



# ANARK SDK REFERENCE

## CONTENTS

Introduction.....	5
Exogeneous Data .....	5
Workflow Automation Architectures .....	6
Exemplary Workflow Architecture .....	6
Anark Plug-In API .....	7
Installing Your Plug-In.....	7
Anark Workstation .....	7
Anark Publish.....	7
Anark External XML Data.....	8
Install Stand-Alone XML Format.....	8
Importing XML Data In Anark Workstation .....	9
Install XML Data Connector .....	9
Importing XML Data In Anark Workstation .....	9
Importing XML using Anark Publish SOA .....	9
Using Imported Data in PDF and HTML Templates .....	10
XML Schema .....	10
Anark Data Connector Libraries and Example Code.....	12
Development Basics .....	12
Visual Studio .....	12

## Anark SDK Reference

Deployment .....	12
Datatype Support .....	12
Best Practices .....	12
Implementation .....	13
Requirements .....	13
Preferences.....	13
Providing the Attribute Metadata .....	13
Sample Code.....	16
Google Spreadsheet Example.....	16
Editable Attribute Example.....	17
Anark Publish REST Client Example Code .....	17
Anark Publish CLI Application .....	18
Installing the Application.....	18
Configuring ACSCLI .....	18
Encrypting User login and password in ACSCLI Configuration File .....	19
Using ACSCLI.EXE .....	20
Running Anark Recipes that have been deployed to Anark Publish.....	20
GetRecipeInfo .....	21
RunRecipe.....	22
Recipe Management.....	24
GetDocumentProperties .....	25
SetDocumentProperties .....	26
Kill a Run .....	26
Get the Status of a Run.....	26
Get Run MetaData.....	27
Get the Run Log Data.....	27
Watermark a PDF .....	27



## Anark SDK Reference

Remove Watermark from a PDF .....	29
Assemble a PDF .....	30
Disassemble a PDF .....	30
Get Comments from a PDF .....	31
Get Fields from a PDF .....	31
All Possible ACSCLI.EXE Return Codes .....	31
Anark Publish RESTful API .....	34
Using Anark RESTful API .....	34
Running Anark Recipes that have been deployed to Anark Publish .....	34
GET api/Recipes?recipeName={recipeName} .....	34
GET api/Recipes .....	35
GET api/Recipes/recipeName .....	35
PUT api/Recipes/recipeName .....	35
DELETE api/Recipes/recipeName .....	36
POST api/Recipes .....	36
POST Anark Collaborate Login .....	39
GET Anark Collaborate ContentProperties .....	39
POST api/DocumentProperties .....	40
PUT Properties .....	41
GET api/Logs/{id} .....	42
api/Runs .....	42
POST api/Watermark .....	43
DELETE api/Watermark .....	47
POST api/Attachments .....	47
DELETE api/Attachments .....	48
GET api/Attachments .....	48
GET api/Comments .....	49



Anark SDK Reference

GET api/Fields .....	49
UpdateRecipe.exe Command Line Application .....	51
Instructions for Deployment .....	51
Instructions for Use .....	51
Anark Publish Log Viewer .....	52
Instructions for Deployment .....	52
Instructions for Use .....	52



## INTRODUCTION

The Anark SDK allows IT professionals to integrate Anark Workstation and Anark Publish software products into a variety of automated workflows and extend the data model for Anark Workstation and Publish software to include exogenous data sources. ***Note that if your objective is to incorporate external data into Anark Collaborate content items where Anark Workstation is not required, there is a new way to do this that provides more flexibility. Please see the Anark Collaborate API Reference document for discussions regarding publishing data components to Anark Collaborate content items.***

## EXOGENEOUS DATA

Anark SDK allows IT and services organizations to automate the transformation of critical technical product data and files into role-and-use-case-specific content for downstream consumption:

- **Engineering**—engineering release content that includes eBOM data, 2D drawings, material specs, manufacturing notes, 3D CAD models (with or without MBD), and other critical product data;
- **Supply chain**—technical data package (TDP) content that aggregates various types of engineering data for use by OEM buyers for conducting RFx processes with qualified suppliers;
- **Manufacturing**—technical data package (TDP) content that aggregates various types of engineering data for consumption by manufacturing or quality teams, as well as manufacturing process-oriented content with information such as mBOM, bill of process, and bill of resource, for communicating manufacturing fabrication or assembly processes to the factory floor; and
- **Field service**—technical field service content that can include sBOM, maintenance documents, together with maintenance, repair, and troubleshooting process information.

