articy:draft API - Programmer's Guide

Table of contents

1. Overview

1.1. What is articy:draft API?

The articy:draft API allows an external application to connect to an articy:server, open a project and work with the elements of this project. It is very easy to write a program that searches certain articy:draft objects, lists or exports specific values or even write those values back into the articy:draft project. This API allows articy:draft to be integrated into a complex tool chain

1.2. The basic principle

Your program (from the articy:draft perspective this is the external program) basically invokes an GUI-less articy:draft and operates just like a regular user. This means, your program needs to connect to an articy:server, login with an user account known to and licensed by the articy:server and opening a project that is allowed for the particular user account.
Just like when using the articy:draft GUI, projects can be opened in normal or in exclusive mode with all possibilities and restrictions coming with those modes.
Once the project has been opened successfully, objects can be searched using a simple search that resembles the search function of the articy:draft GUI. In addition to that, the internal query language allows formulating complex SQL-like queries.
Queried objects and their respective properties can be listed, changed or exported using the same exporter functionality that the articy:draft GUI provides. Before objects can be changed, claiming their partition is a pre-requisite.
The API even allows creating articy objects from scratch so your program can easily import tons of your own data like characters, locations or dialogs right into articy:draft ready to be used by the game designers.
Finally, opened projects can be saved, claimed partitions can be unclaimed/published and the project can be closed.
Every API method provides detailed error codes for failed actions and a logger callback allows your program to receive all information, that is also available to the user when running articy:draft in GUI mode.

2. How to use articy:draft API?

2.1. Preparations

2.1.1. Language of your program

Write your application or service in any .NET language (e.g. C#, VB.NET, Delphi.NET, managed C++ (for a complete list please check http://en.wikipedia.org/wiki/List_of_CLI_languages)

2.1.2. .NET Framework

Your application needs to use .NET Framework 4.0 or higher.
Be careful **NOT** to use the *.NET Framework 4.0 client profile*.

2.1.3. Main method

Your main method needs to be attributed with `STAThread`.

2.1.4. app.config

Since the components of articy:draft use a mix of .NET 2.0 and .NET 4.0 your program needs to apply an `app.config` containing:

```
<?xml version="1.0"?>
<configuration>
	<startup useLegacyV2RuntimeActivationPolicy="true">
		<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.0"/>
	</startup>
</configuration>
```

2.1.5. DLLs

Many DLLs and other files of articy:draft will be needed when running your own program. So you should set the output directory of your program's compiler to the **bin\x64** or **bin\x86** folders of the API installer's installation directory.
You may install the API files several times to different directories. The installer is more or less a simple automated copy machine. No registry keys are created in the process.

2.1.6. References

When starting a new project you must add references to the following assemblies

- PublicApiTypes
- ArticyApi
- QueryLayer

2.1.7. Logging

The API provides a logging callback for your program. Use this to learn, what the API methods are doing exactly. The logging callback follows the delegate:

```
public delegate void LoggingCallback( string aSource, EntryType aType, string aText);
```

2.1.7.1. Example

This is a very simple example for a method that receives logging messages from the API:

```
private static void CaptureLog(string aSource, EntryType aType, string aText)
{
	string prefix = string.Format("{1} ({0,-8})", aSource, aType.ToString()[0]);
	Console.WriteLine(String.Format("{0}: {1}", prefix, aText));
}
```

