

**DropMind**<sup>™</sup>  
*picture your thoughts*

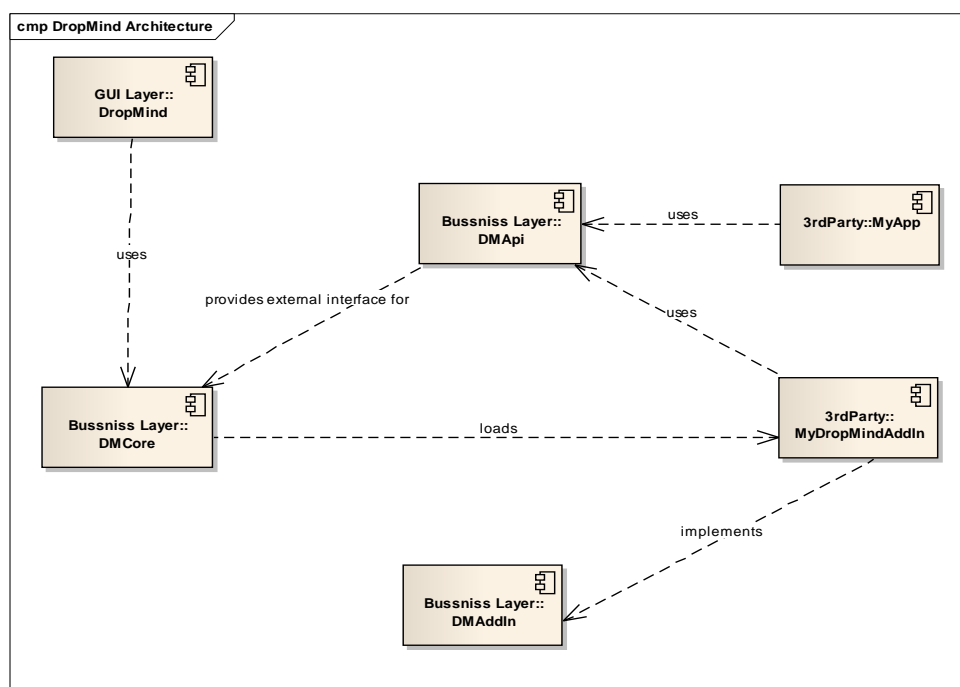
# API Overview

## **CONTENTS**

<b>1</b>	<b><u>DROP MIND ARCHITECTURE OVERVIEW</u></b>	<b>3</b>
<b>2</b>	<b><u>DROP MIND API</u></b>	<b>5</b>
<b>3</b>	<b><u>HOW TO USE DROP MIND FROM YOUR APPLICATION</u></b>	<b>7</b>
<b>4</b>	<b><u>HOW TO CREATE DROP MIND ADDIN</u></b>	<b>8</b>

# 1 DROPMIND ARCHITECTURE OVERVIEW

Seavus Desktop DropMind is a dynamic mind mapping software tool for visual thinking, brainstorming, planning and organizing in real time. Seavus DropMind helps map out information and improve memorization, productivity and creativity. It is great for solving problems, improving communication, and making plans quicker and more creatively. With its smooth usability and highly interactive interface, it saves time, energy and effort, and improves every aspect of your work. Seavus DropMind™ allows capturing all important information on a simple, easy-to-use, cross-platform friendly mind map.



**Figure 1:** DropMind Architecture

DropMind is composed of the following components (component diagram is presented into a Figure 1):

- *DropMind* – DropMind GUI module;
- *DMCore* – DropMind core component. It contains classes for working with map documents, map elements and allows access to the DropMind libraries (icons, images, shapes, etc.);
- *DMApi* – provides interface to the DMCore classes. It wraps DMCore module and gives the possibility to develop, extend and use already implemented features of DropMind for managing DropMind documents. The API is implemented in C++/Qt4 (see details in §2);
- *DMAAddn* – contains abstract and helper classes which have to be implemented by each addin library you are developing (see details in §4);



- *MyApp* – 3th party application - you can use DropMind object model for creation/managing of map documents from your application (see DropMind API documentation, located in DropMind SDK folder). See details for using of DropMind in other applications in §3.
- *MyDropMindAddIn* – 3rd party modules which can extend DropMind functionalities. AddIn modules are loaded by DropMind during runtime and are allowed to perform various actions within DropMind (for example: export to word, cvs, etc...). See details for creation of AddIn libraries in §4.

