Daminion Export API 0.9.1

Daminion Export API allows you to expand your Daminion export capabilities by writing export add-ons. The standard export features are: Export to Folder (Export > Copy to Folder) and Email (Export > Send to Email).

Supported Languages

Daminion Add-ons can be written in any .NET compliant language, including C# or VB .NET.

Creating a New Add-on (C#)

First Steps

- Launch Visual Studio and create a new Class Library.
- Specify the .NET 2.0 or .NET 3.5 as the Target Framework.
- Add the Daminion.API.dll from Daminion Export SDK.

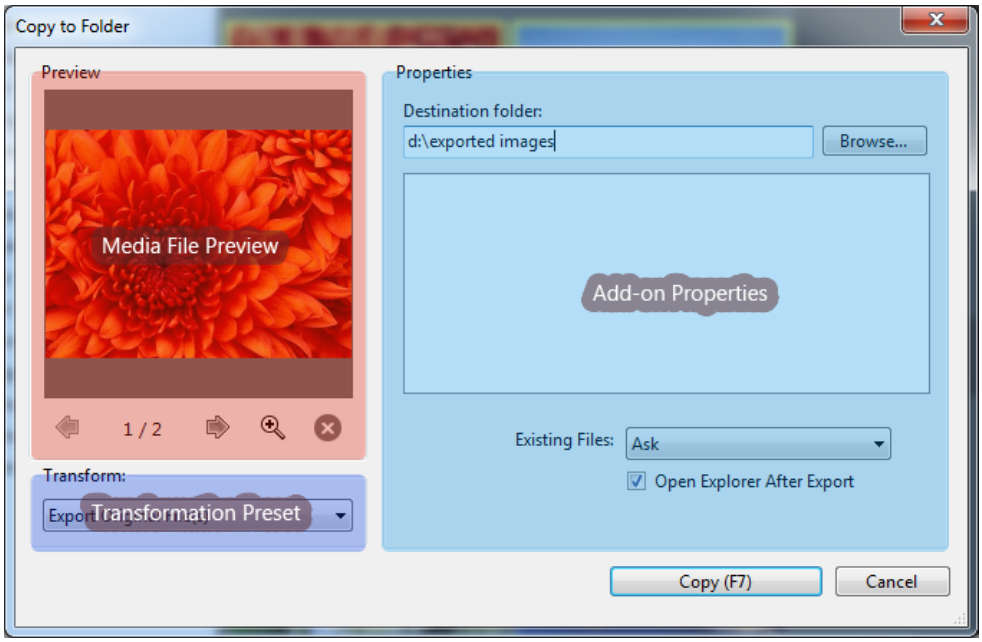
Your add-on must:

1. Be inherited from the BaseExportAddon, the base class for all the export add-ons.
2. Implement the IAddonInfo interface

```
public class MyExportAddon : BaseExportAddon, IAddonInfo
{
    // add-on implemenation
}
```

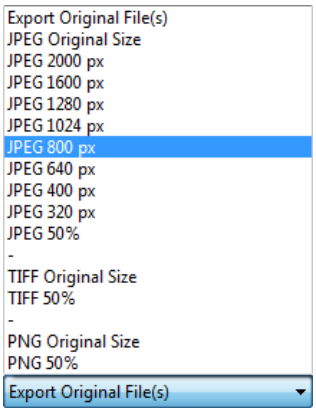
UI

All export add-ons are launched inside the same Export window with preview options.



"Copy to Folder" add-on

Users can select a Transformation Preset before exporting the files. By default JPEG, TIFF or PNG based presets are available.



Transformation Presets

If your add-on needs some add-on customization, you can add a Property Panel with the necessary controls to the Export Form. See "Copy to Folder" screenshot above.