3. Example programs

3.1. List all approved entities

This sample program searches for all entities, that have a custom property “workflow_state” (drop-down list) set to “Approved” and lists them on the console:

```
public void ListAllApprovedObjects()
{
	// init API
	ArticyApi.Startup( Log );
	ApiSession session = ArticyApi.CreateSession();

	// connect & login
	session.ConnectToServer( "myserver", 16707 );
	session.Login( "jack.black", "jack's password" );

	// open project
	session.OpenProject( new OpenProjectArgs
	{
		ProjectGuid = new Guid( "291528ff-19ca-4d80-94ac-29ac2a03f854" ),
		ProjectFolder = @"C:\articy\myProject",
		ScmUsername = "jack's scm account name",
		ScmPassword = "jack's scm password",
		OpenExclusively = false,
		ForceProjectFolder = true,
		CleanCheckout = true
	} );

	// search objects
	var entityList = session.RunQuery(
		"SELECT * FROM Entities " +
		"WHERE workflow_info.workflow_state == Approved" );

	// list objects
	foreach( var entity in entityList.Rows )
	{
		Log( "myApp", EntryType.Info, entity.GetDisplayName() );
	}

	// tidy up
	session.CloseProject( new CloseProjectArgs() );
}

private void Log( string aMessageSource, EntryType aType, string aText )
{
	Console.Out.Write( aText );
}
```

3.2. Export some objects to Excel

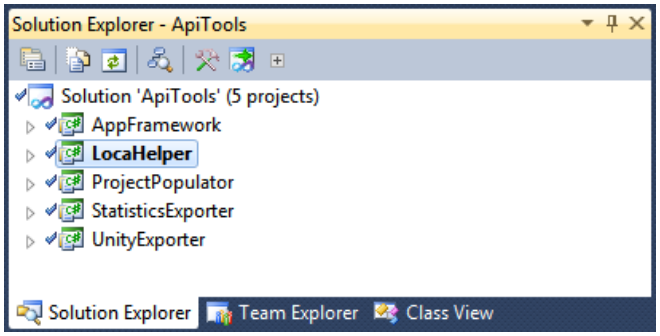
This program opens a project, searches for all objects whose Description property starts with "[ToDo]" and exports those objects to an Excel file.

```
public void ExportToXlsx()
{
	// init API
	ArticyApi.Startup( Log );
	ApiSession session = ArticyApi.CreateSession();

	// connect & login
	session.ConnectToServer( "localhost", 16707 );
	session.Login( "john.smith", "john's password" );

	// open project
	session.OpenProject( new OpenProjectArgs
	{
		ProjectGuid = new Guid( "291528ff-19ca-4d80-94ac-29ac2a03f854" ),
		ProjectFolder = @"C:\articy\myProject",
		OpenExclusively = false,
		ForceProjectFolder = false,
		CleanCheckout = false
	} );

	// search objects
	Guid projectRoot = session.GetProjectRoot().Id;
	var list = session.SimpleFindObjects( projectRoot, "[ToDo]",
		TextFilterRelation.StartsWith, SearchTextIn.Description, true );

	// export
	session.ExportData( new ExportToXlsxArgs
	{
		LogSourceName = "Export",
		Filename = @"C:\articy\export.xlsx",
		PreselectedObjects = list,
		CollectStatistics = true,
		FeatureOrder = ExportFeatureOrder.Alphabetical,
		GroupedBy = ExportGroupedBy.Template,
		PageSize = PageSize.A4
	} );

	// tidy up
	session.CloseProject( new CloseProjectArgs() } );
}
```

3.3. API-Tools

For more complex and realistic examples of what can be achieved via the articy:draft API please download the ApiTools.zip from the Nevigo download center. Use the same username and password that you have received after applying for the articy:draft trial.

