What is an Add-On?

Add-On is any third party software that wants integration with Alibre. This add-on gets launched from within Alibre when the end user wants to use the third party product. This add-on can be of one of the following categories:

1. Complete stand-alone Add-On
2. Add-On using Alibre CAD data
3. Add-On completely integrated with Alibre

Depending on what your needs are, you can choose one of the above options. The one common thing that is shared by all of them is that, add-on is always launched from within the running instance of Alibre. When an add-on is created one can find that Add-on on Alibre's Home Window under Tools-> Add-Ons-> [Name Of the Add-On]. The add-on can be launched from this place by selecting the add-on you want to launch. Also the add-on infrastructure is dependent on the Microsoft's **COM (Component Object Model)** technology**.** So it is highly recommended to have a basic knowledge of COM technology before dealing with the Add-Ons. COM based infrastructure gives user the flexibility to create his/her product in any of the COM supporting languages (e.g. Visual C++, Visual Basic, Microsoft Java etc.). Having said that, lets see one by one, how each of the above categories of add-ons differs from each other.

Complete stand-alone Add-On

This is the simplest of the three types. When user clicks on the Tools->Add-ons->[Name of the Add-On] this launches the Add-on (in a separate GUI if provided by add-on or does whatever the app is created for). An example of this could be an add-on that just opens a new Internet browser. It doesn't use any of the Alibre API's as no CAD data is queried from the product.

Add-On using Alibre CAD data

This is fundamentally similar to the above type. The main difference being, though this also launches the add-on in a stand-alone mode it uses Alibre API's to query the CAD data from Alibre. It then uses this CAD data in its format to do whatever it is intended to do.

Add-On completely integrated with Alibre

These types of add-ons are more involved from programming point of view, but on the other hand they give the user the flexibility to integrate the add-on completely within Alibre product. By this it means

- One can expose his/her own menu structure in the Alibre menu and the product is launched in Alibre's GUI.
- A new tab window is created for the add-on along with the design explorer of Alibre. Add-on user can create his/her own controls on this tab window.
- Also the add-on gets notified of the various events such as keyboard and mouse events or the events such as when user selects a part or a face in Alibre.
- The add-on can save its own data in the Alibre file (part, assembly, drawing) file as a separate data stream. And this data can be loaded when next time Alibre is launched.

Under the Hood

Programming Side:

On programming front also these three types of add-ons share one basic infrastructure that is discussed below. If you want to create one of the first two types of add-ons this is all that you need to read and you should be ready to go!

Before getting into how Alibre recognizes the Add-on application and how it loads the add-on applications lets start with how one can create an add-on application. An add-on application is a simple **dll** application that gets launched when Alibre is launched (and the pre-requisites, that will be discussed in due course, are taken care of before hand).

To begin with, create a simple Win32 Dynamic Link Library (Visual Studio 6.0 using VC++). In the main header file of the application following are the functions that would be needed to create an add-on.

Three important functions

These are the prototypes of the main functions that need to be present in any add-on application irrespective of its type.

```
1. void AddOnLoad (HWND windowHandle,
                   VOID *pAutomationHook,
                   VOID *reserved);

2. void AddOnInvoke (HWND windowHandle,
                     VOID *pAutomationHook,
                     LPCSTR sessionName,
                     BOOL isLicensed,
                     VOID *reserved1,
                     VOID *reserved2);

3. void AddOnUnload (HWND windowHandle,
                     BOOL forceUnload,
                     BOOL *cancel,      // set TRUE to cancel
                     VOID *reserved1,
                     VOID *reserved2);
```

1. **AddOnLoad**

   As the name suggests this function this function gets called when the end user launches Alibre.

   Arguments Description:

   HWND windowHandle:

   This is a handle to the main window that is passed to the AddOn application on the launch.

   *VOID *pAutomationHook:*

   This is the **most important** interface pointer as far as the add-on integration is concerned. As the name suggests this pointer is the hook into the Alibre running object. Anything and everything that needs to be queried from Alibre has to start from this pointer. For first type of Add-On this would be insignificant as the add-on doesn't use the CAD data from Alibre or doesn't get connected to Alibre.

2. **AddOnInvoke**

   This function is called when user selects the Add-On from the Tools->Add-Ons either on the home window or from the session that is opened.

