawn again.

**Related XML Attributes:**

android:textSelectHandleLeft (/reference/android/widget/TextView#attr_android:textSelectHandleLeft)

### Parameters

| | |
|---|---|
| `textSelectHandleLeft` | `int` |

**See also:**

`setTextSelectHandleLeft(Drawable)` (/reference/android/widget/TextView#setTextSelectHandleLeft(android.graphics.drawable.Drawable))

## setTextSelectHandleLeft

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public void setTextSelectHandleLeft (Drawable (/reference/android/graphics/drawable/Drawable) textSelectHandleLeft)`

Sets the Drawable corresponding to the left handle used for selecting text. The Drawable defaults to the value of the textSelectHandleLeft attribute. Note that any change applied to the handle Drawable will not be visible until the handle is hidden and then drawn again.

**Related XML Attributes:**

android:textSelectHandleLeft (/reference/android/widget/TextView#attr_android:textSelectHandleLeft)

### Parameters

| | |
|---|---|
| **`textSelectHandleLeft`** | **`Drawable`**: This value must never be **`null`**. |

### See also:

**`setTextSelectHandleLeft(int)`** (/reference/android/widget/TextView#setTextSelectHandleLeft(int))

## setTextSelectHandleRight

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`public void setTextSelectHandleRight (Drawable (/reference/android/graphics/drawable/Drawable) textSelectHandleRight)`

Sets the Drawable corresponding to the right handle used for selecting text. The Drawable defaults to the value of the textSelectHandleRight attribute. Note that any change

applied to the handle Drawable will not be visible until the handle is hidden and then drawn again.

**Related XML Attributes:**

android:textSelectHandleRight (/reference/android/widget/TextView#attr_android:textSelectHandleRight)

### Parameters

| | |
|---|---|
| `textSelectHandleRight` | **`Drawable`**: This value must never be **`null`**. |

**See also:**

**`setTextSelectHandleRight(int)`** (/reference/android/widget/TextView#setTextSelectHandleRight(int))

## setTextSelectHandleRight

Added in API level 29 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setTextSelectHandleRight (int textSelectHandleRight)
```

Sets the Drawable corresponding to the right handle used for selecting text. The Drawable defaults to the value of the textSelectHandleRight attribute. Note that any change applied to the handle Drawable will not be visible until the handle is hidden and then drawn again.

**Related XML Attributes:**

android:textSelectHandleRight (/reference/android/widget/TextView#attr_android:textSelectHandleRight)

### Parameters

| | |
|---|---|
| `textSelectHandleRight` | `int` |

### See also:

`setTextSelectHandleRight(Drawable)` (/reference/android/widget/TextView#setTextSelectHandleRight(android.graphics.drawable.Drawable))

## setTextSize

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setTextSize (int unit,
                float size)
```

Set the default text size to a given unit and value. See `TypedValue` (/reference/android/util/TypedValue) for the possible dimension units.

Note: if this TextView has the auto-size feature enabled than this function is no-op.

### Related XML Attributes:

android:textSize (/reference/android/widget/TextView#attr_android:textSize)

### Parameters

| | |
|---|---|
| `unit` | `int`: The desired dimension unit. |

| | |
|---|---|
| **size** | **float**: The desired size in the given units. |

## setTextSize

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setTextSize (float size)
```

Set the default text size to the given value, interpreted as "scaled pixel" units. This size is adjusted based on the current density and user font size preference.

Note: if this TextView has the auto-size feature enabled than this function is no-op.

**Related XML Attributes:**

android:textSize (/reference/android/widget/TextView#attr_android:textSize)

### Parameters

| | |
|---|---|
| **size** | **float**: The scaled pixel size. |

## setTransformationMethod

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public final void setTransformationMethod (TransformationMethod (/reference/android/text/method/TransformationMethod) method)
```

Sets the transformation that is applied to the text that this TextView is displaying.

**Related XML Attributes:**

android:password (/reference/android/widget/TextView#attr_android:password)

android:singleLine (/reference/android/widget/TextView#attr_android:singleLine)

**Parameters**

| | |
|---|---|
| `method` | `TransformationMethod` |

## setTypeface

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setTypeface (Typeface (/reference/android/graphics/Typeface) tf)
```

Sets the typeface and style in which the text should be displayed. Note that not all Typeface families actually have bold and italic variants, so you may need to use `setTypeface(android.graphics.Typeface, int)` (/reference/android/widget/TextView#setTypeface(android.graphics.Typeface,%20int)) to get the appearance that you actually want.

**Related XML Attributes:**

android:fontFamily (/reference/android/widget/TextView#attr_android:fontFamily)

android:typeface (/reference/android/widget/TextView#attr_android:typeface)

android:textStyle (/reference/android/widget/TextView#attr_android:textStyle)

### Parameters

| | |
|---|---|
| `tf` | **`Typeface`**: This value may be **`null`**. |

### See also:

**`getTypeface()`** (/reference/android/widget/TextView#getTypeface())

## setTypeface

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setTypeface (Typeface (/reference/android/graphics/Typeface) tf,
                int style)
```

Sets the typeface and style in which the text should be displayed, and turns on the fake bold and italic bits in the Paint if the Typeface that you provided does not have all the bits in the style that you specified.

### Related XML Attributes:

android:typeface (/reference/android/widget/TextView#attr_android:typeface)

android:textStyle (/reference/android/widget/TextView#attr_android:textStyle)

### Parameters

| | |
|---|---|
| `tf` | `Typeface`: This value may be `null`. |
| `style` | `int`: Value is `Typeface.NORMAL` (/reference/android/graphics/Typeface#NORMAL), `Typeface.BOLD` (/reference/android/graphics/Typeface#BOLD), `Typeface.ITALIC` (/reference/android/graphics/Typeface#ITALIC), or `Typeface.BOLD_ITALIC` (/reference/android/graphics/Typeface#BOLD_ITALIC) |

## setWidth

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public void setWidth (int pixels)
```

Sets the width of the TextView to be exactly `pixels` wide.

This value is used for width calculation if LayoutParams does not force TextView to have an exact width. Setting this value overrides previous minimum/maximum width configurations such as `setMinWidth(int)` (/reference/android/widget/TextView#setMinWidth(int)) or `setMaxWidth(int)` (/reference/android/widget/TextView#setMaxWidth(int)).

**Related XML Attributes:**

android:width (/reference/android/widget/TextView#attr_android:width)

### Parameters

| | |
|---|---|
| `pixels` | `int`: the exact width of the TextView in terms of pixels |

**See also:**

`setEms(int)` (/reference/android/widget/TextView#setEms(int))

## showContextMenu

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean showContextMenu ()
```

Shows the context menu for this view.

### Returns

| | |
|---|---|
| **boolean** | **true** if the context menu was shown, **false** otherwise |

## showContextMenu

Added in API level 24 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
public boolean showContextMenu (float x,
                float y)
```

Shows the context menu for this view anchored to the specified view-relative coordinate.

### Parameters

| | |
|---|---|
| **x** | **`float`**: the X coordinate in pixels relative to the view to which the menu should be anchored, or **`Float#NaN`** (/reference/java/lang/Float#NaN) to disable anchoring |
| **y** | **`float`**: the Y coordinate in pixels relative to the view to which the menu should be anchored, or **`Float#NaN`** (/reference/java/lang/Float#NaN) to disable anchoring |

### Returns

| | |
|---|---|
| **`boolean`** | **`true`** if the context menu was shown, **`false`** otherwise |

## Protected methods

### computeHorizontalScrollRange

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected int computeHorizontalScrollRange ()
```

Compute the horizontal range that the horizontal scrollbar represents.

The range is expressed in arbitrary units that must be the same as the units used by **`computeHorizontalScrollExtent()`** (/reference/android/view/View#computeHorizontalScrollExtent()) and **`computeHorizontalScrollOffset()`** (/reference/android/view/View#computeHorizontalScrollOffset()).

The default range is the drawing width of this view.

### Returns

| | |
|---|---|
| **int** | the total horizontal range represented by the horizontal scrollbar |

## computeVerticalScrollExtent

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected int computeVerticalScrollExtent ()
```

Compute the vertical extent of the vertical scrollbar's thumb within the vertical range. This value is used to compute the length of the thumb within the scrollbar's track.

The range is expressed in arbitrary units that must be the same as the units used by **computeVerticalScrollRange()** (/reference/android/view/View#computeVerticalScrollRange()) and **computeVerticalScrollOffset()** (/reference/android/view/View#computeVerticalScrollOffset()).

The default extent is the drawing height of this view.

### Returns

| | |
|---|---|
| **int** | the vertical extent of the scrollbar's thumb |

## computeVerticalScrollRange

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected int computeVerticalScrollRange ()
```

Compute the vertical range that the vertical scrollbar represents.

The range is expressed in arbitrary units that must be the same as the units used by **`computeVerticalScrollExtent()`** (/reference/android/view/View#computeVerticalScrollExtent()) and **`computeVerticalScrollOffset()`** (/reference/android/view/View#computeVerticalScrollOffset()).

### Returns

| | |
|---|---|
| **`int`** | the total vertical range represented by the vertical scrollbar<br>The default range is the drawing height of this view. |

## drawableStateChanged

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected void drawableStateChanged ()
```

This function is called whenever the state of the view changes in such a way that it impacts the state of drawables being shown.

If the View has a StateListAnimator, it will also be called to run necessary state change animations.

Be sure to call through to the superclass when overriding this function.
If you override this method you *must* call through to the superclass implementation.

## getBottomPaddingOffset

Added in API level 2 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected int getBottomPaddingOffset ()
```

Amount by which to extend the bottom fading region. Called only when **`isPaddingOffsetRequired()`** (/reference/android/view/View#isPaddingOffsetRequired()) returns true.

### Returns

| | |
|---|---|
| **`int`** | The bottom padding offset in pixels. |

## getDefaultEditable

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected boolean getDefaultEditable ()
```

Subclasses override this to specify that they have a KeyListener by default even if not specifically called for in the XML options.

### Returns

| |
|---|
| **`boolean`** |

## getDefaultMovementMethod

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`protected MovementMethod (/reference/android/text/method/MovementMethod) getDefaultMovementMethod ()`

Subclasses override this to specify a default movement method.

### Returns

| | |
|---|---|
| **`MovementMethod`** (/reference/android/text/method/MovementMethod) | |

## getLeftFadingEdgeStrength

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

`protected float getLeftFadingEdgeStrength ()`

Returns the strength, or intensity, of the left faded edge. The strength is a value between 0.0 (no fade) and 1.0 (full fade). The default implementation returns 0.0 or 1.0 but no value in between. Subclasses should override this method to provide a smoother fade transition when scrolling occurs.

### Returns

| | |
|---|---|
| **`float`** | the intensity of the left fade as a float between 0.0f and 1.0f |

## getLeftPaddingOffset

Added in API level 2 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected int getLeftPaddingOffset ()
```

Amount by which to extend the left fading region. Called only when **`isPaddingOffsetRequired()`** (/reference/android/view/View#isPaddingOffsetRequired()) returns true.

### Returns

| | |
|---|---|
| **`int`** | The left padding offset in pixels. |

## getRightFadingEdgeStrength

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected float getRightFadingEdgeStrength ()
```

Returns the strength, or intensity, of the right faded edge. The strength is a value between 0.0 (no fade) and 1.0 (full fade). The default implementation returns 0.0 or 1.0 but no value in between. Subclasses should override this method to provide a smoother fade transition when scrolling occurs.

### Returns

| | |
|---|---|
| **`float`** | the intensity of the right fade as a float between 0.0f and 1.0f |

## getRightPaddingOffset

Added in API level 2 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected int getRightPaddingOffset ()
```

Amount by which to extend the right fading region. Called only when **`isPaddingOffsetRequired()`** (/reference/android/view/View#isPaddingOffsetRequired()) returns true.

### Returns

| | |
|---|---|
| **`int`** | The right padding offset in pixels. |

## getTopPaddingOffset

Added in API level 2 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected int getTopPaddingOffset ()
```

Amount by which to extend the top fading region. Called only when **`isPaddingOffsetRequired()`** (/reference/android/view/View#isPaddingOffsetRequired()) returns true.

### Returns

| | |
|---|---|
| **`int`** | The top padding offset in pixels. |

## isPaddingOffsetRequired

Added in API level 2 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected boolean isPaddingOffsetRequired ()
```

If the View draws content inside its padding and enables fading edges, it needs to support padding offsets. Padding offsets are added to the fading edges to extend the length of the fade so that it covers pixels drawn inside the padding. Subclasses of this class should override this method if they need to draw content inside the padding.

### Returns

| | |
|---|---|
| **boolean** | True if padding offset must be applied, false otherwise. |

## onAttachedToWindow

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected void onAttachedToWindow ()
```

This is called when the view is attached to a window. At this point it has a Surface and will start drawing. Note that this function is guaranteed to be called before **`onDraw(android.graphics.Canvas)`** (/reference/android/view/View#onDraw(android.graphics.Canvas)), however it may be called any time before the first onDraw -- including before or after **`onMeasure(int, int)`** (/reference/android/view/View#onMeasure(int,%20int)).
If you override this method you *must* call through to the superclass implementation.

## onConfigurationChanged

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected void onConfigurationChanged (Configuration (/reference/android/content/res/Configuration) newConfig)
```

Called when the current configuration of the resources being used by the application have changed. You can use this to decide when to reload resources that can changed based on orientation and other configuration characteristics. You only need to use this if you are not relying on the normal **`Activity`** (/reference/android/app/Activity) mechanism of recreating the activity instance upon a configuration change.

### Parameters

| | |
|---|---|
| **`newConfig`** | **`Configuration`**: The new resource configuration. |

## onCreateContextMenu

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected void onCreateContextMenu (ContextMenu (/reference/android/view/ContextMenu) menu)
```

Views should implement this if the view itself is going to add items to the context menu.

### Parameters

| | |
|---|---|
| **`menu`** | **`ContextMenu`**: the context menu to populate |

## onCreateDrawableState

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected int[] onCreateDrawableState (int extraSpace)
```

Generate the new **`Drawable`** (/reference/android/graphics/drawable/Drawable) state for this view. This is called by the view system when the cached Drawable state is determined to be invalid. To retrieve the current state, you should use **`getDrawableState()`** (/reference/android/view/View#getDrawableState()).

### Parameters

| | |
|---|---|
| **`extraSpace`** | **`int`**: if non-zero, this is the number of extra entries you would like in the returned array in which you can place your own states. |

### Returns

| | |
|---|---|
| **`int[]`** | Returns an array holding the current **`Drawable`** (/reference/android/graphics/drawable/Drawable) state of the view. |

## onDraw

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected void onDraw (Canvas (/reference/android/graphics/Canvas) canvas)
```

Implement this to do your drawing.

### Parameters

| | |
|---|---|
| **`canvas`** | **`Canvas`**: the canvas on which the background will be drawn |

## onFocusChanged

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected void onFocusChanged (boolean focused,
                int direction,
                Rect (/reference/android/graphics/Rect) previouslyFocusedRect)
```

Called by the view system when the focus state of this view changes. When the focus change event is caused by directional navigation, direction and previouslyFocusedRect provide insight into where the focus is coming from. When overriding, be sure to call up through to the super class so that the standard focus handling will occur.
If you override this method you *must* call through to the superclass implementation.