3.3.1. Microsoft Visual Studio 10 solution

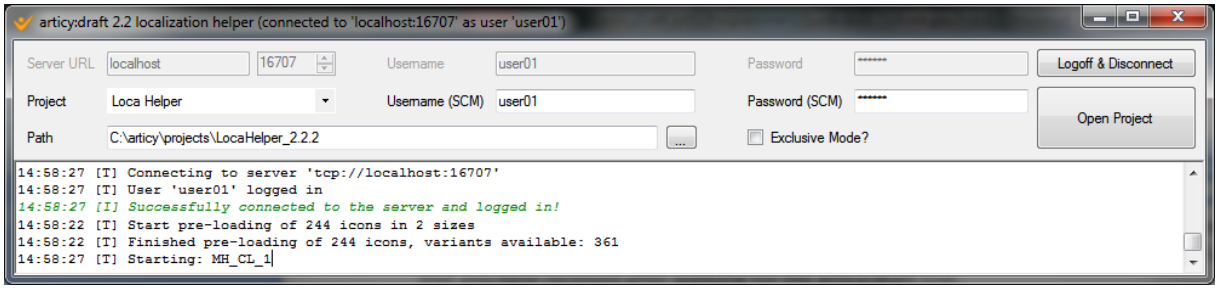


Those API-Tools are delivered together with source code and a Microsoft Visual Studio 10 solution.

This VS solution consists of an "AppFramework" project and four example projects "LocaHelper", "ProjectPopulator", "StatisticsExporter" and "UnityExporter".

3.3.2. AppFramework

The application framework contains a test bed GUI that already comes with all necessary controls to enter the required parameters for connecting to an articy:server, opening a project and receive logging output. You also have the option to customize the icon or a background image so that you can bring in your companies CI.



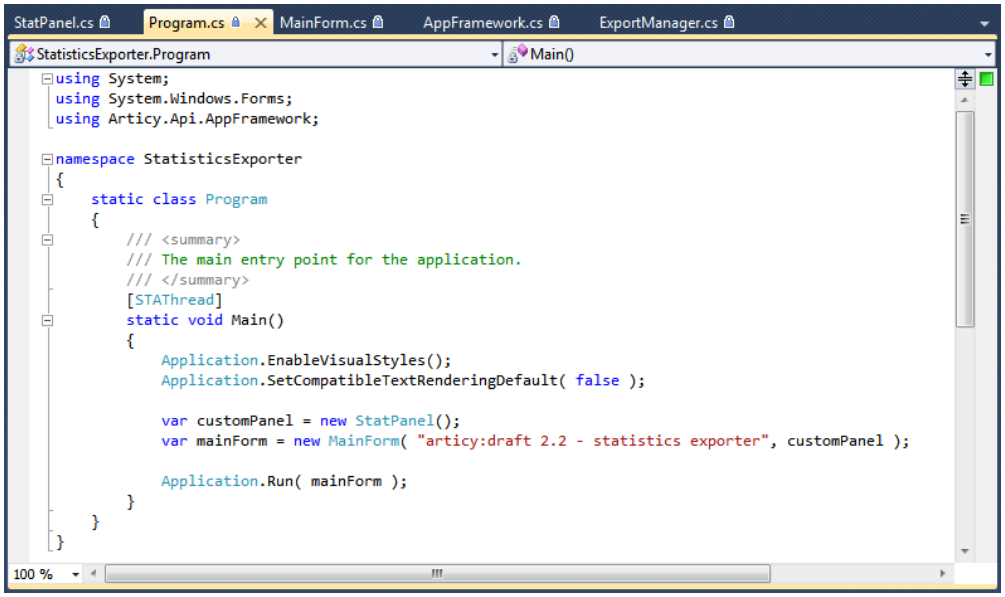
The AppFramework makes it easy to develop your own articy:draft API applications and test them conveniently before integrating it into the tool chain of your team's content pipeline.

3.3.3. Your own application

To create your own application, just create a new Visual Studio project as described under 2.1 Preparations or copy/clone one of the four already existing projects.

Basically the only two files you need to provide for your own application is the `Program.cs` containing the initialization of the AppFramework and an own panel class that is derived from `CustomPanel`. This custom panel contains your special GUI strip that replaces the strip with the connection parameters (see screenshot above) after you have successfully opened the articy:draft project.

Here is how the `Program.cs` should look like:



Place your business logic in the GUI callbacks that are linked to your custom panel's controls.

3.3.4. Need help?

If you should have questions about the API-Tools or the articy:draft API itself, just write an email to support@nevigo.com. The support team can help you even with the most technical questions.