You can use the Kentico API to manage the data that users submit through forms. Two different approaches are available:

- Work with existing form data – load or modify the data stored for any form.
- Handle form actions directly when they occur – use event handlers to execute code before or after form actions occur in the system (for example whenever users submit records for a form).

Working with existing form records

Use the following approach to access the data of a form at any time:

1. Get a **BizFormInfo** object representing the form itself (call the *BizFormInfoProvider.GetBizFormInfo* method).
2. Get the form's class name (use the *DataClassInfoProvider.GetDataClassInfo* method with the form's class ID as the parameter, access the *ClassName* property of the *DataClassInfo* object).
3. Call one of the following methods to get the form's data:
    - **BizFormItemProvider.GetItems(class name)** – loads all of the given form's records (or a filtered subset). You can loop through the *BizFormItem* objects in the data.
    - **BizFormItemProvider.GetItem(item ID, class name)** – gets one *BizFormItem* record specified by ID.

To get the values of specific fields from *BizFormItem* objects, call the *GetValue* methods – either the general *GetValue,* which returns an object, or methods for specific data types, such as *GetStringValue.*

To set values for fields, use the *SetValue* method for individual fields and then call *SubmitChanges* for the *BizFormItem* object to save the results.

Example

```
using CMS.OnlineForms;
using CMS.DataEngine;
using CMS.SiteProvider;
using CMS.Helpers;

...

// Gets the form info object for the 'ContactUs' form
BizFormInfo formObject = BizFormInfoProvider.GetBizFormInfo("ContactUs", SiteContext.
CurrentSiteID);

// Gets the class name of the 'ContactUs' form
DataClassInfo formClass = DataClassInfoProvider.GetDataClassInfo(formObject.
FormClassID);
string className = formClass.ClassName;

// Loads the form's data
ObjectQuery<BizFormItem> data = BizFormItemProvider.GetItems(className);

// Checks whether the form contains any records
if (!DataHelper.DataSourceIsEmpty(data))
{
        // Loops through the form's data records
        foreach (BizFormItem item in data)
        {
                string firstNameFieldValue = item.GetStringValue("FirstName", "");
                string lastNameFieldValue = item.GetStringValue("LastName", "");

                // Perform any required logic with the form field values

            // Variable representing a custom value that you want to save into the
form data
                object customFieldValue;

                // Programatically assigns and saves a value for the form record's
'CustomField' field
                item.SetValue("CustomField", customFieldValue);
                item.SubmitChanges(false);
        }
}
```

Handling form events

To perform custom actions directly when form actions occur, implement handlers for BizFormItemEvents events.

You need to assign handlers for the events at the beginning of the application's life cycle – create a custom module class and override the module's **OnInit** method.

To access the data of the active form in the event handler, use the **Item** property *(BizFormItem* type) of the **BizFormItemEventArgs** parameter. The parameter is available for all form item event handlers.

Example

```
using CMS;
using CMS.DataEngine;
using CMS.OnlineForms;

// Registers the custom module into the system
[assembly: RegisterModule(typeof(FormHandlerModule))]

public class FormHandlerModule : Module
{
        // Module class constructor, the system registers the module under the name
"CustomFormHandlers"
        public FormHandlerModule()
                : base("CustomFormHandlers")
        {
        }

        // Contains initialization code that is executed when the application starts
        protected override void OnInit()
        {
                base.OnInit();

                // Assigns a handler to the BizFormItemEvents.Insert.After event
                // This event occurs after the creation of every new form record
                BizFormItemEvents.Insert.After += FormItem_InsertAfterHandler;
        }

        // Handles the form data when users create new records for the 'ContactUs' form
        private void FormItem_InsertAfterHandler(object sender, BizFormItemEventArgs e)
        {
                // Gets the form data object from the event handler parameter
                BizFormItem formDataItem = e.Item;

                // Checks that the form record was successfully created
                // Ensures that the custom actions only occur for records of the
'ContactUs' form
                // The values of form class names must be in lower case
                if (formDataItem != null && formDataItem.BizFormClassName == "bizform.
contactus")
                {
                        string firstNameFieldValue = formDataItem.GetStringValue
("FirstName", "");
                        string lastNameFieldValue = formDataItem.GetStringValue
("LastName", "");

                        // Perform any required logic with the form field values

                        // Variable representing a custom value that you want to save
into the form data
                        object customFieldValue;

                        // Sets and saves a value for the form record's 'CustomField'
field
                        formDataItem.SetValue("CustomField", customFieldValue);
                        formDataItem.SubmitChanges(false);
                }
        }
}
```