### Parameters

| | |
|---|---|
| **`focused`** | **`boolean`**: True if the View has focus; false otherwise. |
| **`direction`** | **`int`**: The direction focus has moved when requestFocus() is called to give this view focus. Values are **`View.FOCUS_UP`** (/reference/android/view/View#FOCUS_UP), **`View.FOCUS_DOWN`** (/reference/android/view/View#FOCUS_DOWN), **`View.FOCUS_LEFT`** (/reference/android/view/View#FOCUS_LEFT), **`View.FOCUS_RIGHT`** (/reference/android/view/View#FOCUS_RIGHT), **`View.FOCUS_FORWARD`** (/reference/android/view/View#FOCUS_FORWARD), or **`View.FOCUS_BACKWARD`** (/reference/android/view/View#FOCUS_BACKWARD). It may not always apply, in which case use the default. Value is **`View.FOCUS_BACKWARD`** (/reference/android/view/View#FOCUS_BACKWARD), **`View.FOCUS_FORWARD`** (/reference/android/view/View#FOCUS_FORWARD), **`View.FOCUS_LEFT`** (/reference/android/view/View#FOCUS_LEFT), **`View.FOCUS_UP`** (/reference/android/view/View#FOCUS_UP), **`View.FOCUS_RIGHT`** (/reference/android/view/View#FOCUS_RIGHT), or **`View.FOCUS_DOWN`** (/reference/android/view/View#FOCUS_DOWN) |

| | |
|---|---|
| **previouslyFocusedRect** | **Rect**: The rectangle, in this view's coordinate system, of the previously focused view. If applicable, this will be passed in as finer grained information about where the focus is coming from (in addition to direction). Will be **null** otherwise. This value may be **null**. |

## onLayout

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected void onLayout (boolean changed,
                int left,
                int top,
                int right,
                int bottom)
```

Called from layout when this view should assign a size and position to each of its children. Derived classes with children should override this method and call layout on each of their children.

### Parameters

| | |
|---|---|
| **changed** | **boolean**: This is a new size or position for this view |
| **left** | **int**: Left position, relative to parent |
| **top** | **int**: Top position, relative to parent |
| **right** | **int**: Right position, relative to parent |

| | |
|---|---|
| **bottom** | **int**: Bottom position, relative to parent |

## onMeasure

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected void onMeasure (int widthMeasureSpec,
                int heightMeasureSpec)
```

Measure the view and its content to determine the measured width and the measured height. This method is invoked by **measure(int, int)** (/reference/android/view/View#measure(int,%20int)) and should be overridden by subclasses to provide accurate and efficient measurement of their contents.

**CONTRACT:** When overriding this method, you *must* call **setMeasuredDimension(int, int)** (/reference/android/view/View#setMeasuredDimension(int,%20int)) to store the measured width and height of this view. Failure to do so will trigger an **IllegalStateException**, thrown by **measure(int, int)** (/reference/android/view/View#measure(int,%20int)). Calling the superclass' **onMeasure(int, int)** (/reference/android/view/View#onMeasure(int,%20int)) is a valid use.

The base class implementation of measure defaults to the background size, unless a larger size is allowed by the MeasureSpec. Subclasses should override **onMeasure(int, int)** (/reference/android/view/View#onMeasure(int,%20int)) to provide better measurements of their content.

If this method is overridden, it is the subclass's responsibility to make sure the measured height and width are at least the view's minimum height and width (**getSuggestedMinimumHeight()** (/reference/android/view/View#getSuggestedMinimumHeight()) and **getSuggestedMinimumWidth()** (/reference/android/view/View#getSuggestedMinimumWidth())).

### Parameters

| | |
|---|---|
| **widthMeasureSpec** | **int**: horizontal space requirements as imposed by the parent. The requirements are encoded with **View.MeasureSpec** |

(/reference/android/view/View.MeasureSpec).

| | |
|---|---|
| **heightMeasureSpec** | **int**: vertical space requirements as imposed by the parent. The requirements are encoded with **View.MeasureSpec** (/reference/android/view/View.MeasureSpec). |

## onScrollChanged

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected void onScrollChanged (int horiz,
                int vert,
                int oldHoriz,
                int oldVert)
```

This is called in response to an internal scroll in this view (i.e., the view scrolled its own contents). This is typically as a result of **scrollBy(int, int)** (/reference/android/view/View#scrollBy(int,%20int)) or **scrollTo(int, int)** (/reference/android/view/View#scrollTo(int,%20int)) having been called.

### Parameters

| | |
|---|---|
| **horiz** | **int**: Current horizontal scroll origin. |
| **vert** | **int**: Current vertical scroll origin. |
| **oldHoriz** | **int**: Previous horizontal scroll origin. |

| | |
|---|---|
| **oldVert** | **int**: Previous vertical scroll origin. |

## onSelectionChanged

Added in API level 3 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected void onSelectionChanged (int selStart,
                int selEnd)
```

This method is called when the selection has changed, in case any subclasses would like to know.

**Note:** Always call the super implementation, which informs the accessibility subsystem about the selection change.

If you override this method you *must* call through to the superclass implementation.

### Parameters

| | |
|---|---|
| **selStart** | **int**: The new selection start location. |
| **selEnd** | **int**: The new selection end location. |

## onTextChanged

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected void onTextChanged (CharSequence (/reference/java/lang/CharSequence) text,
                int start,
                int lengthBefore,
                int lengthAfter)
```

This method is called when the text is changed, in case any subclasses would like to know. Within **`text`**, the **`lengthAfter`** characters beginning at **`start`** have just replaced old text that had length **`lengthBefore`**. It is an error to attempt to make changes to **`text`** from this callback.

### Parameters

| | |
|---|---|
| **`text`** | **`CharSequence`**: The text the TextView is displaying |
| **`start`** | **`int`**: The offset of the start of the range of the text that was modified |
| **`lengthBefore`** | **`int`**: The length of the former text that has been replaced |
| **`lengthAfter`** | **`int`**: The length of the replacement modified text |

## onVisibilityChanged

Added in API level 8 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected void onVisibilityChanged (View (/reference/android/view/View) changedView,
                int visibility)
```

Called when the visibility of the view or an ancestor of the view has changed.

### Parameters

| | |
|---|---|
| **changedView** | **View**: The view whose visibility changed. May be **this** or an ancestor view. This value must never be **null**. |
| **visibility** | **int**: The new visibility, one of **View.VISIBLE** (/reference/android/view/View#VISIBLE), **View.INVISIBLE** (/reference/android/view/View#INVISIBLE) or **View.GONE** (/reference/android/view/View#GONE). Value is **View.VISIBLE** (/reference/android/view/View#VISIBLE), **View.INVISIBLE** (/reference/android/view/View#INVISIBLE), or **View.GONE** (/reference/android/view/View#GONE) |

## setFrame

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected boolean setFrame (int l,
                int t,
                int r,
                int b)
```

### Parameters

| | |
|---|---|
| **l** | **int** |

| | |
|---|---|
| `t` | `int` |
| `r` | `int` |
| `b` | `int` |

### Returns

`boolean`

## verifyDrawable

Added in API level 1 (/guide/topics/manifest/uses-sdk-element#ApiLevels)