## 2 DROPMIND API

DropMind API module allows creation/managing of map documents using DropMind object model, without knowing exact structure/format of DropMind map document file. Globally DropMind API classes are divided in two groups. First group of classes is used for accessing of DropMind resources (documents, icons, images, shapes, etc...) and second group of classes is used for managing of map document.

In the following table are given classes used for accessing of DropMind resources:

Class	Description
<b>IDMApplication</b>	Application object interface. You can access icon, image, shape library or to access container with all documents via this interface.
<b>IDMIconLibrary</b>	Icon library object interface. This interface allows access to the DropMind icons and icon groups.
<b>IDMImageLibrary</b>	Image library object interface. This interface allows access to the DropMind images and icon groups. This interface also allows managing of DropMind image library (you can add, remove or rename image/image group from DropMind image library via this interface)
<b>IDMLayoutLibrary</b>	Layout library object interface. This interface allows access to the DropMind layouts.
<b>IDMMapStyleLibrary</b>	Map style library object interface. This interface allows access to the DropMind map styles.
<b>IDMShapeLibrary</b>	Shape library object interface. This interface allows access to the DropMind shapes. This interface also allows managing (adding/removing) of shapes from this library.

In the following table are given classes used for managing of DropMind map document:

Class	Description
<b>IDMMapDocument</b>	Document object interface.
<b>IDMMapSelection</b>	Map selection object interface.
<b>IDMAttachment</b>	Attachment object interface.
<b>IDMBoundary</b>	Boundary object interface.
<b>IDMBoundaryStyle</b>	Boundary style object interface

<b>IDMCallout</b>	Callout topic object interface
<b>IDMIcon</b>	Icon object interface
<b>IDMIconGroup</b>	Icon group object interface
<b>IDMIconMarker</b>	Icon marker object interface.
<b>IDMImage</b>	Image object interface
<b>IDMImageGroup</b>	Image group object interface
<b>IDMLayout</b>	Layout object interface
<b>IDMMapElement</b>	Map element object interface
<b>IDMMapStyle</b>	Map style object interface
<b>IDMMapStyleGroup</b>	Map style object interface
<b>IDMPresentation</b>	Presentation object interface
<b>IDMRelationship</b>	Relationship object interface
<b>IDMRelationStyle</b>	Relationship style object interface
<b>IDMShape</b>	Shape object interface
<b>IDMShapeGroup</b>	Shape group object interface
<b>IDMSlide</b>	Slide object interface
<b>IDMTopic</b>	Topic object interface
<b>IDMTopicStyle</b>	Topic style object interface

**Note:** You can find more details about these classes into DropMind SDK documentation.

### 3 HOW TO USE DROPMIND FROM YOUR APPLICATION

The developer is able to create DropMind documents without starting the DropMind application. This functionality will allow creation of map documents using object model. With this, third party application will be able to create documents without knowing exact format of the file. API allows just creation of document, for viewing of the created document the DropMind application has to be used.

