provided as part of the accessible name for the field. If standard Flex form validation is used, this text will automatically be appended to the `accessibilityName` property of the form field. If the standard Flex form validation is not used, developers can explicitly set this information in the `accessibleName` property of each form field in error.

Note: If field-specific information is provided, it should be provided *in addition* to providing an overall indication of the form error state. This ensures that users of assistive technology will be made immediately aware that the form is in error and will have specific instructions on how to fix each error in the form.

```
// import the form validator class
import mx.validators.Validator;
function validateForm(e:MouseEvent):void
{
  // validate the fields requiring validation on submit
  var validatorArray:Array = Validator.validateAll(a1);
  // if no errors found submit data, otherwise focus the error message (not shown)
}
...
<mx:FormItem label="I agree to the license agreement" required="true">
  <mx:CheckBox id="agreement"/>
</mx:FormItem>
...
<mx:Array id="a1">
  <mx:StringValidator source="{agreement}" required="true" property="selected"
     maxLength="4" requiredFieldError="You must accept the license agreement."
      tooLongError="You must accept the license agreement."/>
<mx:Array>
...
<mx:Button label="Submit" click="validateForm(event)" />
```

Real-time form validation

Some Flex forms are designed with internal validation checks that respond to user input in real time as the user is completing the form. These applications immediately prompt the user with a visible or audible indication when a validation error—such as entering letters in a field that requires numerical input—is detected. Some applications also move the focus to the next input field when the user has entered data consistent with the rules. Often, real-time feedback can improve accessibility. However, real-time validation raises some accessibility challenges not present in forms that are validated upon submission. See, for example, "Avoid forced focus changes" on page 39 for guidance on moving the focus automatically.

When audio is used to indicate an error in a form, additional on-screen text should be provided for users who cannot hear or do not have access to audio. For example, producing an audible beep when the phone number length exceeds ten characters should be accompanied by explanatory error text:

```
...
lblPhoneError: Label = new Label();
lblPhoneError.text = "The phone number can only be 10 characters";
// visual placement not shown
lblPhoneError.tabIndex = 12;
lblPhoneError.visible = false;
addChild(lblPhoneError);
...
txtPhoneNumber.addEventLisentener(TextEvent.TEXT _ INPUT, checkLenght);
function checkLength(e : TextEvent): void
{
  if (txtPhoneNumber.length > 10)
  {
    // produce beep (code not shown)
```