```
protected boolean verifyDrawable (Drawable (/reference/android/graphics/drawable/Drawable) who)
```

If your view subclass is displaying its own Drawable objects, it should override this function and return true for any Drawable it is displaying. This allows animations for those drawables to be scheduled.

Be sure to call through to the super class when overriding this function.
If you override this method you *must* call through to the superclass implementation.

### Parameters

| | |
|---|---|
| **who** | **`Drawable`**: This value must never be **`null`**. |

## Returns

| | |
|---|---|
| **`boolean`** | boolean If true than the Drawable is being displayed in the view; else false and it is not allowed to animate. |



Last updated 2020-02-12.

# Exhibit 34

# THIS EXHIBIT HAS BEEN REDACTED IN ITS ENTIRETY

# Exhibit 35

☆☆☆☆☆

# Frequently Asked Questions

This page provides answers to some frequently asked questions (FAQs).



## Open Source

### What is the Android Open Source Project?

Android Open Source Project (AOSP) refers to the people, processes, and source code that make up Android.

The people oversee the project and develop the source code. The processes are the tools and procedures that we use to manage the development of the software. The net result is the source code, which you can use in mobile phones and other devices.

### Why did we open the Android source code?

Google started the Android project in response to our own experiences launching mobile apps. We wanted to make sure there would always be an open platform available for carriers, OEMs, and developers to use to make their innovative ideas a reality. We also wanted to avoid any central point of failure, so no single industry player could restrict or control the innovations of any other. Our single most important goal with AOSP is to make sure that open source Android software is implemented as widely and compatibly as possible, to everyone's benefit.

### What kind of open source project is Android?

Google oversees the development of the core Android open source platform and works to create robust developer and user communities. For the most part, the Android source code is licensed under the permissive Apache License 2.0, rather than a *copyleft* license. We chose the Apache 2.0 license because we believe that it encourages widespread Android software adoption. For details, see Licenses.

### Why is Google in charge of Android?

Launching a software platform is complex. Openness is vital to the long-term success of a platform, because openness attracts investment from developers and ensures a level playing field. The platform must also be a compelling product to users.

Google has committed the professional engineering resources necessary to ensure that Android is a fully competitive software platform. Google treats the Android project as a full-scale product development operation and strikes the business deals necessary to make sure great devices running Android make it to market.

By making sure Android is a success with users, we help ensure the vitality of Android as a platform and as an open source project. After all, who wants the source code to an unsuccessful product?

Google's goal is to ensure a successful ecosystem around Android. We opened the Android source code so that anyone can modify and distribute the software to meet their own needs.

### What is Google's overall strategy for Android product development?

We release great devices into a competitive marketplace. We then incorporate the innovations and enhancements we made into the core platform as the next version.

In practice, this means that the Android engineering team focuses on a small number of "flagship" devices and develops the next version of Android software to support those product launches. These flagship devices absorb much of the product risk and blaze a trail for the broad OEM community w

flagship devices absorb much of the p[illegible] and blaze a trail for the broad OEM community, who follow up with more devices that take advantage of the new features. In this way, we make sure that the Android platform evolves according to the needs of real-world devices.

### How is Android software developed?

Each platform version of Android (such as 1.5 or 8.1) has a corresponding branch in the open source tree. The most recent branch is considered the *current stable* branch version. This is the branch that manufacturers port to their devices. This branch is kept suitable for release at all times.

Simultaneously, there's a *current experimental* branch, which is where speculative contributions, such as large next-generation features, are developed. Bug fixes and other contributions can be included in the current stable branch from the experimental branch as appropriate.

Finally, Google works on the next version of the Android platform in tandem with developing a flagship device. This branch pulls in changes from the experimental and stable branches as appropriate.

For details, see Codelines, Branches, and Releases.

### Why are parts of Android developed in private?

It typically takes more than a year to bring a device to market. And, of course, device manufacturers want to ship the latest software they can. Meanwhile, developers don't want to constantly track new versions of the platform when writing apps. Both groups experience a tension between shipping products and not wanting to fall behind.

To address this, some parts of the next version of Android including the core platform APIs are developed in a private branch. These APIs constitute the next version of Android. Our aim is to focus attention on the current stable version of the Android source code while we create the next version of the platform. This allows developers and OEMs to use a single version without tracking unfinished future work just to keep up. Other parts of the Android system that aren't related to application compatibility are developed in the open. It's our intention to move more of these parts to open development over time.

### When are source code releases made?

When they're ready. Releasing the source code is a fairly complex process. Some parts of Android are developed in the open, and that source code is always available. Other parts are developed first in a private tree, and that source code is released when the next platform version is ready.

In some releases, core platform APIs are ready far enough in advance so that we can push the source code out for an early look prior to the device's release. In other releases, this isn't possible. In all cases, we release the platform source when we feel that the version is stable, and when the development process permits.

### What's involved in releasing the source code for a new Android version?

Releasing the source code for a new version of the Android platform is a significant process. First, the software is built into a system image for a device and put through various forms of certification, including government regulatory certification for the regions the phones will be deployed. The code also goes through operator testing. This is an important phase of the process, because it helps detect software bugs.

When the release is approved by the regulators and operators, the manufacturer begins mass producing devices, and we begin releasing the source code.

Simultaneous to mass production, the Google team kicks off several efforts to prepare the open source release. These efforts include making final API changes, updating documentation (to reflect any modifications that were made during qualification testing, for example), preparing an SDK for the new version, and launching the platform compatibility information.

Our legal team does a final sign-off to release the code into open source. Just as open source contributors are required to sign a Contributors License Agreement attesting to their intellectual property ownership of their contribution, Google must verify that the source is cleared to make contributions.

From the time that mass production begins, the software release process usually takes about a month, so source code releases often happen at around the same time the devices reach users.

### How does AOSP relate to the Android Compatibility Program?

The Android Open Source Project maintains Android software, and develops new versions. Because

it's open source, this software can be used for any purpose, including developing devices that aren't compatible with other devices based on the same source.

The function of the Android Compatibility Program is to define a baseline implementation of Android that is compatible with third-party apps written by developers. Devices that are *Android compatible* may participate in the Android ecosystem, including Google Play; devices that don't meet the compatibility requirements exist outside of that ecosystem.

In other words, the Android Compatibility Program is how we separate Android-compatible devices from devices that merely run derivatives of the source code. We welcome all uses of the Android source code, but to participate in the Android ecosystem, a device must be identified as Android-compatible by the program.

### How can I contribute to Android?

You can report bugs, write apps for Android, or contribute source code to the Android Open Source Project.

There are limits to the kinds of code contributions we accept. For instance, someone might want to contribute an alternative application API, such as a full C++-based environment. We would decline that contribution, because Android encourages applications to be run in the ART runtime. Similarly, we won't accept contributions such as GPL or LGPL libraries that are incompatible with our licensing goals.

We encourage those interested in contributing source code to contact us via the channels listed on the Android Community page prior to beginning any work. For details, see Contributing.

### How do I become an Android committer?

The Android Open Source Project doesn't really have a notion of a *committer*. All contributions (including those authored by Google employees) go through a web-based system known as Gerrit that's part of the Android engineering process. This system works in tandem with the git source code management system to cleanly manage source code contributions.

When submitted, changes need to be accepted by a designated approver. Approvers are typically Google employees, but the same approvers are responsible for all submissions, regardless of origin.

For details, see Submitting Patches.

Back to top

## Compatibility

### What is Android "compatibility"?

We define an *Android-compatible device* as one that can run any application written by third-party developers using the Android SDK and NDK. We use this as a filter to separate devices that can participate in the Android app ecosystem and those that can't. For devices that are properly compatible, device manufacturers can seek approval to use the Android trademark. Devices that aren't compatible are merely derived from the Android source code and may not use the Android trademark.

In other words, compatibility is a prerequisite to participate in the Android apps ecosystem. Anyone is welcome to use the Android source code. But if the device isn't compatible, it isn't considered part of the Android ecosystem.

### What's the role of Google Play in compatibility?

Device manufacturers with Android-compatible devices may seek to license the Google Play client software. Licensed devices become part of the Android app ecosystem, enabling their users to download developers' apps from a catalog shared by all compatible devices. Licensing isn't available to incompatible devices.

### What kinds of devices can be Android compatible?

Android software can be ported to many different devices, including some on which third-party apps won't run properly. The Android Compatibility Definition Document (CDD) spells out the specific device configurations that are considered compatible.

For example, though the Android source code could be ported to run on a phone that doesn't have a camera, the CDD requires all phones to have a camera. This allows developers to rely on a consistent set of capabilities when writing their apps.

The CDD continues to evolve to reflect market realities. For instance, version 1.6 of the CDD supports only cell phones. But version 2.1 allows devices to omit telephony hardware, enabling non-phone devices such as tablet-style music players to be compatible. As we make these changes, we'll also augment Google Play to allow developers to retain control over where their apps are available. To continue the telephony example, an app that manages SMS text messages isn't useful on a media player, so Google Play allows the developer to restrict that app exclusively to phone devices.

### If my device is compatible, does it automatically have access to Google Play and branding?

No. Access isn't automatic. Google Play is a service operated by Google. Achieving compatibility is a prerequisite for obtaining access to the Google Play software and branding. After a device is qualified as an Android-compatible device, the device manufacturer should complete the contact form included in licensing Google Mobile Services to seek access to Google Play. We'll be in contact if we can help you.

### If I'm not a manufacturer, how can I get Google Play?

Google Play is only licensed to handset manufacturers shipping devices. For questions about specific cases, contact android-partnerships@google.com.

### How can I get access to Google apps for Android, such as Maps?

Google apps for Android, such as YouTube, Google Maps, and Gmail are Google properties that aren't part of Android and are licensed separately. Contact android-partnerships@google.com for inquiries related to these apps.

### Is compatibility mandatory?

No. The Android Compatibility Program is optional. The Android source code is open, so anyone can use it to build any kind of device. However, if manufacturers wish to use the Android name with their products, or want access to Google Play, they must first demonstrate that their devices are compatible.

### How much does compatibility certification cost?

There's no cost to obtain Android compatibility for a device. The Compatibility Test Suite is open source and available to anyone for device testing.

### How long does compatibility take?

The process is automated. The Compatibility Test Suite generates a report that can be provided to Google to verify compatibility. Eventually we intend to provide self-service tools to upload these reports to a public database.

### Who determines the compatibility definition?

Google is responsible for the overall direction of Android as a platform and product, so Google maintains the Compatibility Definition Document (CDD) for each release. We draft the CDD for a new Android version in consultation with various OEMs who provide input.

### How long will each Android version be supported for new devices?

Android's code is open source, so we can't prevent someone from using an old version to launch a device. Instead, Google chooses not to license the Google Play client software for use on versions that are considered obsolete. This allows anyone to continue to ship old versions of Android, but those devices won't use the Android name and exist outside of the Android apps ecosystem, just as if they weren't compatible.

### Can a device have a different user interface and still be compatible?

The Android Compatibility Program determines whether a device can run third-party applications. The user interface components shipped with a device (such as home screen, dialer, and color scheme) don't generally have much effect on third-party apps. As such, device builders are free to customize the user interface. The Compatibility Definition Document restricts the degree to which OEMs may alter the system user interface for areas that impact third-party apps.

### When are compatibility definitions released for new Android versions?

Our goal is to release a new version of the Android Compatibility Definition Document (CDD) when the corresponding Android platform version has converged enough to permit it. While we can't release a final draft of a CDD for an Android software version before the first flagship device ships with that software, final CDDs are always released after the first device. However, wherever practical we release draft versions of CDDs.

### How are device manufacturers' compatibility claims validated?

There is no validation process for Android device compatibility. However, if the device is to include Google Play, Google typically validates the device for compatibility before agreeing to license the Google Play client software.

### What happens if a device that claims compatibility is later found to have compatibility problems?

Typically, Google's relationships with Google Play licensees allow us to ask the deveice manufacturer to release updated system images that fix the problems.

Back to top

## Compatibility Test Suite

### What's the purpose of the CTS?

The Compatibility Test Suite is a tool used by device manufacturers to help ensure that their devices are compatible, and to report test results for validations. The CTS is intended to be run frequently by OEMs throughout the engineering process to catch compatibility issues early.

### What kinds of things does the CTS test?

The CTS currently tests that all of the supported Android strong-typed APIs are present and behave correctly. It also tests other non-API system behaviors such as application lifecycle and performance. We plan to add support in future CTS versions to test *soft* APIs such as Intents.

### Will the CTS reports be made public?

Yes. While not currently implemented, Google intends to provide web-based self-service tools for OEMs to publish CTS reports so that anyone can view them. Manufacturers can share CTS reports with as wide an audience as they like.

### How is the CTS licensed?

The CTS is licensed under the same Apache Software License 2.0 that the bulk of Android uses.

### Does the CTS accept contributions?

Yes, please! The Android Open Source Project accepts contributions to improve the CTS just as for any other component. In fact, improving the coverage and quality of the CTS test cases is one of the best ways to help Android.

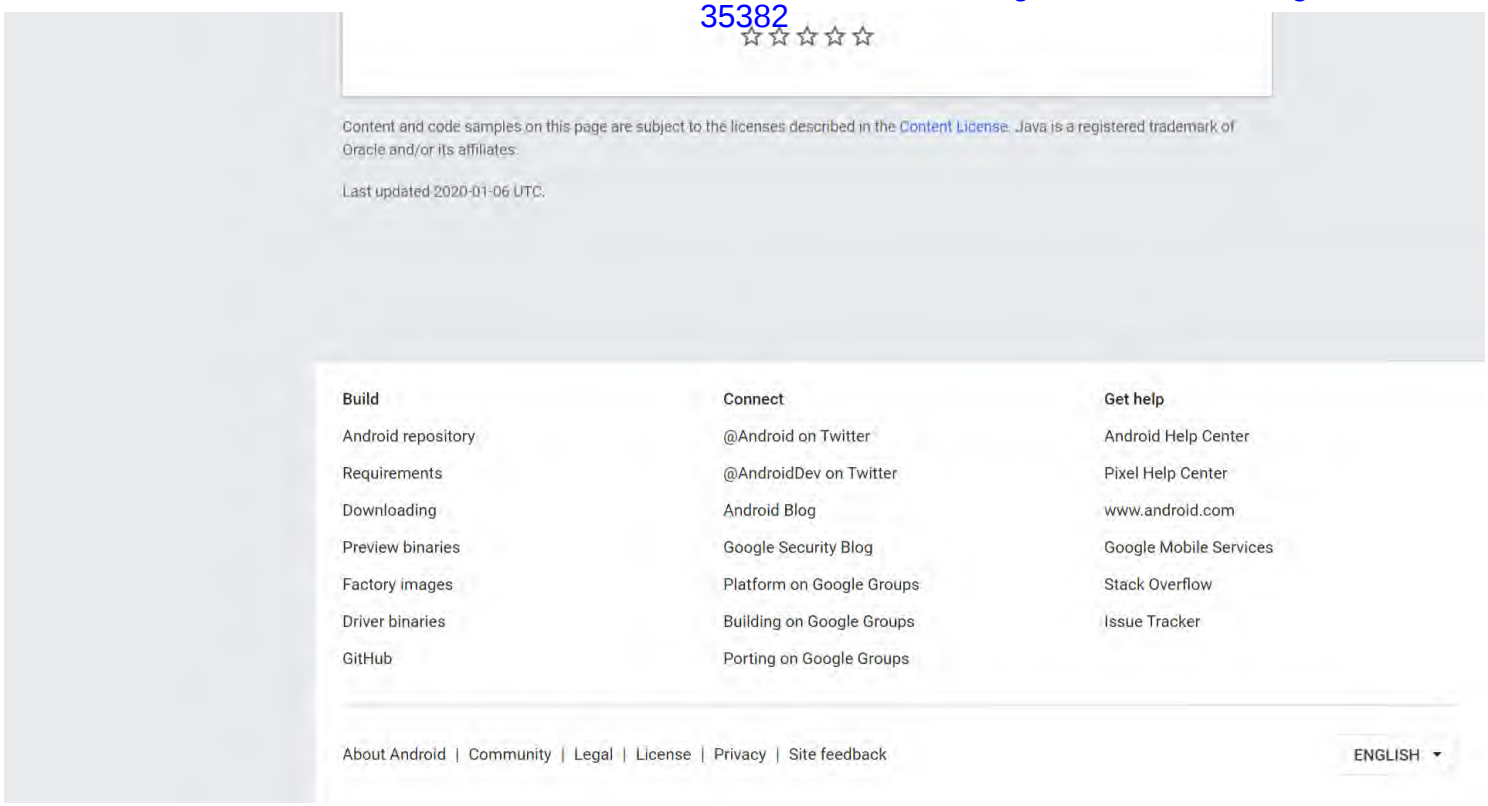
### Can anyone use the CTS on existing devices?

The Compatibility Definition Document requires that compatible devices implement the `adb` debugging utility. This means that any compatible device (including those available at retail) must be able to run the CTS tests.

### Are codecs verified by CTS?

Yes. All mandatory codecs are verified by CTS.

Back to top


Content and code samples on this page are subject to the licenses described in the Content License. Java is a registered trademark of Oracle and/or its affiliates.