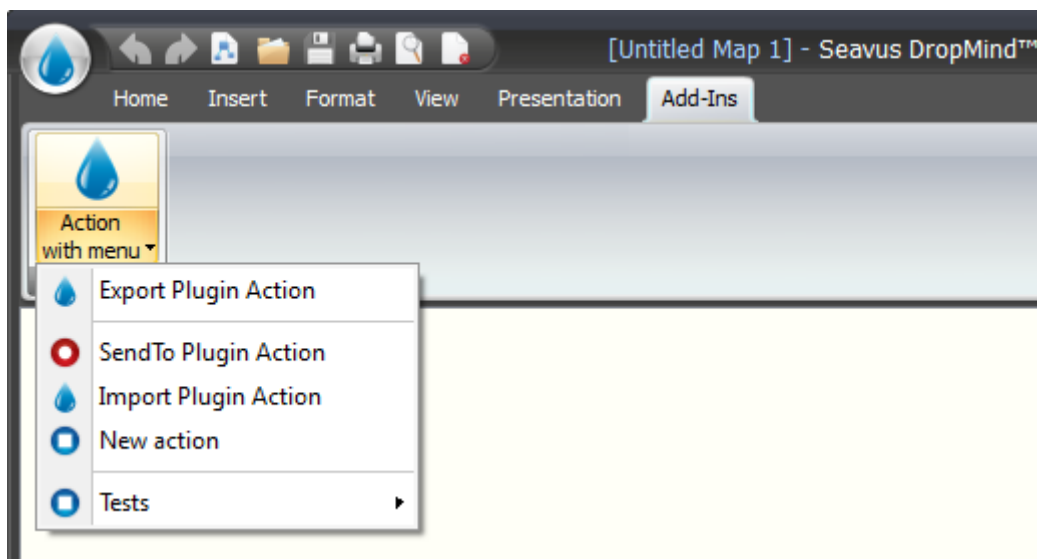
In order to use DropMind API in your application you should perform the following steps:

- Link DropMind API module into a your project,
- Include `dmapi.h` into your project,
- Initialize DropMind via `IDMApplication::init()` method before calling of any DropMind API function,
- Release DropMind resources via `IDMApplication::release()` method after using of DropMind API.

**Note:** Example for using of DropMind API from 3rd party applications you can find into a `DMApiTestsQT` project, located in `DM_SDK\examples` folder.

## 4 HOW TO CREATE DROPMIND ADDIN

Add-in libraries are shared libraries which can be integrated in DropMind application and are allowed to execute various actions within application. AddIn libraries are allowed to add its actions into a DropMind main menu (Import, Export or Send sub-menu) or in Add-In category from ribbon panel (as it is presented in the **Figure 2**). Actions from ribbon menu can be simple or composed from several sub actions. Simple actions are presented as simple ribbon buttons in ribbon panel; composed actions are presented as ribbon button with popup menu (as it is presented in picture below).



**Figure 2:** Add-Ins ribbon panels

In order to create DropMind AddIn you should perform the following steps (see `TestAddIn` project from DM SDK examples):

- Create new shared library project and link it with `DMAddIn` and `DMApi` modules,
- Include `dmapi.h` and `dmaddin.h` interfaces into your project,
- Implement `IDMAddIn` class and export it via `DMADDIN_CLASS_EXPORT` macro,

*Example:*

```
class TestAddIn
    : public QObject
    , public IDMAAddIn
{
    Q_OBJECT

public:
    TestAddIn();
    ~TestAddIn();

    QString name() const;
    QString version() const;
    QString description() const;
```

- Create instances from `MenuAction` class for each action which you like to add in DropMind application. You can add AddIn commands into main menu and/or in Add-ins ribbon panel. Use the following the rules in order to add your action in appropriate location from DropMind application:
  - If you like to add your action (simple or composite) in DropMind ribbon menu then return action via `IDMAAddIn::categoryPanelAction` method.
  - If you like to add your action in main menu then return it via `IDMAAddIn::mainMenuAction` method. `IDMAAddIn::mainMenuAction` is called three times in row, for each item from `ActionType` enumeration type. Check type parameter from `IDMAAddIn::mainMenuAction` method in order to add your action in appropriate submenu.
- Pass information to DropMind application about AddIn module via `name()`, `version()` and `description()` methods. The values which you will return via these methods will be presented in DropMind dialog.

After creation of AddIn module, you should copy into DropMind AddIn folder `%DMApplicationDir%\addins`. DropMind will load all AddIn libraries which are located in this folder during runtime. Information about loaded AddIn modules can be found in DropMind about dialog (DropMind options > About DropMind > Available Addins section).