This page describes how to resolve common errors that you may encounter when working with custom modules.

Modules missing in the application list

Problem: You have created, imported or installed a custom module that has its own administration pages, but the module's application is missing in the application list. The problem may also occur with existing custom modules after you create or import a new site.

Solutions:

- You may need to **reload** the entire Kentico administration page in your browser, including the header section.
- Make sure the module is assigned to the currently active site:
  1. Open the **Modules** application in the Kentico administration interface.
  2. Edit your module.
  3. Switch to the **Sites** tab and assign the module to your sites.

     Tip: Alternatively, you can edit your sites in the **Sites** application and assign modules on the **Assigned objects** tab.
- For users without the Administrator or Global administrator privilege level, you may need to ensure that the module has the **Read** permission defined and that the user has the given permission assigned.

See also:

- Creating custom modules
- Configuring permissions

Custom object types (classes) missing in selectors

Problem: You cannot find custom object types (module classes) in selectors within the administration interface. For example when setting the **Object type** in the properties of UI elements or the **Reference to** value in the settings of foreign key class fields.

Solutions:

- If you are the developer of the custom module, make sure that you have generated the *Info* and *InfoProvider* code for the class. On web application type projects, you also need to include the code files into the project in Visual Studio and build the solution.
- If your module code files are placed in a separate library, you need to allow the system to detect the classes. Make sure the **AssemblyDiscoverable** assembly attribute is added (typically in the *AssemblyInfo.cs* file of the custom project).
- You may be missing the resource string representing the given object type name. If the resource string does not exist, the object type is available under the default name in format: ***ObjectType.<class code name with an underscore>***

See also:

- Creating custom modules
- Creating custom binding classes
- Adding references between classes