Last updated 2020-01-06 UTC.
Build
Android repository
Requirements
Downloading
Preview binaries
Factory images
Driver binaries
GitHub
Connect
@Android on Twitter
@AndroidDev on Twitter
Android Blog
Google Security Blog
Platform on Google Groups
Building on Google Groups
Porting on Google Groups
Get help
Android Help Center
Pixel Help Center
www.android.com
Google Mobile Services
Stack Overflow
Issue Tracker
About Android | Community | Legal | License | Privacy | Site feedback
ENGLISH

# Exhibit 36

ELSEVIER

Knowledge-Based Systems 11 (1998) 3–13

Knowledge-Based SYSTEMS

# CyberDesk: a framework for providing self-integrating context-aware services

Anind K. Dey[a,*], Gregory D. Abowd[a], Andrew Wood[b]

[a]Graphics, Visualization and Usability Center, Georgia Institute of Technology, Atlanta, GA 30332-0280, USA
[b]School of Computer Science, The University of Birmingham, Edgbaston, Birmingham, B15 2TT, UK

Received 19 May 1998; accepted 1 June 1998

Abstract

Applications are often designed to take advantage of the potential for integration with each other via shared information. Current approaches for integration are limited, affecting both the programmer and end-user. In this paper, we present CyberDesk, a framework for self-integrating software in which integration is driven by user context. It relieves the burden on programmers by removing the necessity to predict how software should be integrated. It also relieves the burden from users by removing the need to understand how to make different software components work together. © 1998 Elsevier Science B.V. All rights reserved.

*Keywords:* Context-aware computing; Automated software integration; Dynamic mediation; Ubiquitous computing

## 1. Introduction

Software applications often work on similar information types such as names, addresses, dates, and locations. Collections of applications are often designed to take advantage of the potential for integration via shared information. As an example, an electronic mail reader can be enhanced to automatically recognize Web addresses, allowing a reader to select a URL to automatically launch a Web browser on that location. Even more complex and useful integrating behavior is available in a number of commercial suites of applications (e.g. Microsoft Office 97, Lotus SmartSuite, WordPerfect Suite, and Netscape Communicator).

We recognize the utility from the user's perspective of integrating the behavior of a number of software applications. With the emergence of Web-based applications and personal digital assistants (PDAs), there are even more opportunities to provide integration of software applications. There are some limitations, however, to the current approaches for providing this integration. These limitations impact both the programmer and the user.

From the programmer's perspective, the integrating behavior between applications is static. That is, this behavior must be identified and supported when the applications are built. This means that a programmer has the impossible task of predicting all of the possible ways users will want a given application to work with all other applications. This usually results in there being a limited number of software applications available in an application suite, with limited integration behavior.

From the user's perspective, integrating behavior is limited to the applications that are bound to the particular suite being used. Further integration is either impossible to obtain or must be implemented by the user. In addition, the integrating behavior has a strong dependence on the individual applications in the suite. If a user would like to substitute a comparable application for one in the suite (e.g. use a different contact manager, or word processor), the user does so at the risk of losing all integrating behavior.

The project described in this paper, CyberDesk, is aimed at providing a more flexible framework for integrating software behavior. We aim to reduce the programming burden in identifying and defining integrating behavior, while at the same time retaining as much user freedom in determining how integration is to occur. The main objective of the ubiquitous computing project, CyberDesk, is to provide the infrastructure for self-integrating software in which the integration is driven by user actions. We refer to this as context-aware integration, and it is aimed at producing a paradigm shift in human–computer interactions that is fundamental to ubiquitous computing. Rather than settle for the current situation, in which the user must seek out and find relevant software functionality when it is wanted, we instead want the ubiquitous computing infrastructure to seek out the user when and where it is wanted. In this paper,

* Corresponding author. Tel.: +1 404 894 5103; fax: +1 404 894 2970; e-mail: anind, abowd@cc.gatech.edu


PII: S0950-7051(98)00053-7

we demonstrate how CyberDesk supports this paradigm shift.

First, we describe CyberDesk and motivate our research with a sample scenario. In Section 3, we describe CyberDesk's architecture and show how it supports context-aware computing. This is followed with a discussion on how CyberDesk can be extended to include more applications and integrating behavior. Finally, we summarize the contributions of this research and discuss some future directions.

## 2. What is CyberDesk?

CyberDesk is a component-based framework written in Java, that supports the automatic integration of software applications. The framework is flexible, and can be easily customized and extended. It is designed to allow ubiquitous access to services and data, regardless of whether they came from a desktop application, a PDA-based application, or a Web-based application.

CyberDesk is able to provide this ubiquitous access by having applications automatically provide their services to the user. Rather than displaying all the available services to the user at all times, the interface is limited to displaying those services that are relevant to the user's current context. We define a service to be an action that an application can perform. A user's context is any information about the user and the environment that can be used to enhance the user's experiences. This includes the data the user is working with, the time of day, the user's physical location, and user's emotional state, social environment, objects in the room, etc. Initially, CyberDesk was able to work only with simple strings that a user was working with in a desktop application. Now, CyberDesk is also able to work with time and location in a desktop environment, networked environment, and mobile environment.

Desktop applications incorporated into CyberDesk include e-mail browsers, notepads, schedulers, and contact managers. Network applications include phone number lookups, e-mail writing, mailing address lookups, Web searches, Usenet searches, e-mail address lookups, map lookups, and Web page browsing. PDA-based applications include contact managers and notepads. All applications make their services available to the user via a common interface. The services available at any particular time depend on the user's context at that time. By providing relevant suggestions and data to the user, CyberDesk gives the user useful, and possibly unexpected, help in completing their tasks.

### 2.1. User scenario

To illustrate this behavior, an actual user experience follows.[1] As seen in Fig. 1, a user is checking his e-mail,

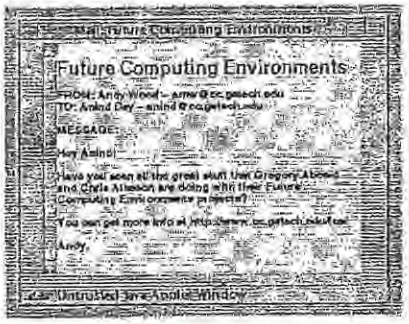



Fig. 1. Content of user's e-mail message.

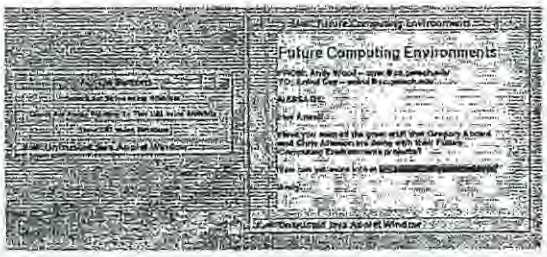



Fig. 2. User selects the URL and is offered suggestions.

and reads a message about some interesting research. The user is interested in the research discussed and highlights the URL in the message. CyberDesk offers the following service suggestions through its interface (Fig. 2): search for the selected text using AltaVista, find pages that reference this URL using AltaVista, and display the URL in Netscape.

The user chooses the last option and views the selected URL in a web browser (Fig. 3).

The user wants to get more information, so selects the name of the person in charge of the research. CyberDesk offers the following suggestions (Fig. 4): search for the selected text using AltaVista, search for a phone number and mailing address for the selected name using Switchboard, lookup the selected name in the contact manager. The user wants to contact this researcher so checks to see if the name is in the contact manager, but it is not.

So, the user selects the phone number and mailing address lookup service (Fig. 5), and then creates a new entry in the contact manager with this new information.

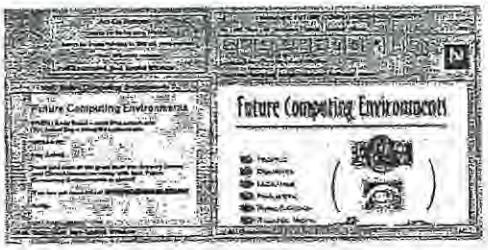



Fig. 3. CyberDesk executes the service and displays the URL.

[1] All the scenarios described in the paper can be executed at http://www.cc.gatech.edu/fce/cyberdesk/ui

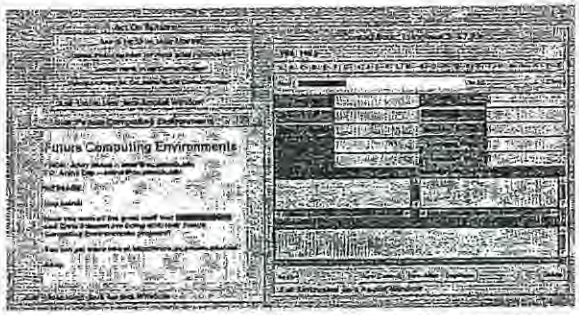



Fig. 4. User selects name and chooses the Contact lookup service.

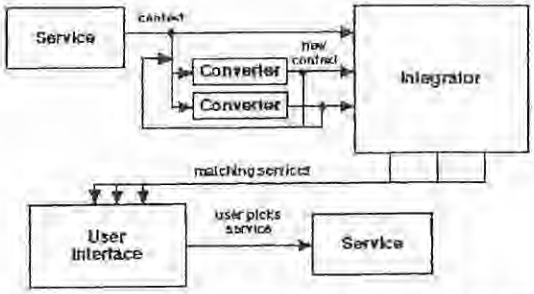



Fig. 6. The run-time architecture of CyberDesk. Arrows indicate the flow of information a control in the system.

## 3. Architecture

The CyberDesk system has a simple architecture, based on the previously developed CAMEO infrastructure [1]. CAMEO defines a component-based framework in which individual components can observe the activities of other components and manipulate their interfaces. A centralized service allows for the dynamic registration of components and run-time support for querying the interfaces of registered components. Observation and manipulation of other components and the dynamic registry of CAMEO were sufficient motivation for us to port it to Java and take advantage of simpler cross-platform network access to a multitude of Web-based, mobile and desktop information services.

The CyberDesk system consists of five main components: the Registry, the information services, the type converters, the Integrator, and the user interface. The Registry maintains a list of components in the system and the interfaces that each supports. The information services are the tools and functions the user ultimately wants to use, such as an e-mail browser, a contact manager, or a Web-based search engine. These services register their interfaces with the Registry and announce events that provide data/information to the rest of the system (e.g. the name selected in the e-mail message in the scenario). The type converters accept announced data from the system and convert it recursively to other forms of data that may be useful (e.g. a string being converted to a URL). The Integrator uses the Registry to automatically find matches between user data and the services that can use that data, a task that would normally be performed by the system designer. The matched services are then displayed to the user through the user interface.

The run-time relationship between the components from the user's perspective is depicted in Fig. 6. The components are described in greater detail below.

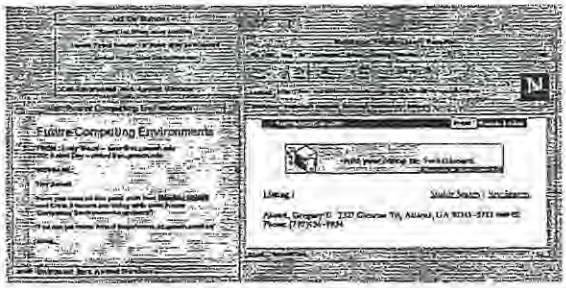



Fig. 5. User selects the phone number lookup service.

### 3.1. Registry

The Registry maintains a directory of all the other components in the system: what interfaces they can support and what data they can provide, if any. As each component joins the CyberDesk system, it provides this information to the Registry. Some components, upon registration, tell the Registry that they are interested in other components. Hereafter, whenever a component joins or leaves the system, the Registry notifies these interested components. The Registry also provides both a white pages and a yellow pages service. When queried, the Registry acts as a white pages directory by supplying information (object reference and interfaces) on individual components. It can also act as a yellow pages directory by providing a list of components that support a particular interface.

### 3.2. Services

Services are end-user function calls that perform actions on supplied data. Services can be stand-alone or part of a larger application. Examples of stand-alone services are network-based Web CGI scripts such as finding a phone number and address using Switchboard or searching for some text in AltaVista. Examples of services in larger applications are creating an entry or searching for a name in a contact manager, or loading a schedule in a day planner.

Service wrappers are used to integrate existing services into CyberDesk. These wrappers adapt the interfaces of the existing services to conform to the CyberDesk registration and communication requirements. They make the functionality of the services accessible to other components, and provide methods for communicating with other components and registering their interfaces with the Registry.

Services not only provide functionality to the user, but they can also provide data to the system, as seen in the

previous scenario. When users select data with a mouse in an application, that data is observed by interested components (a subset of the type converters and the Integrator described below). This data is the contextual information used by CyberDesk, as its origin is some user activity. The author of the service wrapper determines what information and functionality is made available to the CyberDesk system.

### 3.3. Type converters

Type converters are components that take data in the system and attempt to convert it to other forms of data. They use simple techniques to provide complex and intelligent-like behaviors to the system. For example, the scenario showed an example of a conversion from a string to a name. Data in the system can come from actions other than selection with a mouse. For example, position services provide location information such as coordinates within a space. Type converters can be used to convert these coordinates to a room within a building. This allows the user to see services that are only available within a particular room. Type converters create additional data types to match services against.

The type converters provide a separable context-inferencing engine with arbitrary power. As the conversion abilities of the converters improve, the ability of the system to make relevant service suggestions also improves. Therefore, the apparent intelligence of CyberDesk is also contained within the type converters. Since the type converters are represented as a collection of Java classes, it is a simple matter to boost the overall power of this context-inferencing engine without impacting any of the functionality of the rest of the system.

As mentioned in the section on the Registry, some components are interested in the addition and removal of other components. Type converters are an example of this. They monitor which services are added and removed from the system, so they can determine which components can provide data, and can then observe these components.

### 3.4. Integrator

The Integrator also observes components that can provide data. It uses this information to find services that can act on the data. In the user scenario, when the user selected a name in the e-mail message, both a string and a name (via a type converter) were made available to the system. The Integrator took that data and found services that could act on both strings and names.

When components register or remove themselves from the Registry, the Integrator is notified. The Integrator uses this information to update its list of components that can act on various types of data. For example, when the Switchboard service is added to CyberDesk at run-time, it registers that it can perform a function on name information. The Registry notifies all components interested in the addition and removal of components: type converters and the Integrator. The Integrator contacts the Registry to determine the kind of interface the Switchboard service supports and finds out that it can act on name data. When name data enters the system, the Integrator makes the Switchboard service available to the user.

### 3.5. User interface

When the Integrator finds matching services for the data it has observed, it makes these services available to the user. We have experimented with creating buttons on a separate window to display the suggested services to the user, as shown in Fig. 1. Each button is associated with a service and the data the service can act on. When a user clicks on a button, the service is executed with this data.

The user interface, like the other components, is completely interchangeable. If the provided user interface does not meet with the user's approval, it can be easily replaced by another user interface that better informs the user of the connection to the current context and suggestions for future actions based on that context.

## 4. Adding applications to CyberDesk

As discussed earlier, the applications used in CyberDesk are the actual tools the user wants to use. CyberDesk provides an easier and faster way for users to access the functionality of these applications and the data they contain (i.e. their services). From the user's perspective, adding an application (or any CyberDesk component) to CyberDesk simply requires the addition of HTML applet tags to a CyberDesk Web page. From the programmer's perspective, adding an application requires a little more effort.

Currently, CyberDesk is unable to automatically determine the services an application provides. A service programmer must construct a wrapper around each application. This wrapper performs two main functions [2]: registration of the provided services with CyberDesk and execution of the services when called. During the registration process, each application registers with the Integrator giving a list of services it provides: both actions it can perform on different data types and the data types it can produce. Examples of this from the user scenario are:

- the AltaVista wrapper declaring it can search the Web for a string and find pages that reference a given URL;
- the Contact Manager wrapper declaring it can lookup a given name, create a new entry, and can produce string objects when a user selects data in the contact manager.

The second portion of the wrapper deals with actually executing the services that were registered. When the user selects a service from the interface, a method in the wrapper is called to execute the service. This method takes the user's

context, retrieves the relevant parameters for the service and calls a method that will execute the user-selected service.

All the Web-based applications employed by CyberDesk use HTML forms. By analyzing the form, a programmer can easily generate the service wrapper. These applications generally have straightforward interfaces and require a small set of input parameters. The parameters are passed to a URL, which generates a resulting HTML page that can be displayed in a Web browser. For example, the AltaVista Web search service simply requires an input string and returns a list of all Web pages that match this string.