With Anark software, a user can bring technical graphical information together with structured data. Excel spreadsheets, XML, and other data sources can be imported into Anark Workstation to author and publish role-and-use-case-specific content to be used throughout the extended enterprise.

[Anark External XML Data Connector and Schema](#) provides IT professionals an XML schema (.xsd) and associated XML data connector that imports and validates basic business objects from other systems such as PLM, ERP, etc., into Anark software. Data that is imported into Anark software becomes available for publishing alongside other files and data. All resources for the Anark External XML Data Connector and XML Schema can be found in the **Anark Core SDK.zip** file under the **\Anark Core External XML Data Connector and Schema** directory.

[Anark Data Connector Libraries and Example Code](#) allows developers to create .NET drop-in data connector plug-ins for Anark Workstation and Anark Publish that can import data from a variety of external data sources. For example, custom data connectors can be written to import data from enterprise SOAs, relational and flat-file databases, custom XML files, and more. Data that is imported into Anark software becomes available for publishing alongside other files and data. All resources for the Anark Data Connector Libraries and Example Code can be found in the **Anark Core SDK.zip** file under the **\Anark Core Data Connector Libraries and Example Code** directory.



## WORKFLOW AUTOMATION ARCHITECTURES

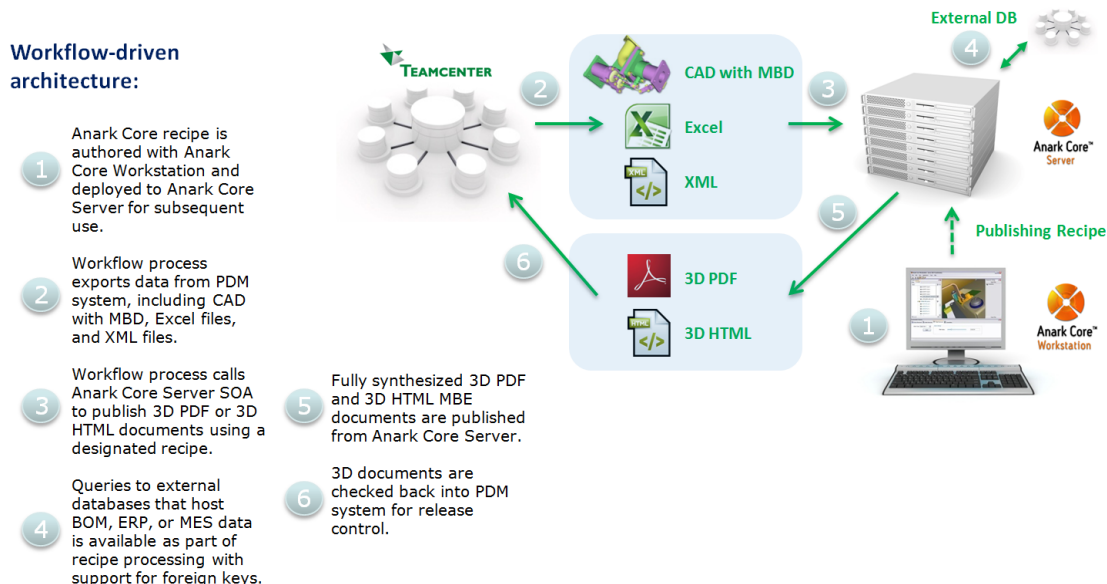
For certain use cases such as engineering technical data packages (TDP), the sheer volume of documents makes manual authoring and publishing workflows impractical, and technical data packages are nearly always managed as part of the engineering release process. With Anark SDK, cost-effective systems integrations between Anark Publish and existing PDM, MES, or ERP infrastructure can be achieved—to support enterprise-class workflow automation for publishing MBD-enabled design-and manufacturing-engineering documentation.

[Anark Publish REST Client Example Code](#) provides IT professionals with platform-independent control over Anark Publish in a variety of existing enterprise workflows using web services. Provides a fully-customizable means to control Anark Publish SOA automated publishing. Resources for the Anark Publish REST Client Example Code can be found in the **Anark Core SDK.zip** file under the **\Anark Publish REST Client Application\REST Client Example Code** directory.