   Arguments Description:

HWND windowHandle:

Refer to AddOnLoad

*VOID *pAutomationHook:*

Refer to AddOnLoad

LPCSTR sessionName:

If the Add-on is invoked from the Home window this parameter will be an empty string. But if the add-on is invoked from any of the open session, name of that session is passed as a parameter to the function.

BOOL isLicensed:

Depending upon whether the user has a valid license for the add-on this flag is set. Add-on developer can use this flag to allow or block the user from using the add-on. Or they can simply neglect it.

*VOID *reserved1*
*VOID *reserved2*

These are the reserved parameters and should always be NULL.

3. **AddOnUnload**

HWND windowHandle:

Refer to AddOnLoad

BOOL forceUnload:

This parameter will notify the add-on if the Add-on is getting unloaded forcibly because of some unexpected failure or not.

*BOOL *cancel*

This parameter can be changed by the add-on to notify Alibre whether to cancel unloading of the add-on or not. If returned true add-on will not be loaded. Default value for this variable would be false.

*VOID *reserved1*
*VOID *reserved*

These are the reserved parameters and should always be NULL.

<u>Integration Side:</u>

For making Alibre aware of the presence of add-on and so that Alibre can load the dll for the add-on there are some important guidelines that need to be followed. This section deals with that integration aspect of the add-on. It is required to have a basic knowledge of the windows registry, how to create a key in the registry, to understand the procedure completely.

Each add-on folder on the end-user's machine will typically resemble the following structure.

Main folder with name of the Add-On

[Add-On Name] .adc file (AddOn Configuration File)
[Add-On Name] .ico file
[Add-On Name] .dll file

A folder named Data (if add-on is of the third category)

ADC File

.adc file is read when Alibre loads the add-on. This is just an XML file with extension .adc. This needs to be created by the add-on developer. A typical .adc file is shown on the next page.

```xml
<AlibreDesignAddOn specificationVersion="1" friendlyName="My AddOn">
  <Author name="MyName" link="http://URL"/>
  <DLL loadedWhen="Startup" location="MyAddOnName.dll"/>
  <Copyright> My CopyRight Information</Copyright>
  <Icon location="MyAddOn.ico"/>
  <Menu text="My AddOn"/>
  <Description> XYZ </Description>
  <Workspace type="Part"/>
  <Property name="Identifier" value="{DA8322C2-CE32-450C-2E43-5CG77C52D4B4}"/>
</AlibreDesignAddOn>
```

<AlibreDesignAddOn>

specificationVersion –

You can update this as and when a newer version of the add-on is developed

FriendlyName -

This is just a friendly name for the add-on and not used anywhere else.

<Author>

AuthorName –

Name of the owner of the Add-On.

Link –

Link to website for the Add-On.

<DLL>

Lodedwhen –

This tells Alibre when to load the add-on dll. There are two options to specify i.e. **“StartUp”** or **“Invoke”.** This means the add-on will either be loaded when Alibre is launched or when user actually selects the add-on under Tools->Add-Ons.

Location –

This specifies the location of the dll on the end user’s machine where the dll for this Add-on can be found. This is the responsibility of the Add-on developer to lay the dll file on the user’s machine on the same path as mentioned here. Though any path is valid as long as it exists its always good to have the dll in the same folder as .adc file.

<Copyright>

Copyright information if any.

<Icon location>

Path of the icon file for the Add-on if any.

<Menu text>

Name of the Add-On as you want it to appear on the Alibre menu.

<Description>

A brief description about what this add-on does.

<Workspace>

Type of the workspace for the Add-On. The available choices are:

“Part”, “Drawing”, “SheetMetal”, “Repository”, “Always” (for any workspace).

<Property name>

Identifier -

This is a unique identifier for the add-on. The number is a GUID (Globally Unique Identifier). This needs to be generated by the Add-on developer.

Depending on whether the add-on is licensed or not this ADC file will be used to load the Add-on when Alibre is launched.

ICO file

This file is the icon file created for the add-on. The path of this file is provided in the adc file. This icon is shown besides the Add-On name under the list of add-ons. It’s up to the Add-on developer whether he/she wants to have an icon with the add-on.

DLL File

This is the main add-on dll file which is loaded at the time of launch of Alibre or during the invoke of the add-on (depending on what adc file contains.) The path of the dll is also mentioned in the adc file and its the responsibility of the add-on developer to have a right path in the adc file and then have it laid at the right location on the user's machine.