A service writer program has been written to automatically generate a wrapper for Web-based applications. This program is intended for use by service programmers, but is simple enough to be used by an end-user wanting to add a service to CyberDesk. The program takes a URL containing a form as input and presents an interface, as shown in Fig. 7.

The service programmer selects the data type the service can act on, the values for the service parameters, and the output data type, if any. Upon receiving this information, the service writer program generates a wrapper for the service.

Other Georgia Tech students have written all of the desktop applications added to CyberDesk. This allows us access to both the application programming interfaces (APIs) and the source code. The APIs were needed to determine the names of methods for services the applications provided and the parameters that each method required. The source code allowed the service programmers to add additional services to the applications and add the data selection ability shown in the original scenario (e.g. user selecting the URL). It should be made clear that source code was only modified to enhance the existing applications. If only the API were available for an application, and not the source code, a wrapper could still be written to take advantage of the application's existing functionality.

In addition, an automated service writer, similar to the one mentioned, is being designed for applications that are not Web-based. The service writer will be based on the newest release of the Java language (version 1.1). It provides an automatic data selection feature, allowing for the transfer of data between (Java and non-Java) applications via a clipboard-style interface, and supports the use of reusable software components called JavaBeans [3]. The data transfer feature will eliminate the need for any application source code. The use of JavaBeans would allow the service writer to query an application and determine its API at run-time, removing the need for a compile-time API and allowing the use of third-party applications.

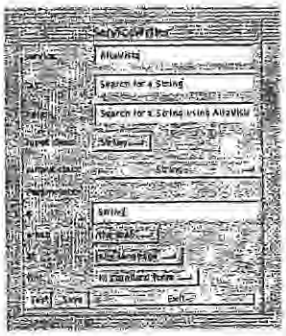



Fig. 7. Service writer interface.

We have tested the scalability of CyberDesk by adding more and more services and context types. Standard desktop applications currently included in the CyberDesk prototype include two e-mail browsers, a calendar, a scheduler, a contact manager, and a notepad. Currently, there are over 70 Web-based applications that have been integrated into CyberDesk.

### 4.1. Case study: accessing mobile data

LlamaShare [4], a research project at Georgia Tech, is an architecture and set of applications that provide users and programmers easy access to information stored on mobile devices. There are two main goals for the LlamaShare project. The first is to create an infrastructure that makes simple for programmers to take advantage of mobile data in their applications. The second is to provide applications that demonstrate ubiquitous access to information, a goal shared by CyberDesk. As a test of CyberDesk's extensibility, we integrated the LlamaShare project with CyberDesk.

Currently, it is very difficult to retrieve information from a mobile device (a PDA like a Newton, for example) both for programmers and for users. From a user's perspective, it is also very difficult to deal with information stored on a mobile device. The current method of accessing this data is typically through a 'synchronization' process, which does a reasonable job of copying the data to a user's desktop machine, but does nothing to aid the users in actually doing anything with that information, such as integrating relevant pieces in their daily tasks. The LlamaShare infrastructure, consisting of a central server called the LlamaServer, provides routing for information requests between any mobile device on the network (wired or wireless) and any desktop machine on the Internet.

There were two reasons for integrating CyberDesk with LlamaShare. First, we wanted to illustrate the platform-neutrality and language-neutrality of the LlamaServer, which CyberDesk allows us to do. More importantly, however, CyberDesk's vision of ubiquitous information access was the deciding factor. While LlamaShare provides a concrete, visible object to represent the data on a mobile device, CyberDesk takes the approach that information is distributed throughout a rather nebulous space (consisting of Internet, desktop, and mobile data) that can be retrieved at any moment depending on the context in which a user is currently working. This new metaphor of seamless integration between mobile data and Internet (remote) data that CyberDesk supports, was a good match with LlamaShare.

Adding services and viewers to CyberDesk was quite simple. CyberDesk already supported the most common

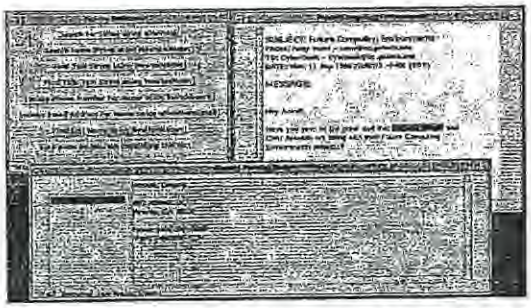


Fig. 8. Screenshot of LlamaShare being used in CyberDesk. The user selects a name (a) in the e-mail tool. CyberDesk offers a number of integrating suggestions (b), including four that access data from a remote Newton. The user chooses the second suggestion and sees the results (c), obtained from a remote Newton.

data types that users would be interested in on their PDA (text, names, phone numbers, and dates), so nothing new needed to be added.

The next task involved adding the services that recognized the appropriate data types and created appropriate user actions for them. We added two services (NewtonNames and NewtonNotes) which request contact information from a Newton about a selected name and request notes from the Newton containing selected text in the body or title. The results of the integration between CyberDesk and LlamaShare can be seen in Fig. 8.

Here are some other examples of how we are using LlamaShare and CyberDesk.

- A user is writing an e-mail and needs to retrieve some relevant text from the Newton. The user selects a keyword and searches Newton for all the notes containing that text and chooses the appropriate one for inclusion in the message.
- A user receives an e-mail message from a colleague and wants to call that colleague back. The user has the sender's phone number on Newton, so simply selects the sender's name and retrieves the contact entry from the Newton.
- While a user is in a meeting, his/her assistant takes down the number of someone who called, but not the caller's name. The user can select the phone number and let the Newton search its name database. The Newton returns the name of the person/company with that phone number.
- A user needs to schedule a meeting with two other colleagues. To find a time when all three parties are available, the user can select each of their names and lets CyberDesk display their calendars. The user can schedule the meeting and make the change effective immediately in the colleagues' Newton calendars.

## 5. Adding type converters to CyberDesk

CyberDesk applications can generate changes in the data the user is working with. As described above, CyberDesk uses type converters to convert this user context (or location or time information, as will be shown in an upcoming section) into other useful forms of user context. For example, in the user scenario, a StringToURL converter took the data selected by the user and successfully converted it to a URL. This resulted in two pieces of data being sent to the Integrator, a string and a URL. The Integrator sought integrating behavior for both these types, allowing the user to access URL-relevant services where originally they would not have had the option.

The type converters work in a recursive fashion. That is, the new data is generated from a successful conversion is sent to the type converters. This process continues until no new data is created, or a cycle is found.

Initially, applications were hardcoded to generate different data types. For example, the e-mail browser declared that it could generate strings when text is highlighted, but also EmailAddress objects when the 'To:' or 'From:' field in an e-mail message was selected. When EmailAddress objects were generated, they were passed through the CyberDesk system, as described before, to the user interface, which displayed services that could consume EmailAddress objects (e.g. send an e-mail message to this e-mail address using Netscape). However, this required the applications themselves to be aware of the CyberDesk type system. It was also limiting since e-mail addresses could also appear in the unformatted body text of an e-mail message but would only be recognized as a string selection.

Consequently, a decision was made to use type converters. Using simple heuristics, it is possible to identify potential text strings that might be e-mail addresses. It would have been desirable to augment the e-mail browser with this capability, so that any time text was selected in it, it would try to convert the text to an e-mail address and create an EmailAddress object rather than just a string. But, instead of just giving this type conversion capability to the e-mail browser, that ability should be added to the system once, and allowed to be used in every application where e-mail addresses might appear. The type detection ability was removed from the individual applications and type converters, an independent and extensible layer in the architecture, were created.

For the programmer, writing a type converter involves writing a method that accepts one data type and converts it to another. For the user, adding type converters to a CyberDesk session allows for the use of a wider variety of user context. When user context changes (change in time, location, or data selection), type converters improve the list of suggested actions given by CyberDesk by providing services specific to the content of the user context, not relying simply on the type of user context that has changed.

Currently the list of CyberDesk types includes Strings, Dates, PhoneNumbers, MailingAddresses, Names, URLs, EmailAddresses, GPSPositions, and Times. For each data

type, there is a corresponding StringTo (data type) type converter.

## 6. Chaining

By making a simple extension to the service wrappers, applications can gain the same advantages as type converters. Most of the services provided by applications require data types as input parameters and display their results through a graphical interface. This was seen in the scenario when the user searched for a phone number and mailing address (a Web page was displayed) and when the user looked for a name in the contact manager (a contact entry was displayed). Through the use of simple parsing, this data encoded in the graphical interface can be obtained. In the phone number search, the HTML page returned can be examined and parsed to retrieve a matching phone number. Similarly, if the name being looked up in the contact manager had an existing entry containing an e-mail address, the e-mail address could be easily retrieved. This additional data is part of the user's context and is made accessible to CyberDesk.

When applications are able to generate additional pieces of context and perform user-selected actions, they are behaving both as type converters and as applications, providing the advantages of both. Now, applications can asynchronously suggest both actions directly related and indirectly related to the change in user context, reducing the effort required by the user to find these services. This process of generating additional context for the purpose of increasing integrating behavior is called chaining.

A sample user scenario is described below. A user is reading an appointment in the scheduler and elects the name of the person he/she is supposed to be meeting (Fig. 9). As an experienced user, he/she expects to be presented with a list of all possible services that can use a Name: search for a phone number, mailing address, look for an entry in the contact manager, search for the name on the Web, etc. However, by using chaining, more powerful suggestions can be provided. The WhoWhere Web application takes a name as input and returns a Web browser showing a list of possible e-mail addresses corresponding to that name. If we make the assumption (not always a good one) that the first e-mail address returned in the list is the correct one, we can now use this service to convert the name to an e-mail address. The service now creates a related EmailAddress object, and the user is supplied with all possible suggestions for both a Name and an EmailAddress.

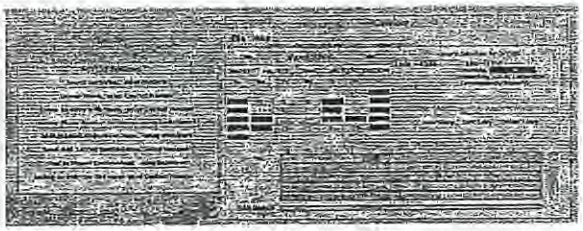


Fig. 9. Chaining example.

Chaining is potentially a very powerful tool for the user to take advantage of. It provides another dimension of suggestions for each data type that the user context can be converted to.

## 7. Combining

Along the same line of thought, chaining can be used along with the concept of combining to make services more powerful. The services previously described were designed to only operate on a single data type (at a time). With data being converted to multiple types via chaining, we should enable services to take advantage of these multiple types. They can, through a process we call combining.

Combining, in CyberDesk terms, is the ability to collect multiple data types and dynamically bind them together, as needed, to create meta-objects which services can use. These meta-objects can be used to perform substantially more powerful actions.

Using the above example of a user reading an appointment in the scheduler, the user selects a name, and a chaining service like Four11 is used to obtain a mailing address (and create a related MailingAddress object) for that name. Using combining, a meta-object containing both the name and the mailing address may now be used as input to a phone number lookup service like Switchboard. Switchboard can find phone numbers when given simply a name, but it can perform a more accurate search when provided with both a name and a mailing address.

Most services will perform better when provided with pertinent, additional context to work with. CyberDesk determines how to bind data together based on the data it currently has (the sum total of the current user context) and on the services available. It will not offer a suggestion to use Switchboard with just a name as input, when it can suggest it with both a name and mailing address.

Now that the concept of combining has been explained, a more complete example demonstrating its power is given below. Again, we will use the example of the user reading an appointment in the scheduler (Fig. 10). The user selects the name of a person to meet tomorrow. Immediately, the user is offered suggestions of actions that can be performed with the selected string and name. As the chaining-enhanced applications return their data, this suggested list of actions is asynchronously augmented with actions that can use an e-mail address (via WhoWhere), phone numbers and mailing addresses (via Switchboard) and URLs (via AltaVista). At the same time, the Integrator is dynamically binding these individual pieces of data for services that benefit from multiple data inputs.

The user chooses to create a new entry in the contact

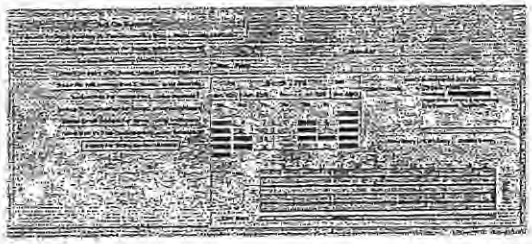


Fig. 10. Combining example—user selects a name and is offered many integrating suggestions.

manager. This results in a rich entry (Fig. 11), containing the original name selected, an e-mail address, a URL, a phone number, and a mailing address.

Combining, like chaining, can be very powerful to the user. It does not inhibit the list of options for individual pieces of user context, while at the same time it combines those pieces, improving the available services and their results. In essence, chaining and combining enhance the context-inferencing engine in CyberDesk.

## 8. Other forms of context

In a mobile setting, there are additional forms of context that are not necessarily available in a desktop environment. Examples include a user's changing location, the changing objects in the environment, and the familiarity with the environment. As described earlier, CyberDesk has integrated services that allow access to data and applications on mobile devices. Now, we are looking at integrating services that are available when the user is mobile, to take advantage of these other forms of context.

Up until now, all of the examples shown have only used changes in context based on the data the user is currently working with. CyberDesk can deal with other forms of context in the same way as it deals with the user's data. Integrated examples include significant changes in time and position. One application that has been added to CyberDesk is a clock that updates the system time every 5 min. Currently, only one service that can use time has been integrated

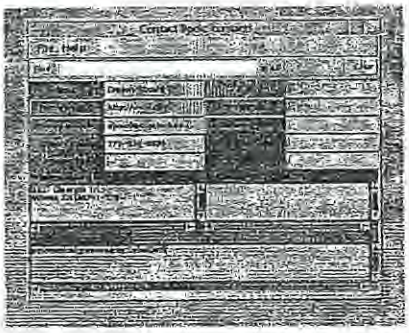


Fig. 11. Combining example—user creates rich contact entry.

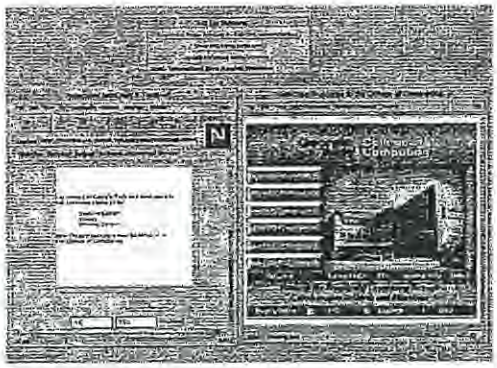


Fig. 12. Screenshot of position service: GPS coordinates are being input (a), causing changes in the user interface (b) when the coordinates correspond to a different Georgia Tech building. The user is keeping track of his trip in the scratchpad (c), and is able to view the building URLs in the Web browser (d).

into CyberDesk. This service is part of the scheduler and it acts as a reminder service for events listed in the scheduler. When the time input into the system is within 15 min of an event, the scheduler offers a suggestion to the user to check their scheduler. Another, but more intrusive option would be to create a window displaying the relevant information to the user. Ideally, the user would be able to set the type of feedback desired, and the event windows (i.e. how often the time service updates and how close to an event should a user be warned).

Position information has also been incorporated into CyberDesk, for use in a mobile setting. The current system uses global positioning system (GPS) data and is intended for outdoor use. The application providing GPS data updates the system position whenever the GPS coordinates change. Again, how often the application updates will be a user-controlled parameter. A service has been written that accepts GPS information for a location on the Georgia Tech campus and returns a URL corresponding to that location. CyberDesk then suggests all the activities it can perform with a URL, including displaying it in a Web browser. An example of this is shown in Fig. 12.