[Anark Publish CLI Application](#) allows IT professionals to introduce the Anark Publish into a variety of existing enterprise workflows—with support for common scripting languages such as JavaScript, Perl, and Python, and compiled languages such as C, C++, Java, and Microsoft .NET languages such as C#, on the Windows platform. Provides an .EXE application with command-line parameters to control Anark Publish SOA automated publishing—without the need to write web service calls. Provides asynchronous control and feedback. Fully configurable via a simple XML file. All resources for the Anark Publish CLI Application can be found in the **Anark Core SDK.zip** file under the **\Anark Publish REST Client Application** directory.

## EXEMPLARY WORKFLOW ARCHITECTURE

Anark SDK supports publishing automation from any workflow process running on any platform that can make REST (HTTP/HTTPS) web service calls to Anark Publish. An exemplary PDM-centric workflow architecture for publishing 3D PDF is depicted below.



## ANARK PLUG-IN API

Plug-ins can be developed to execute custom code as part of Anark Recipe execution. These plug-ins make use of an SDK API that provides access to various information and pieces of content that have been loaded into the Anark object model. The API also allows certain types of entities, such as annotations, to be created. Certain types of modifications can also be performed on the content by using the API.

Note that the plug-ins referred to in this section are distinctly different than Attribute Data Connector plug-ins.

Plug-ins have a view of content in its final publishing state; in other words, they have access to the content after all other Anark Recipe transformations have taken place.

Sample code for a reference plug-in is provided that provides usage examples in order to make it easy to develop a new plug-in. Note that you should provide your own unique plug-in ID rather than use the ID provided in the reference plug-in (use a GUID generation tool to obtain your own ID). The SDK is contained in a single DLL that should be referenced (CoreSdkApi.dll) and no other Anark DLLs should be referenced by your plug-in. Plug-ins must be developed in Microsoft .NET.

Detailed **API documentation** is in this folder from the SDK zip file: **Anark Core SDK\API\Documentation**. Open the file **index.html** in a web browser.

## INSTALLING YOUR PLUG-IN

### ANARK WORKSTATION

Place your .NET plug-in DLL file into: C:\Program Files\Anark\Anark Workstation\PlugIns

- Create the PlugIns directory if necessary.
- After your plug-in is installed, you can include it in a Recipe by selecting File -> Add Plug-In(s) to Recipe. The plug-in will not execute unless it is part of a Recipe.

### ANARK PUBLISH

Place your .NET plug-in DLL file into: C:\Program Files\Anark\Anark Publish\CadTransformerWebInterface\bin\PlugIns

- Create the PlugIns directory if necessary.
- If the plug-in is missing, yet specified in a Recipe that is executed, an error will be logged and the content will not be published.



## ANARK EXTERNAL XML DATA

The Anark External XML Data Connector and Stand-Alone Format are available for use in Anark Workstation and Anark Publish to directly import XML files that adhere to the included Anark External XML Schema (.xsd).

The Anark External XML Schema supports basic business objects that can originate from any data system or format:

- Simple Types
- Urls/Hyperlinks
- Text Attributes
- Tables/Lists
- Trees/Hierarchies

Once data is written in the Anark External XML format, the Anark software products can import the XML files with the Anark External XML Data Connector or Stand-Alone Format. This provides a simple-yet-robust mechanism to import, synthesize, and publish exogenous data, including:

- Engineering data
- Provisioning data
- Process/manufacturing data
- Tooling data
- Supplier data
- And more

Once the XML is imported into Anark Workstation or Publish, the data is available for publishing into PDF and HTML 5 formats. PDF and HTML 5 templates can express different views of this data as required by the use case.

## INSTALL STAND-ALONE XML FORMAT

To use the Anark External Stand-Alone XML Importer with Anark Workstation, copy *XmlFormat.dll* to the Anark Workstation installation directory:

**[Anark Workstation Install Path] \Formats**

To use the Anark External Stand-Alone XML Importer with Anark Publish, copy *XmlFormat.dll* to the Anark Publish installation directory:

**[Anark Publish Install Path] \CadTransformerWebInterface\bin\Fformats**

**[Anark Publish Install Path] \CoreServiceWebInterface\bin\Fformats**

After installation, the software can import Anark External XML files from **Import File...**





## IMPORTING XML DATA IN ANARK WORKSTATION

Once the Anark XML Format is installed, to import an XML file:

- Click on **File > Import File...** The **Choose an Import File** dialog will appear.
- If you want to specify a file format to use during import, click on the menu labeled **All Importable Types** and select the desired file format from the drop-down menu. Choose a file to import and click **Open**.
- Click **OK** to finalize the import. The file will import into Anark Workstation and will be displayed in the **Structured Data Viewer**.
- After import, the Stand-Alone XML attributes will be added to the Workspace Pane in the Anark Workstation session. This Anark Workstation project can be saved, and the recipe can be deployed to Anark Publish for automation processing.