To add this panel you need to override the virtual method GetPropertyPanel of the BaseExportAddon class. Below is the code fragment from the "Copy to Folder" add-on.

```
public override UserControl GetPropertyPanel(string settingsAsXml)
{
  if (this.propertyPanel == null)
  {
    this.propertyPanel = new CopyToFolderPropertyPanel();
    this.Settings = this.DeserializeSettings(settingsAsXml) as
CopyToFolderSettings;
    if (Settings == null)
    {
      Settings = new CopyToFolderSettings();
    }

    this.propertyPanel.RecentFolders = this.Settings.RecentFolders;
    if (this.propertyPanel.RecentFolders.Length > 0)
    {
      this.propertyPanel.DestinationFolder =
this.propertyPanel.RecentFolders[0];
    }
    this.propertyPanel.CResolverOptions =
this.Settings.CollisionResolverOptions;
    this.propertyPanel.OpenExplrAfterExport =
this.Settings.OpenExplrAfterExport;
  }
  return propertyPanel;
}
```

Save Property Panel state

Daminion allows you to store each export add-on setting in a single Settings Repository. To do this you will need to override the following two methods:

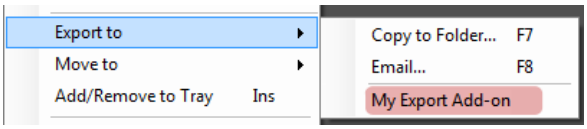
- `public override object DeserializeSettings(string settingsAsXml)`
- `public override string SerializeSettings()`

Deployment

All the Daminion add-ons are stored in the **[Program Files]\Daminion Software\Daminion\Addons** folder.

Build your add-on project and copy the compiled .dll file to the Addons folder. Do not include theDaminion.API.dll!

After doing this please restart Daminion. If you've done everything correctly your add-on will appear in the Export Menu List.



If you think your export add-on might be useful to other users, you're welcome to send the compiled project to us at addons@daminion.net

It will be published on our Daminion Add-ons page and available for public download after our review.

IAddonInfo Interface

Information about Daminion add-ons

Namespace: Daminion.API
Assembly: Daminion.API.dll

Properties

string	Name	Add-on Name
Guid	Guid	Add-on Guid
string	WhatsNew	What's new in this add-on version
string	Description	Add-on Description
string	Copyright	Add-on Developer Copyrights
string	Version	Add-on Version
string	Authors	Add-on Developer(s)
string	MinimalVersion	Minimal Daminion version needs to launch this add-on
string	UpdateUrl	Url to check the latest add-on versions
string	HomePage	Add-on HomePage

BaseExportAddon

Base class for all Daminion export add-ons.

Namespace: Daminion.API
Assembly: Daminion.API.dll

Abstract Properties & Methods

string	DoExportFile	Do export the specified file **Parameters:** • **fileRef** - Reference to a file on the local disk • **sourceFileName** - File name of the source file

		• **cancelExport** - Return False to cancel the export process • **skipThisFile** - Return True to cancel the export process • **catalogTags** - A list of catalog tags associated with the exporting file

Virtual Properties & Methods

string	CustomExportButtonText	A replacement for the default Export Button caption of the Export Preview Form. Example of usages: "Send to Email", "Upload to Flickr".
UserControl	GetPropertyPanel	This panel will appear inside the Properties GroupBox of the Export Preview Form (you will not seen this GroupBox if you don't override this method). Override this method if your addon allows customization. **Returns:** Reference to a panel (inherited from UserControl) with controls to customize addon properties.
DefaultTransformationPresets	PreferredTransformationPreset	This transformation preset will be selected when a user first starts the addon.
void	DoLaunchJobInMainThread	Launch a job in the main thread. Useful to launch some Forms in the main thread.
void	DeserializeSettings	Deserialize Addon settings from the xml string. **Parameters:** • **settingsAsXml** - Addon settings is stored as xml in the Single Settings Repository. **Returns:** Instance of a User class deserialized from the specified settingsAsXml.
object	SerializeSettings	Serialize add-on settings to the xml string **Returns:** XML string of the serialized Add-on Settings instance.
bool	CheckInputBeforeExport	Invoked right after a user clicks on the Export button **Returns:** • Return **False** if input was invalid. In this case the Export Form will not be closed after pressing the Export button. • Return **True** to continue the export process.
void	FinishExport	Override this method if you need to perform any actions after the export process has finished.