A prototype of an indoor positioning system has been built at Georgia Tech [5]. This prototype could be used as an application offering information on a user's location within a building, updated as a user moves between rooms. Possible services that incorporate both indoor and outdoor positioning information are real-time mapping and directions, access to equipment in the environment, and providing information on important landmarks (washrooms, ATMs, etc.). If these services were combined with knowledge of a user's history, the services could be made even more useful to the user. When a user approaches a building or room they have never been to, the CyberDesk system should offer introductory information on the location. If the user has been there before, different sets of information should be offered.

Additional forms of context can be used to generate new and more appropriate suggestions to the user. As CyberDesk's knowledge of the user's context grows, it is able to create more informed suggestions for the user. We are interested in using CyberDesk's context-inferencing engine as the basis for context-aware applications that we are developing-applications that take advantage of knowing a user's position, history, behavior, etc.

## 9. Background and related work

The underlying framework of CyberDesk that allows integration of isolated services is based on the concept of dynamic mediation. Mediation consists of two basic steps: registration of components and handling of events. Other systems that use mediation include UNIX pipes, Field [6], Smalltalk-80 MVC [7], Common Lisp Object System (CLOS) [8], and APPL/A [9]. UNIX pipes act as mediators that integrate UNIX programs. They are limited to reading and writing streams of data, where stream outputs can only be input to one stream, and they use only a single event. Field (and its extension Forest) integrate UNIX applications that have events and methods which can be manipulated through a method interface. Like CyberDesk, it uses centralized mediation and implicit registration, allowing greater run-time flexibility. However, it suffers from the use of special object components, creating inconsistencies in the way that data is handled. Smalltalk uses a general event mechanism like CyberDesk, but it merges relationships between components into the components themselves, limiting flexibility. CLOS uses wrappers to access data and methods within objects, much like CyberDesk. But it limits the action a component can perform to a simple method call and return, thereby limiting its usefulness. Sullivan et al. [10] have developed a very flexible dynamic mediation system. However, their system allows only one-to-one relationships between components and requires explicit registration of event–action pairs, while CyberDesk allows one-to-many relationships and allows a looser, more flexible, registration process.

CyberDesk also depends on the use of component software and network objects. These concepts are important for system flexibility and reuse. Other systems that provide for these concepts are CORBA (Common Object Request Broker Architecture) [11] and IIOP [12] (Internet Inter-ORB Protocol), IBM's DSOM [13] (Distributed System Object Model) and Microsoft's OLE and DCOM [14]. These are object models that allow cross-network and cross-language integration of applications.

There are three systems that provide functionality similar to CyberDesk. They are OpenStep's services facility [15], Intel's Selection Recognition Agent [16], and Apple Research Lab's Data Detectors [17].

The OpenStep computing environment uses a uniform object-messaging interface between objects in all of its applications, similar to CyberDesk. Using this ability, applications can declare the types of data they can generate and are integrated with services that can operate on that data. The largest difference between CyberDesk and OpenStep services is that CyberDesk acts on both the content and data type being used, rather than just the data type. OpenStep services are primarily used to convert file formats, create dynamic links between objects (i.e. updating an object updates the linking documents), and providing global services such as spell checking and printing. It works only with data the user is attending to, limiting the context types it can use, and does not support the concepts of chaining or combining.

Intel's Selection Recognition Agent attempts to address the same issues as CyberDesk. Unlike CyberDesk, it uses a fixed data type–action pair, allowing for only one possible set of actions for each data type recognized. The actions performed by the agent are limited to launching an application. When a user selects data in an application, the agent attempts to convert the data to a particular type, and displays an icon representative of that type (e.g. a phone icon for a phone number). The user can view the available option by right clicking on the icon with a mouse. For applications that do not 'reveal' the data selected to the agent, the user must copy the selected data to an application that will reveal it. It does not support any of the advanced features of CyberDesk, like chaining or combining, nor does it use any other forms of context like time or position.

Apple Data Detectors is another component architecture that supports automatic integration of tools. It works at the operating system level, using the selection mechanism and Apple Events that most Apple applications support. It allows the selection of a large area of text and recognizes all user-registered data types in that selection. Users view suggested actions in a pop-up menu by pressing a modifier key and the mouse button. Like CyberDesk, it supports an arbitrary number of actions for each data type. It does not support chaining and supports only a very limited notion of combining. When a data type is chosen, a service can collect related information and use it, but this collected information is not made available to other services. The Apple Data Detectors system does not support the use of other forms of context. Its focus appears to be desktop applications, as opposed to CyberDesk's ubiquitous services, existing either locally or remotely.

The three systems discussed deal only with informational context, i.e. the data a user is working with. The majority of context-aware computing to date has been restricted to location-aware computing for mobile applications. This includes the PARCTab [18] from Xerox PARC, the InfoPad project at Berkeley [19], and the Olivetti Active Badge system [20]. A more general programming framework for describing location-aware objects was the subject of Schilit's thesis [21] and reflected a lot of the work done at PARC. While there has been a lot of research in context-aware computing [22], we are not aware of a general toolkit

that supports such a wide variety of user context and integration behavior like CyberDesk does.

There has been some interesting work recently directly related to context-aware computing. Essa and Pentland [23] have used computational perception techniques in an attempt to match actual facial expressions with some prescribed expressions indicating the state of the human (e.g. smiling, frowning, surprized, etc.). Though this work does not claim to be a way to predict human emotions, there is a clear suggestion of how this and related perception research can improve the quality of contextual information that can be gathered. Picard's work on affective computing [24] suggests a similar objective, only through the use of bioelectric signals, coupled with theories on emotion and cognition. The wearable computing community is also looking at the use of context-aware computing. As with mobile computing, a wearable computer user's context is constantly changing. Applications in this area have ranged from understanding sign language [25] to tour guides [26] to airplane maintenance [27]. In all these applications, context has been used to improve the user's experience.

## 10. Issues and future work

The CyberDesk framework was designed to be easily extensible and easy to use; however, it suffers from a few limitations. There is still a programming burden involved with integrating services into CyberDesk. This process is not yet entirely automated. This issue is being addressed through the efforts of the Web-based service writer and the investigation of JavaBeans.

CyberDesk is still limited by the number of different types of user context it utilizes. Note that this is not a limit imposed by the CyberDesk infrastructure. The use of history, personal preferences, and the location of physical objects and landmarks is currently being examined for integration into CyberDesk.

Perhaps the biggest limitation of the system is the user interface. It consists of a window that displays a list of suggested user actions. Although the system looks for repeated suggestions, it is clear that the number of possible suggestions could quickly become overwhelming to the user. Possible methods for limiting the number of suggestions are:

- before displaying a suggestion, contact the service corresponding to the suggestion and ensure that it can successfully perform the action;
- pass the service name along with the data it generates, to see if the service has already acted on the data;
- use user history and preferences to select suggestions most likely to be accepted;
- use context to filter out suggestions.

For example, if a user selects a name in an e-mail message and the system knows that the name is not in the contact manager, CyberDesk should not offer a suggestion to lookup the name in the contact manager, but instead should suggest to create a new entry in the contact manager. Another example is when the system has access to a user's history, it could determine the most likely actions a user is likely to take, and limit the suggestions to those, or at least order the suggestions accordingly.

We are currently looking at different ways to adapt the interface to initially show actions that the user is likely to take, while providing a way for the user to see other possible actions as well. We are also looking at different presentation methods for the suggestions, including pop-up hierarchical menus, having menus associated with each individual application, and document lenses [28].

CyberDesk has also shown the potential for supporting more complex forms of context. For example, if an e-mail message contains information about a meeting, and the user selects the message content, a type converter could potentially convert the text to a Meeting object to be inserted in the user's Calendar Manager. Of course, retrieving context from arbitrary text is a very difficult problem being investigated by the artificial intelligence learning community. But the power of CyberDesk supports the ability to use this richer context, if available.

We will continue to add services to expand CyberDesk's library of components but this will not be our main focus. We are more interested in the following research areas.

- Examining the use of advanced techniques like chaining and combining and searching for others.
- Investigating learning-by-example techniques [29] to allow the CyberDesk system to dynamically create chained suggestions based on a user's repeated actions.
- Incorporating rich forms of context into CyberDesk, other than time, position, and meta-types. This will allow us to use CyberDesk as the platform for developing context-aware, mobile applications.
- Experimenting with adaptive interfaces and different interface representations in order to determine better ways of presenting suggestions to our users.
- Applying CyberDesk's context-inferencing engine to build other context-aware applications. This will include the use of both physical and emotional context, and group context (as opposed to that of a single individual).

## 11. Conclusions

Providing intelligence in a ubiquitous environment can be achieved by taking advantage of the user's context. Context includes the information a user interacts with on a desktop or mobile device, location, time, etc. We have developed a framework for integrating software services based on a user's context. CyberDesk eases the burden on programmers by relieving the necessity to determine all integrating possibilities and eases the burden on users by relieving the

necessity to understand how applications work together. Through the use of advanced techniques like chaining and combining, we have shown the potential for integrating behavior that is too complicated for a programmer to statically design. Context-aware integration changes the paradigm of interaction from a user seeking out functionality in software applications to the infrastructure seeking out the user at relevant times.

## Acknowledgements

AD is supported by Motorola Corporation through the University Partnerships in Research (UPR) Program sponsored by Dr Ron Borgstahl. The authors would like to thank the members of the Future Computing Environments Group and the numerous other undergraduate and graduate students at Georgia Tech who have provided much inspiration and support in the development of the initial CyberDesk prototype and have offered us a lot of evidence for the scalability of the infrastructure.

## References

[1] A. Wood, CAMEO: Supporting Observable APIs. Position paper for the WWW5 Programming the Web Workshop, Paris, France, May 1996.
[2] A.K. Dey et al., CyberDesk: a framework for providing self-integrating ubiquitous software services, Technical Report, GVU Center, Georgia Institute of Technology, GIT-GVU-97-20, 1997.
[3] Java Soft, JavaBeans homepage. Available at http://splash.javasoft.com/beans/.
[4] M. Pinkerton, Ubiquitous computing: extending access to mobile data, Masters Thesis, Georgia Institute of Technology, June 1997.
[5] S. Long et al., CyberGuide: prototype context-aware mobile applications, in: Proceedings of CHI '96, ACM Press, Vancouver, Canada.
[6] D. Garlan et al., Low-cost, adaptable tool integration policies for integrated environments, in: Proceedings of SIGSOFT 90: Fourth Symposium on Software Development Environments, Irvine, CA, 1990.
[7] G. Krasner et al., A cookbook for using the model-view-controller user interface paradigm in Smalltalk-80, Journal of Object Oriented Programming 1 (3 August/September) (1988) 26–49.
[8] D. Bobrow et al., Common Lisp Object System Specification X3J13, Document 88-02R, ACM SIGPLAN Notices 23, September 1988.
[9] S. Sutton et al., APPL/A: A prototype language for software process programming, University of Colorado Technical Report CU-CS-448-89, University of Colorado, Boulder, CO, 1989.
[10] K. Sullivan et al., Reconciling environment integration and component independence, in: Proceedings of SIGSOFT 90: Fourth Symposium on Software Development Environments, Irvine, CA, 1990.
[11] T. Brando, Interoperability of the CORBA specification, MITRE Document MP-95B-58, February, 1995.
[12] Object Management Group homepage. Available at http://www.omg.org.
[13] SOM Object homepage. Available at http://www.software.ibm.com/ad/somobjects/.
[14] Microsoft OLE development homepage. Available at http://www.microsoft.com/oledev.
[15] OpenStep, Topics in OpenStep programming. Available at http://developer.apple.com/techpubs/rhapsody/system/Documentation/Developer/YellowBox/TasksAndConcepts/ProgrammingTopics/services.pdf.
[16] M. Pandit, S. Kalbag, The selection recognition agent: instant access to relevant information and operations, in: Proceedings of Intelligent User Interfaces '97, ACM Press, Atlanta, GA.
[17] Apple Research Labs, Apple Data Detectors homepage. Available at http://www.research.apple.com/research/tech/AppleDataDetectors/.
[18] R. Want et al., An overview of the PARCTAB ubiquitous computing experiment, IEEE Personal Communications 2 (6) (1995) 28–43.
[19] A.C. Long, Jr., et al., A prototype user interface for a multimedia terminal, in: Proceedings of CHI '95, Interactive experience demonstration, ACM Press, Atlanta, GA, 1995.
[20] R. Want et al., The active badge location system, ACM Transactions on Information Systems 10 (1) (1992).
[21] B. Schilit, A context-aware system architecture for mobile distributed computing, Ph.D. Thesis, Columbia University, 1995.
[22] G.D. Abowd et al., Context-awareness in wearable and ubiquitous computing, Technical Report, GVU Center, Georgia Institute of Technology, GIT-GVU-97-22, 1997.
[23] I. Essa, A. Pentland, A vision system for observing and extracting facial action parameters, in: Proceedings of the Computer Vision and Pattern Recognition Conference, IEEE Computer Society, 1994, pp. 76–83.
[24] R. Picard, Affective computing, Technical Report 321, MIT Media Lab, Perceptual Computing, November 1995. Available as MIT Media Lab Perceptual Computing Techreport 362.
[25] T. Starner et al., A wearable computing based American sign language recognizer, in: Proceedings of the IEEE International Symposium on Wearable Computers, Cambridge, MA, 1997.
[26] S. Feiner et al., A touring machine: prototyping 3D mobile augmented reality systems for exploring the urban environment, in: Proceedings of the IEEE International Symposium on Wearable Computers, Cambridge, MA, 1997.
[27] L. Bass et al., The design of a wearable computer, in: Proceedings of CHI '97, ACM Press, Atlanta, GA, 1997.
[28] E.A. Bier et al., ToolGlass and Magic Lenses: the see-through interface, in: Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, 1993, pp. 73–80.
[29] A. Cypher, EAGER: Programming repetitive tasks by example, in: Proceedings of CHI '91, ACM Press, Atlanta, GA, 1991.

# Exhibit 37

# Apple Internet Address Detectors User's Manual

Apple Internet Address Detectors utilizes a new Apple technology called *Data Detectors*. Data Detectors enables your computer to recognize and then act on certain types of information, or *data*, in your documents. Apple Data Detectors can recognize several different types of data, and soon software developers will extend the capabilities of Apple Data Detectors even further.

Apple Internet Address Detectors for Mac OS 8 can recognize and act on data that's in the form of Internet addresses, which includes the following:

- e-mail addresses
- Web sites
- newsgroup names
- filenames on FTP (file transfer protocol) sites
- names of remote computers

For example, if you have a word-processing document that contains several e-mail addresses, Apple Data Detectors can quickly scan the document, identify all the addresses, and then open a new e-mail message addressed to the one you select. Or, let's say that someone sends you a World Wide Web address in an e-mail message. You can use Apple Data Detectors to find the address within the message, then open the Web page in your favorite Web browser program.

For each type of information that Apple Data Detectors identifies, you can select an action to perform with it. The actions available with Apple Internet Address Detectors include

- addressing a new e-mail message to the selected address
- opening a Web browser program and connecting to the selected Web site
