an `accessibilityName` or `accessibilityDescription`. For the Button element—as with many Flex elements—the accessibility information is automatically set when the proper visual label is provided. In practice, developers generally do not need to provide more information regarding the accessibility of the button, which simplifies any localization or modification of the label in the future.

Additional properties

In addition to name and description, there are a number of other properties supported by the AccessibilityProperties class:

- **shortcut** — A string indicating the shortcut keystroke associated with a component. Setting this property does not bind the actual keyboard shortcut, but simply exposes the string to assistive technologies. The convenience accessor for this property is `UIComponent.accessibilityShortcut`; for example:

```
// ActionScript
btnSearch.accessibilityShortcut = "Alt+S";
// MXML
<s:Button label="Search" id="btnSearch" accessibilityShortcut="Alt+S" />
```

- **silent** — A Boolean property that controls whether the accessibility implementation for this component is enabled or disabled. If set to `true`, no accessibility information is exposed for this component instance. This can be useful when the component is hidden off-screen or behind another component. The convenience accessor for this property is `UIComponent.accessibilityEnabled`. Note that the Boolean logic for the convenience accessor is the opposite for the actual property. Setting `accessibilityEnabled` to `false` will hide the accessibility implementation for the component instance, whereas setting `silent` to `true` achieves the same effect. By default, accessibility information is not exposed for components that are rendered out of the Flex application's coordinates or for those that have the `visible` property set to `false`.

```
//ActionScript
btnNextPageSearchResults.accessibilityEnabled = false;
// MXML
<s:Button id=”btnNextPageSearchResults” label=”Next” accessibilityEnabled=”false” />
```

- **forceSimple** — A Boolean property that controls whether child objects have accessibility exposed. There is no convenience accessor for this property, as it is typically used in creating accessibility implementations in Flash and is less likely to be used in Flex applications unless a custom accessibility implementation is created.

Note: When the `AccessibilityProperties` object's properties are changed, a call to `Accessibility.updateProperties()` is required to inform assistive technologies that a change has occurred and that any cached information should be marked as invalid. If a convenience accessor is used, this method need not be called. Because it can be processor intensive for assistive technology to rebuild an accessible view of the Flex application, developers are encouraged to call this method only after all accessibility property changes for a given action have been made.

If `updateProperties()` is called on a platform that does not support accessibility, unpredictable results may occur. Developers must first check to see if the platform supports accessibility by importing `flash.system.Capabilities` and checking the Boolean `hasAccessibility` property; for example:

```
import flash.system.Capabilities;
...
if (Capabilities.hasAccessibility)
  {
    // set accessibility properties and call updateProperties();
  }
```

Setting component name information

The primary accessibility requirement for Spark and MX components is properly setting and exporting an accessible name. While this information can be set directly using the `accessibilityName` property (a technique explained above), generally it will be provided automatically by the platform when the developer sets a text label (either on or near the element) or sets a tooltip for the element.