- bookmarking the selected Web site in a Web browser program
- saving a Web document as a file on your hard disk
- downloading the selected file from an FTP site
- connecting to the selected remote computer
- opening a newsgroup with your news reader program

### What's in this manual

Read this manual for information about these topics:

- system requirements
- installing the software
- getting started with Apple Data Detectors
- using Apple Data Detectors
- configuring Apple Data Detectors
- customizing detectors
- troubleshooting

## System requirements

To use Apple Data Detectors, you must have the following:

- a computer with a PowerPC processor
- Mac OS 8 or later

## Installing Apple Data Detectors

To install Apple Data Detectors, follow these steps:

1. **Double-click the Installer icon.**


2. **Read the information that appears, then click Continue.**

3. **In the window that appears, read the software license agreement. If you agree to the terms, click Agree. Otherwise, click Cancel.**

   If you click Cancel, you won't be able to install the software.

4. **In the dialog box that appears, click OK.**

   The dialog box tells you that the Installer will install actions that require the use of a variety of application programs. If you don't use a specific program, you can turn off the actions that require it using the Apple Data Detectors control panel. See "Configuring Apple Data Detectors" later in this manual for instructions.

5. **In the installation dialog box, click Install.**


6. **When the installation is complete, click Restart.**

## Getting started with Apple Data Detectors

Apple Data Detectors works with any Mac OS program in which you can select, or *highlight,* text. Apple Data Detectors quickly scans the selected text for information in specific formats. It uses *detectors* that are programmed to recognize specific types of information. For example, this version of Apple Data Detectors includes several detectors that can recognize Internet addresses and *uniform resource locators* (URLs).

Once a detector has identified a piece of information it recognizes, Apple Data Detectors creates a menu of *actions* for you to choose from. Actions are things you can do with the detected information, such as sending it to another program or saving it for later use. The actions that are available depend on the type of information detected.

This table summarizes the actions supplied with Apple Internet Address Detectors:

| Type of data | Example | Actions |
|---|---|---|
| e-mail address | moof@apple.com | send e-mail to address |
| Web address | http://www.apple.com/file.html | view Web site, bookmark the site, or save as document |
| newsgroup | comp.sys.mac | read newsgroup |
| file on an FTP site | ftp://apple.com/file.sit | download the file |
| host address | research.apple.com | connect to the remote computer |

Use the Apple Data Detectors control panel to learn more about what the actions and detectors do. See "Configuring Apple Data Detectors" later in this manual for more information.

## Using Apple Data Detectors

To use Apple Data Detectors, follow these steps:

1 **Select some text in any application that allows you to highlight text.**

Make sure the selected text contains at least one type of data that Apple Data Detectors recognizes. (In this example, the selected text contains one complete Internet address.)


2 **Hold down the Control key, then press and hold down the mouse button.**

A "contextual menu" appears. It lists all the recognized data found in the selection.


*Tip:* If the contextual menu is empty, or the message "no structures found in selection" appears, the text you selected did not contain any recognizable data.

After the contextual menu appears, you can release the Control key. Be sure to keep holding down the mouse button or the menu will disappear.

**3** **Choose a recognized data item from the contextual menu, then choose an action from the submenu that appears.**


**4** **If asked, locate the application associated with the action you selected.**

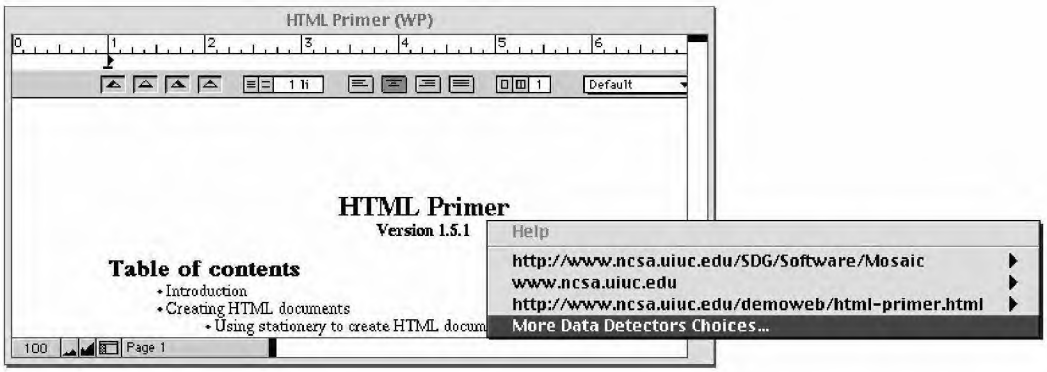
The first time you perform each action, you may need to locate the application program the action requires. In this example, you are asked to locate the Netscape Navigator™ 3.01 program. Use the dialog box to locate the program on your system, then click Open.


You must have the requested program installed in order to use the action. If you are asked for a specific version of a program, just select the version you have. In most cases, the action will still work.

After you locate the program and click Open, the action takes place automatically.

## Apple Data Detectors and large selections

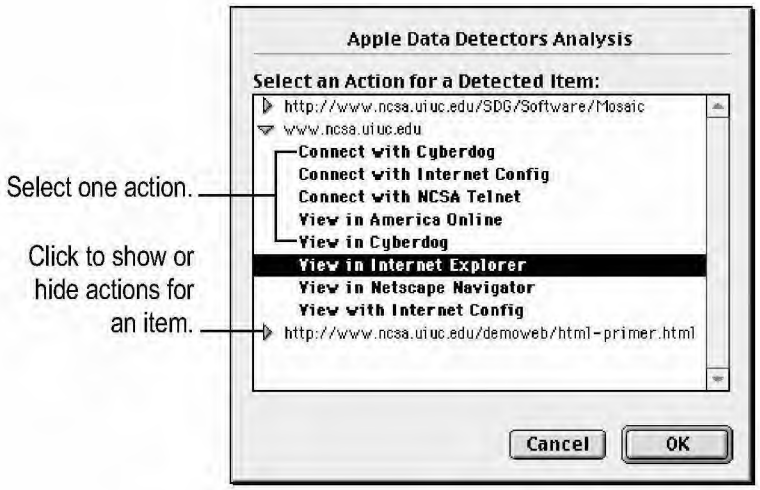
When you select a large amount of text, it takes more time for Apple Data Detectors to recognize all the data in the selection. If it takes longer than half a second to analyze the whole selection, the contextual menu will list only the data found so far, along with a command called More Data Detectors Choices.



If you don't see the data you want in the menu, you can use the More Choices command to find it. Do the following:

- Choose More Data Detectors Choices from the contextual menu.

Apple Data Detectors completes the analysis of the selected text. A dialog box appears listing all the detected data in the selected text, and the actions you can apply to each item of data.




To apply an action, follow these steps:

1. **In the list of detected items, locate one item that you want to use.**
2. **Click the triangle next to the item.**

   Actions available for the item appear when you click the triangle.
3. **Select one of the actions listed for the item.**
4. **Click OK.**

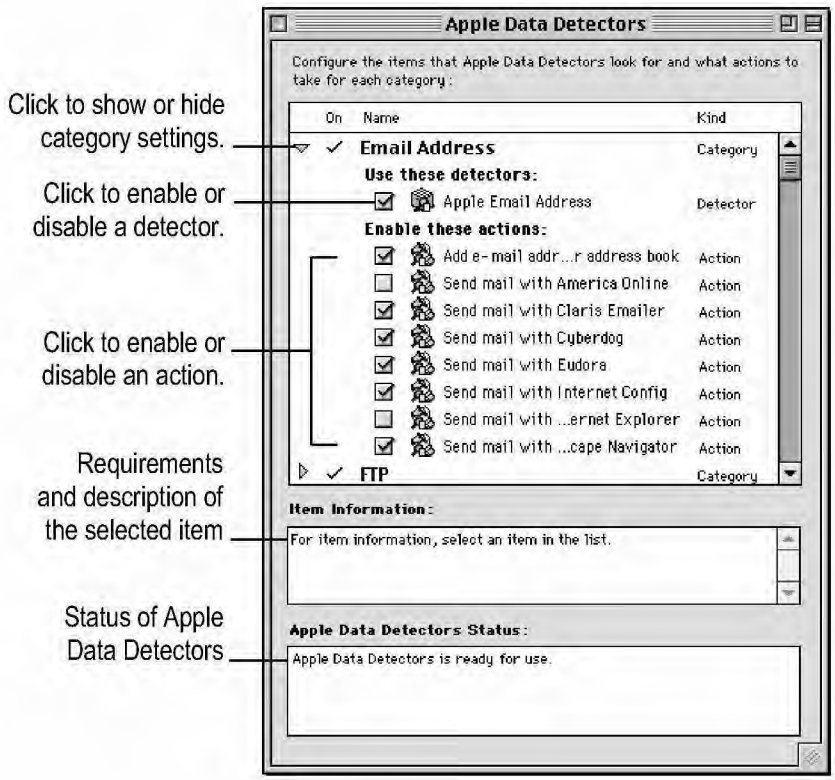
   The selected action takes place immediately.

## Configuring Apple Data Detectors

You can control what kinds of data Apple Data Detectors looks for in your document, and the type of actions that are available to apply to the detected data. For example, you can turn off the detection of e-mail addresses, or you can turn off the actions that require the use of a specific program.

To configure detectors and actions, use the Apple Data Detectors control panel. The control panel lists the categories of information that Apple Data Detectors looks for, the detectors that recognize the format of data in each category, and the actions you can apply to each recognized item.

To open the Apple Data Detectors control panel, choose Control Panels from the Apple (🍎) menu, then choose Apple Data Detectors from the submenu.



## Learning about detectors, actions, and categories

To learn more about each of the items in the control panel, select an item, then read the Item Information box. For example, if you click an action, the information box tells you what the action does and any application programs it requires. (You might have to use the arrows to scroll through the text in order to read all the information.)

## Enabling or disabling detectors

You can limit what types of data are detected by turning individual detectors on or off. For example, if you don't read newsgroups, you can turn off the newsgroup detector.

- To turn a detector off, click its checkbox to remove the X.

## Enabling or disabling actions

You can limit the actions listed in the Apple Data Detectors contextual menu. For example, if you don't use Cyberdog to send e-mail, you can turn off the Cyberdog action for e-mail addresses.

- To turn an action off, click its checkbox to remove the X.

*Note:* If an action's checkbox is dimmed, the detector that the action requires is probably turned off. Select the action, then read the item information to determine which detector the action requires.

## Installing new detectors and actions

If you obtain additional detectors or actions, you install them using the Apple Data Detectors control panel.

To install detectors or actions, follow these steps:

1. **Choose Control Panels from the Apple (🍎) menu, then choose Apple Data Detectors from the submenu.**
2. **Open the File menu and choose Install Detector File or Install Action File.**
3. **In the dialog box that appears, locate the detector or action you want to install.**
4. **Click OK.**

5 **Wait a moment while the new detector or action is added.**

The status message in the control panel tells you what Apple Data Detectors is doing. When the item has been added, the status message indicates that Apple Data Detectors is ready for use.

### Removing detectors and actions

If you are certain that you'll never use a detector or action, you can remove it from the Apple Data Detectors control panel.

1 **Click once to select the action or detector you want to remove.**

2 **Open the File menu and choose Remove.**

The selected item is immediately and permanently deleted.

*Tip:* Instead of removing actions or detectors, you can disable those you aren't currently using. This way you won't permanently delete something you may later want to use.

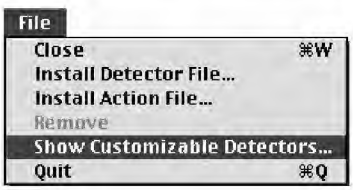
## Customizing detectors

Some detectors can be modified to improve their recognition of data. For example, you can modify the domains that the World Wide Web address detector can identify.

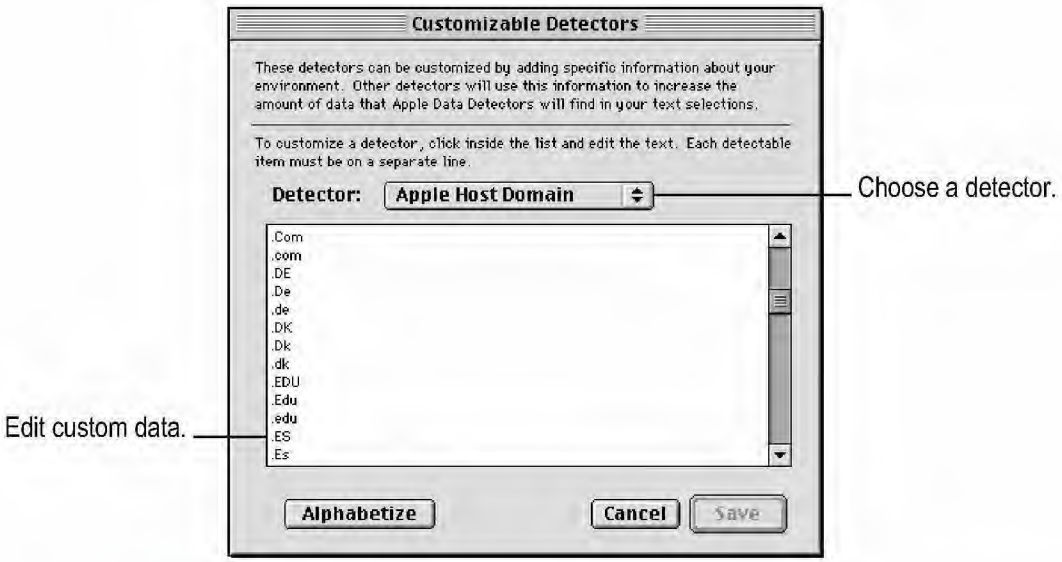
To modify a customizable detector, follow these steps:

1 **Choose Control Panels from the Apple (🍎) menu, then choose Apple Data Detectors from the submenu.**

2 **Choose Show Customizable Detectors from the File menu.**



The Customizable Detectors dialog box appears.




**3** **Open the Detector pop-up menu and choose the detector you want to customize.**

*Tip:* This version of Apple Data Detectors has two detectors you can customize. If you have additional customizable detectors installed, such as those from a third party, more choices may be available.

**4** **To delete a domain suffix or newsgroup prefix, select it in the list, then press the Delete key. To add a domain suffix or newsgroup prefix, type it in the list.**

- The host detector searches for addresses of remote computers. Many of the common domains are already listed, but you can specify additional domains (such as ".br" for computers in Brazil) that you want the detector to recognize.
- The newsgroup detector searches for newsgroup names. Many of the popular Usenet hierarchies are already listed, but you can specify additional newsgroup prefixes (such as "Cyberdog" for Cyberdog topics) that you want the detector to recognize.

*Note:* Apple Data Detectors considers uppercase and lowercase letters to be different. For example, you should add ".BR," ".br," and ".Br" to the list to ensure proper recognition of host computers in Brazil.

5 **Click Save.**

The status message in the control panel tells you what Apple Data Detectors is doing. When the items have been added, the status message indicates that Apple Data Detectors is ready for use.

## Troubleshooting

**The same information appears more than once in the contextual menu.**

This is normal. Some detectors recognize multiple parts of an address, and display each part separately. For example, if the selected text contains a reference to "http://www.apple.com/news.html," both the host site (www.apple.com) and the specific document (www.apple.com/news.html) will appear in the contextual menu.

**An action asks me to locate an application I don't have.**

If an action asks for a specific version of an application, such as Cyberdog 2.0, try selecting the latest version that you do have. In most cases, the action will still work.

If you don't have the application you're asked to locate, you cannot use the action. You can disable actions for programs you don't have installed so they won't appear in the contextual menu. See "Enabling or Disabling Actions" earlier in this manual for more information.

**The address I want is always in the More Choices dialog box, or at the bottom of the contextual menu.**

Apple Data Detectors lists the recognized information in the same order it appears in the selected text. If analyzing the text takes longer than half a second, the More Choices menu item appears, and you must choose it to finish the analysis. For better results, select less of the text in your document so the address you want is nearer the top of the list and all of the analysis can take place at once.

**The contextual menu doesn't appear.**

Make sure that your pointer is positioned over the selected text and that you hold down the Control key before pressing the mouse button. (Once the menu appears, you can release the Control key.)

Make sure you continue to hold down the mouse button after the menu appears. When you release the button, the menu disappears.

Check the status message in the Apple Data Detectors control panel. Apple Data Detectors can't be used while detectors and actions are being compiled.

Make sure Apple Data Detectors is properly installed. Use the Extensions Manager to make sure that the Apple Data Detectors control panel and the Text Encoding Converter, Apple Data Detectors Extension, SOM Objects for Mac OS, Contextual Menu Enabler, and Contextual Menu Extension are all turned on. Then restart your computer.

If you continue to experience problems, reinstall Apple Data Detectors.

**Every time I select an action, I'm asked to locate a program.**

The first time you use each action you must locate the application program that the action requires. For example, the first time you use the action "Open in Cyberdog," you must locate the Cyberdog application. Also, the first time you use "Send e-mail using Cyberdog," you must locate the Cyberdog application, and so on until you have used each action at least once. If you later move or rename the application, or reinstall the actions, you may again be asked to locate applications for the actions.

**Actions that require Claris Emailer™ don't work.**

If Emailer is configured to display the Connect Now dialog box automatically when Emailer opens, the actions will not be completed. You should turn off the Connect Using option in the Emailer preference settings.

**Actions that require Fetch don't work.**

If Fetch is configured to display the New Connection dialog box automatically when Fetch opens, the actions will not be completed. You should turn off the Show Sign-on Dialog At Startup option in the Fetch preferences.

**The Help menu item is unavailable.**

The Help menu item in the contextual menu provides help for the application in which you've highlighted the text. For help using Apple Data Detectors, refer to this manual.

**Apple Data Detectors doesn't recognize a valid Internet address.**

Make sure that you highlight the entire address before invoking Apple Data Detectors.

Make sure the host domain (or newsgroup prefix, if applicable) is listed in the detector's configuration. See "Customizing Detectors" earlier in this manual for more information.

Apple Data Detectors considers uppercase and lowercase letters to be different. If a domain or newsgroup prefix is in mixed-case (".COm") it may not be recognized unless you customize the detector. See "Customizing Detectors" for more information.

**Apple Data Detectors replaces the contents of the Clipboard.**

If an application program doesn't directly support the contextual menu, the Contextual Menu Enabler copies the selected text to the Clipboard and then invokes Apple Data Detectors. Any information you had on the Clipboard is replaced by the selected text. If an application supports the contextual menu, this won't occur.

Apple Computer, Inc.



Every effort has been made to ensure that the information in this manual is accurate. Apple is not responsible for printing or clerical errors.

Apple Computer, Inc.
1 Infinite Loop
Cupertino, CA 95014-2084
408-996-1010
http://www.apple.com



Simultaneously published in the United States and Canada.

Last updated August 28, 1997.

# Exhibit 38

# The Selection Recognition Agent:

# Instant Access to Relevant Information and Operations

**Milind S. Pandit**
+1 503 264 8880
milind_s_pandit@ccm.jf.intel.com

**Sameer Kalbag**
+1 503 264 9644
sam_kalbag@ccm.jf.intel.com

Intel Architecture Laboratories
Mailstop JF2-64
5200 NE Elam Young Parkway
Hillsboro, OR 97124-5961 USA

**ABSTRACT**
We present the Selection Recognition Agent (SRA), a personal computer application which recognizes meaningful words and phrases in text, and enables useful operations on them. The SRA includes six recognition modules for geographic names, dates, email addresses, phone numbers, Usenet newsgroup name components, and URLs, as well as a module that enables useful operations on text in general. The SRA runs on Microsoft Windows 95 and Windows NT* and is currently available free from Intel's home page (http://www.intel.com).

**Keywords**
selection, recognition, agent, object-oriented interface, geographic name, date, email address, phone number, Usenet news, URL, web

**INTRODUCTION**
The authors' charter is to integrate technologies into advanced prototype applications that aid users in finding, storing, using, and communicating Internet information. The SRA addresses this charter by enabling users to utilize desktop information and acquire relevant information more efficiently and effectively.

The SRA is an unobtrusive program that a user constantly runs on his PC. The SRA monitors operating system events to determine when the user has selected text in a window. It then uses fast, simple recognition processes to identify meaningful objects in the selected text. The SRA can currently recognize geographical names, dates, email addresses, Usenet newsgroup name components, world-wide web site names (URLs), and phone numbers. If the SRA recognizes one of these in the selected text, it alerts the user. The user can then use SRA to perform operations that are relevant to the recognized text object. For example, the SRA can start a web browser on a page referenced by a selected URL, or download a Usenet newsgroup's list of Frequently Asked Questions (FAQs).

The SRA is available free from Intel's home page, http://www.intel.com/.

The value added by the SRA can be understood from a few different perspectives:

- A user is prompted to initiate many common but tedious desktop operations by the appearance of text in a window. The SRA automates these operations.
- The SRA automatically turns plain text into a kind of hypertext, by quickly recognizing selected text, and then linking it to related information and applications. Thus, the SRA turns the entire desktop into a kind of hypertext document.
- The SRA provides, on the fly, an object-oriented interface to all text objects visible on the desktop.

**RELATED WORK**
Although the technologies in the SRA are relatively simple and widely known, we know of no other widely available software with its capabilities. The strengths of SRA are:

- It recognizes several classes of text objects.
- It recognizes text objects appearing in any desktop window.

*Third-party brands and names are the property of their respective owners.


IUI 97, Orlando Florida USA
© 1997 ACM 0-89791-839-8/96/01 ..$3.50

- It links text objects to multiple sources of relevant information by automating the process of acquiring that information.
- It uses context menus, a well-known component of the Microsoft Windows 95 user interface [6]. As a result, the SRA is familiar and immediately usable for Windows 95 users.

The SRA currently uses simple lookup tables, hand-generated parsers, and parsers generated using GNU Flex and Bison to classify text strings. The strength of this approach is that the SRA's recognition processes are fast and predictable. The weakness of this approach is that the recognition algorithms are inflexible. It is not possible for the end user to modify the classes of text they recognize.

The Netscape Navigator does parse URLs and email addresses, converting them to hypertext links when displaying Usenet News and email messages. However, it only parses these two classes of text, and links them only to web pages or to its internal email application.

NetManage's ECCO product includes a Shooter which allows ECCO to acquire text from any application desktop window. However, unlike SRA, the Shooter overwrites the contents of the clipboard when "shooting" text to ECCO. Also, the Shooter does not treat selections as anything other than plain text.

The Remembrance Agent [4] uses information retrieval algorithms to continuously locate documents in a database that are relevant to text surrounding the cursor in GNU Emacs. Not only is its operation is limited to a single application (Emacs), but its document space is limited to a history of documents that have passed through a particular user's instance of Emacs. The use of statistical information retrieval algorithms allows the Remembrance Agent to retrieve more generally relevant information than the SRA. However, this also makes the Remembrance Agent less predictable than the SRA. The Remembrance Agent's client-server architecture also makes it less responsive than the SRA.

Cypher, et. al. [2] and Cypher [3] discuss several sophisticated ways in which an interface can learn to automate repetitive user actions. These systems have advantages over the SRA, which does not learn new ways to recognize information or new ways to link text objects to relevant information.

IntelliSense features in Microsoft Word 7.0 represent more sophisticated recognition of patterns in user actions than are implemented in SRA. Although the features are limited to a single application, they do have an advantage over SRA, which really only detects a single user action: selection of text.

## THE SELECTION RECOGNITION AGENT

Figure 1

When initially executed, the SRA displays an eyeball (Figure 1) on the screen, to suggest that it is now observing the user's selections and other desktop activities. It provides options to place this icon in the desktop area with other application windows, to set one of two sizes for it, to place it in the taskbar notification area, to animate it, to position it with respect to the active application window, to cause it to float above other application windows, or to cause it to appear only when the SRA recognizes something.

When it is displayed with other windows in the desktop, the SRA can be moved by dragging it using the left mouse button. When the user right-clicks on the SRA, a context menu appears (Figure 2) showing operations that are

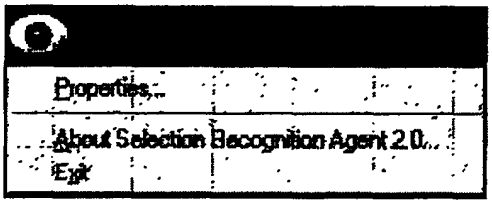



Figure 2

currently available. These features are consistent with Windows 95 user interface guidelines.

The SRA acquires text for recognition in one of two ways. The first way is by monitoring user actions in other applications. In most applications, when the user selects text by dragging the mouse to highlight it, the SRA acquires the text and attempts to recognize meaningful objects within it. *The application need not be aware of the existence of the SRA.* The second way is by monitoring the clipboard. When text is copied to the clipboard, the SRA attempts to recognize meaningful objects within it.

When the SRA recognizes an object, it changes its icon to reflect the class of the object. The icon for recognition of URLs is shown in Figure 3. The SRA can also be configured to make a sound upon recognition of an object.

Figure 3

The tool tip (a small window which appears when the pointer rests on the icon) changes to indicate the class of object recognized. Operations relevant to the class of the recognized object now appear in the context menu (Figure 4). In most cases, the operation involves launching a program. The selected object is either placed in the command line or on the clipboard in a canonical format useable by the program. Thus, the SRA operates in a cycle of selection of text by the user, recognition of an object

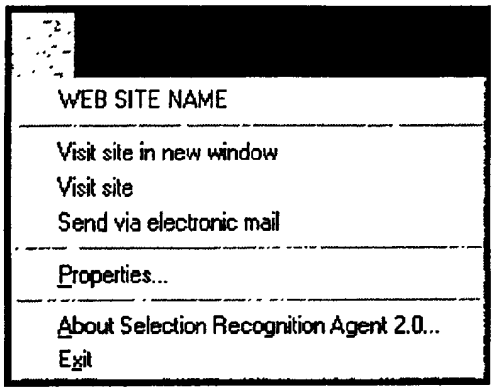



Figure 4

within the text, and operation on the object.

The SRA can recognize multiple objects within selected text. Furthermore, it can classify a single object in multiple ways. For example, the string "June 14, 1996" can be classified both as a date, or more generally as a piece of text. When multiple objects or a single object with multiple classifications are recognized, the SRA adds submenus for each class of operations to its context menu (Figure 5). The icon and main menu refer to the most specific classification

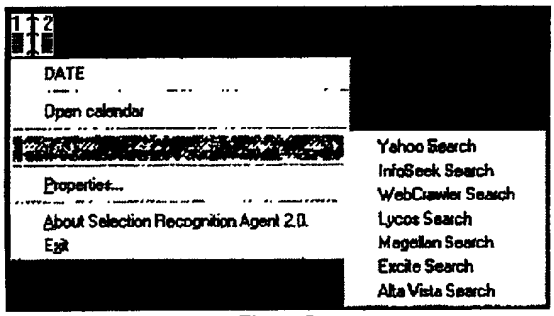



Figure 5

of the first recognized object in the text. (Specificity is defined by a class hierarchy discussed later.) The SRA can be configured to perform a default operation—the first operation in the main menu—when the user double-clicks on its icon.

As currently implemented, the SRA can recognize the following kinds of objects in text:

- The SRA can recognize over 1700 names of cities, states, countries, or continents. The SRA provides an option to visit a CityNet web site containing information about the recognized geographical location.
- The SRA can recognize dates in a variety of formats. The SRA provides an option to start the calendar program of the user's choice. The user has the option to use the recognized date in a canonical format as a command line parameter. The date is also placed on the clipboard.
- The SRA can recognize electronic mail addresses. The SRA provides an option to start the email program of the user's choice. The user has the option to use the recognized email address in a canonical format as a command line parameter.
- The SRA can recognize phone numbers in a variety of formats. The SRA provides an option to start the phone book program of the user's choice. The user has the option to use the recognized phone number in a canonical format as a command line parameter. The phone number is also placed on the clipboard.
- The SRA can recognize words which are the components of Usenet Newsgroup names. The SRA provides an option to retrieve the FAQ for those newsgroups from ftp://rtfm.mit.edu or a mirror site.
- The SRA can recognize URLs. It provides an option to visit the web site, either in a running browser or in a new instance of the browser.
- In addition, the SRA provides an option to retrieve the definition of any single word. It also provides an option to perform web searches on any text.

## ARCHITECTURE AND DESIGN

The SRA coordinates the operation of multiple recognizers, each of which recognizes and provides relevant operations for a single class of text. Recognizers are modular. All recognizers are accessed by the SRA through an identical interface. The list of available recognizers is retrieved from registry information during initialization of the program. As a result, additional recognizers may be implemented and added to the SRA without recompiling the entire system.

The date and phone number recognizers use parsers generated by GNU Flex and Bison to recognize dates and phone numbers in diverse formats. The electronic mail address and URL recognizers use hand-written parsers to recognize email addresses and URLs in one or two standard formats. The city and key word recognizers maintain large lists of recognizeable cities and key words, which are accessed quickly using hash tables.

The recognizers are arranged in a simple hierarchy similar to a class hierarchy (Figure 6). The hierarchy is used to

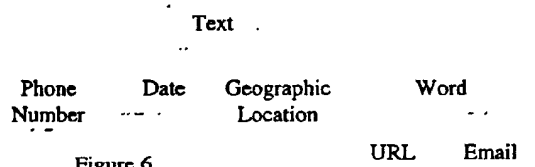



Figure 6

construct the context menu and submenus when multiple recognizers respond to the same object: the lowest (most specific) recognizer in the hierarchy provides the icon and the main menu, and recognizers above it provide sub-menus. The hierarchy is not hard-coded into the SRA, but rather is constructed from registry information during initialization.

The SRA acquires text in the following manner: It first registers with the operating system to receive notification whenever the contents of the clipboard change. It then uses a system hook procedure to receive notification whenever the right mouse button is released in any window. When such a notification arrives, it backs up the current contents of the clipboard and sends a special message to the window. Most windows respond to this message by copying any selected text to the clipboard. When the clipboard-change notification arrives, the SRA submits the clipboard text to the recognizers and then restores the previous contents of the clipboard. In this manner, text can be acquired from most applications, without disturbing the clipboard, immediately after it is highlighted. In other applications, copying the text to the clipboard is sufficient for the SRA to try recognizing it.

When the SRA acquires text, either from another application or from the clipboard, it submits the text to each recognizer to determine if any of them recognize objects within the text. A recognizer is activated when as soon as it recognizes an object within the text. If more than one recognizer is activated, the SRA chooses the most specific one. The SRA asks this recognizer for its icon, and for the names of operations it supports. The SRA changes its icon to that of the recognizer, and adds the recognizer's operations to its context menu. If less specific recognizers were also activated, the SRA asks them for the names of operations they support, and adds these to sub-menus of its context menu.

When the user selects an operation from the context menu, the SRA asks the appropriate recognizer to perform the appropriate operation on the last object it recognized.

**MEASUREMENT OF UTILITY**

We measured the utility of the SRA in two ways. First, we counted, by hand, the operations (keystrokes and mouse manipulations) required to operate on text objects with and without the SRA. Detailed findings are included in the appendix. Note that we have omitted any keystrokes or mouse manipulations required to find the necessary application in the file system. We have also omitted any effort required by a user, when not using the SRA, to convert an object to a format usable by an application. We performed this measurement for every operation on every class of objects supported by the SRA. The measurement was taken from the time text becomes visible to the time the last keystroke or mouse manipulation required for the operation is completed. Below is a table summarizing our results.

| Operation | Without SRA | With SRA |
|---|---|---|
| Acquire info on Detroit, MI | 6 operations | 2 operations |
| View 7/1/1996 in calendar program | 6 | 4 |
| Send electronic mail to milind_s_pandit @ccm.jf.intel.com | 6 | 4 |
| Put phone number in cardfile | 5 | 3 |
| View definition of "apropos" | 6 | 2 |
| View FAQs regarding tennis | 6 | 2 |
| Perform web search on text | 6 | 2 |
| Visit web site in running browser | 4 | 2 |
| Visit web site in new browser. | 5 | 2 |

Second, we measured the time required to operate on text objects with and without the SRA. The stopwatch was started when the text became visible to the user, and stopped when the last keystroke or mouse manipulation required for the operation was completed. The time shown is the average for 5 users.

| Operation | Without SRA | With SRA |
|---|---|---|
| Acquire info on Detroit, MI | 55 s | 12 s |
| View 7/1/1996 in calendar program | 21 s | 11 s |
| Put phone number in cardfile | 16 s | 9 s |
| View definition of word | 26 s | 12 s |
| View FAQ regarding word | 67 s | 19 s |

| | | |
|---|---|---|
| Perform web search on text | 24 s | 15 s |
| Visit web site in running browser | 8 s | 5 s |
| Visit web site in new browser | 13 s | 9 s |

**Interpretation of Results**

Our results confirm that the SRA provides tremendous savings in both mouse/keyboard operations and time for accessing information relevant to text on the desktop. For every task, the SRA saved both time and effort, in some cases over 50%.

We would have liked to plot a Cost-of-Knowledge Characteristic Function [1] for the SRA. The SRA clearly enables direct-walk navigation to relevant information destinations, as defined by Card, et. al. However, the SRA effectively integrates the desktop shell with a number of applications as well as the World-Wide Web. It is not clear where a direct-walk navigation begins in an operating system shell—when the system boots up? Our measurements above start when the appearance of a text object suggests obvious destinations for the user. It is also not clear what such a plot could be compared with, besides the plot for an operating system running without SRA. In any case, we believe the cost of knowledge is very low: the SRA provides direct access to the pages of an on-line dictionary as well as thousands of pages of geographic information and FAQs, so that a huge number of documents is potentially accessible at minimal cost.

## FUTURE WORK

We would like to enhance the Selection Recognition Agent along the lines of Eager [3] allowing it to detect the repetition of action sequences in any application and automate these sequences. Schlimmer and Hermens [5] describe a note-taking software system with features for predicting user actions and automatically creating user interfaces. SRA may also benefit from similar features.

We intend to publish a specification of the recognizer interface and source code for a sample recognizer to encourage other developers to produce recognizers. Recognizers could be customized for "Intranet" access to information at particular organizations.

## CONCLUSIONS

We have presented the Selection Recognition Agent (SRA), an application with an extremely simple user interface that provides immediate and direct access to relevant applications as well as the relevant portions of massive stores of information. Although the SRA integrates a number of simple and well-known technologies, we know of no other application that is equally powerful, widely available, and sufficiently integrated with the operating system to be usable with any other application.

We have shown that the SRA provides a significant savings of both time and effort in accessing information relevant to meaningful text objects. Furthermore, the SRA provides access to the relevant portions of huge numbers of documents at minimal cost.

## ACKNOWLEDGEMENTS

Thanks to Marc Millier and Dave Cobbley for their support in the development and deployment of the Selection Recognition Agent.

## REFERENCES

1. Card, S. K., Pirolli, P., Mackinlay, J., "The Cost-of-Knowledge Characteristic Function: Display Evaluation for Direct-Walk Dynamic Information Visualizations", *Proceedings of CHI 94*, Boston, MA, April 1994
2. Cypher, Allen, et. al., editors, *Watch What I Do: Programming by Demonstration*, 1993, The MIT Press, Cambridge, Massachusetts
3. Cypher, Allen, "Eager: Programming Repetitive Tasks by Example," *Proceedings of CHI '91*, April 28–May 2, New Orleans, LA, pp. 33-39
4. Rhodes, Bradley J., Starner, Thad, "Remembrance Agent: A Continuously Running Automated Information Retrieval System," Presented at AAAI-Spring Symposium '96: Acquisition, Learning and Demonstration: Automating Tasks for Users
5. Schlimmer, J. C., Hermens, L. A., "Software Agents: Completing Patterns and Constructing User Interfaces," *Journal of Artificial Intelligence Research*, November 1993
6. *The Windows Interface Guidelines for Software Design*, Copyright © 1995, Microsoft Corporation

## APPENDIX

This appendix enumerates, for various tasks, the operations required with and without SRA.

**Acquire info on Detroit, MI**

| Without SRA | With SRA |
|---|---|
| Locate browser | Select City |
| Start browser | Select SRA menu item |
| Direct to CityNet home page | |
| Click on country | |
| Click on state | |
| Click on city | |

**View 7/1/1996 in calendar program**

| Without SRA | With SRA |
|---|---|
| Select date | Select date |
| Select copy menu item | Select SRA menu item |
| Locate calendar program | Select go to date menu item |
| Start calendar program | Select paste menu item |
| Select go to date menu item | |
| Select paste menu item | |

**Send mail to milind_s_pandit@ccm.jf.intel.com**

| Without SRA | With SRA |
|---|---|
| Select email address | Select email address |
| Select copy menu item | Select SRA menu item |
| Locate email program | Select prepare message menu item |
| Start email program | Select paste menu item |
| Select prepare message menu item | |
| Select paste menu item | |

**Enter (503) 264-8880 in phone book program**

| Without SRA | With SRA |
|---|---|
| Select phone number | Select phone number |
| Select copy menu item | Select SRA menu item |
| Locate phone book program | Select paste menu item |
| Start phone book program | |
| Select paste menu item | |

**View definition of "apropos"**

| Without SRA | With SRA |
|---|---|
| Select word | Select word |
| Locate browser | Select SRA menu item |
| Start browser | |
| Direct browser to online dictionary | |
| Select paste menu item | |
| Start search | |

**View FAQs regarding tennis**

| Without SRA | With SRA |
|---|---|
| Locate browser | Select word |
| Start browser | Select SRA menu item |
| Direct browser to rtfm.mit.edu/pub/usenet-by-name | |
| Search for newsgroups mentioning tennis | |
| Click on newsgroup name | |
| Retrieve FAQ | |

**Perform web search on "agents"**

| Without SRA | With SRA |
|---|---|
| Select word | Select word |
| Locate browser | Select SRA menu item |
| Start browser | |
| Direct browser to search engine | |
| Select paste menu item | |
| Start search | |

**Visit http://www.intel.com in running browser**

| Without SRA | With SRA |
|---|---|
| Select site | Select site |
| Locate browser | Select SRA menu item |
| Select paste menu item | |
| Press enter | |

**Visit http://www.intel.com in new browser**

| Without SRA | With SRA |
|---|---|
| Select site | Select site |
| Locate browser | Select SRA menu item |
| Start browser | |
| Select paste menu item | |
| Press enter | |