## INSTALL XML DATA CONNECTOR

To use the Anark External XML Data Connector with Anark Workstation, copy *AnarkCoreExternalXmlDataConnector.dll* to the Anark installation directory:

**[Anark Workstation Install Path] \AttributeDataConnectors**

To use the Anark External XML Data Connector with Anark Publish, copy *AnarkCoreExternalXmlDataConnector.dll* file to the Anark Publish installation directories:

**[Anark Publish Install Path] \CadTransformerWebInterface\bin\AttributeDataConnectors**

**[Anark Publish Install Path] \CoreServiceWebInterface\bin\AttributeDataConnectors**

After installation, the software can import Anark External XML files from the data connector.

## IMPORTING XML DATA IN ANARK WORKSTATION

Once the Anark External XML Data Connector is installed, to import an XML file:

- Click on **File > Import from Data Connector...** The **Import from Data Connector** dialog will appear.
- In the **Import from Data Connector** dialog, click on the pull-down menu to select the Anark External XML Data Connector, then click on the **Settings...** button. The **Data Connector Settings** dialog will appear.
- Click on the **"..."** button in the File Path field. Browse to the XML file and click **Open**.
- The Data Connector Settings dialog will appear with the attributes to be added to the Workspace and Components in the Anark Workstation session. Unless you wish to suppress specific attributes, click the **Ok** button. The XML data will be imported. This Anark Workstation project can be saved, and the recipe can be deployed to Anark Publish for automation processing.

## IMPORTING XML USING ANARK PUBLISH SOA



Once the Anark External XML Data Connector is installed, to import an XML file via the Anark Publish SOA:

- An Anark recipe that contains an action to import an Anark External XML file must be deployed to Anark Publish (see the Anark Workstation User Reference).
- In the web service call to publish a PDF or HTML 5 file, can run the recipe without changes or the call must specify the name of the Anark External XML file that is to be imported along with any other imported paginated documents, other structured data, and/or 3D CAD data. The web service will replace the name of the XML file that is specified in the deployed recipe if a change is specified.

Workspace data is exogenous data that is carried separately from the 3D Product structure and can be featured in PDF and HTML templates.

## USING IMPORTED DATA IN PDF AND HTML TEMPLATES

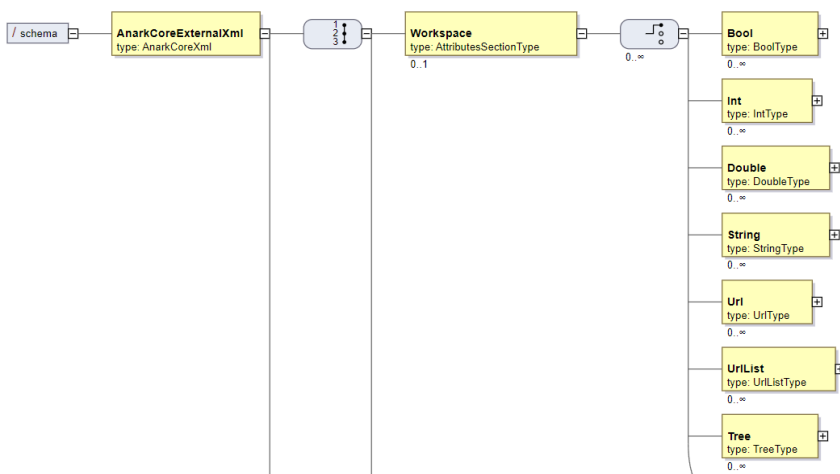
XML files that are imported into Anark software will appear as attributes on the Workspace (represented as the top node in the Anark Workstation Product Structure tree), or depending on the content of the XML file, extend a 3D component (part or assembly) attribute schema and bind with components that have a designated attribute that will serve as a foreign key. The PDF or HTML templates that will utilize the data originating from the imported XML file should expect unique data names and types for display within the PDF document or HTML content item.

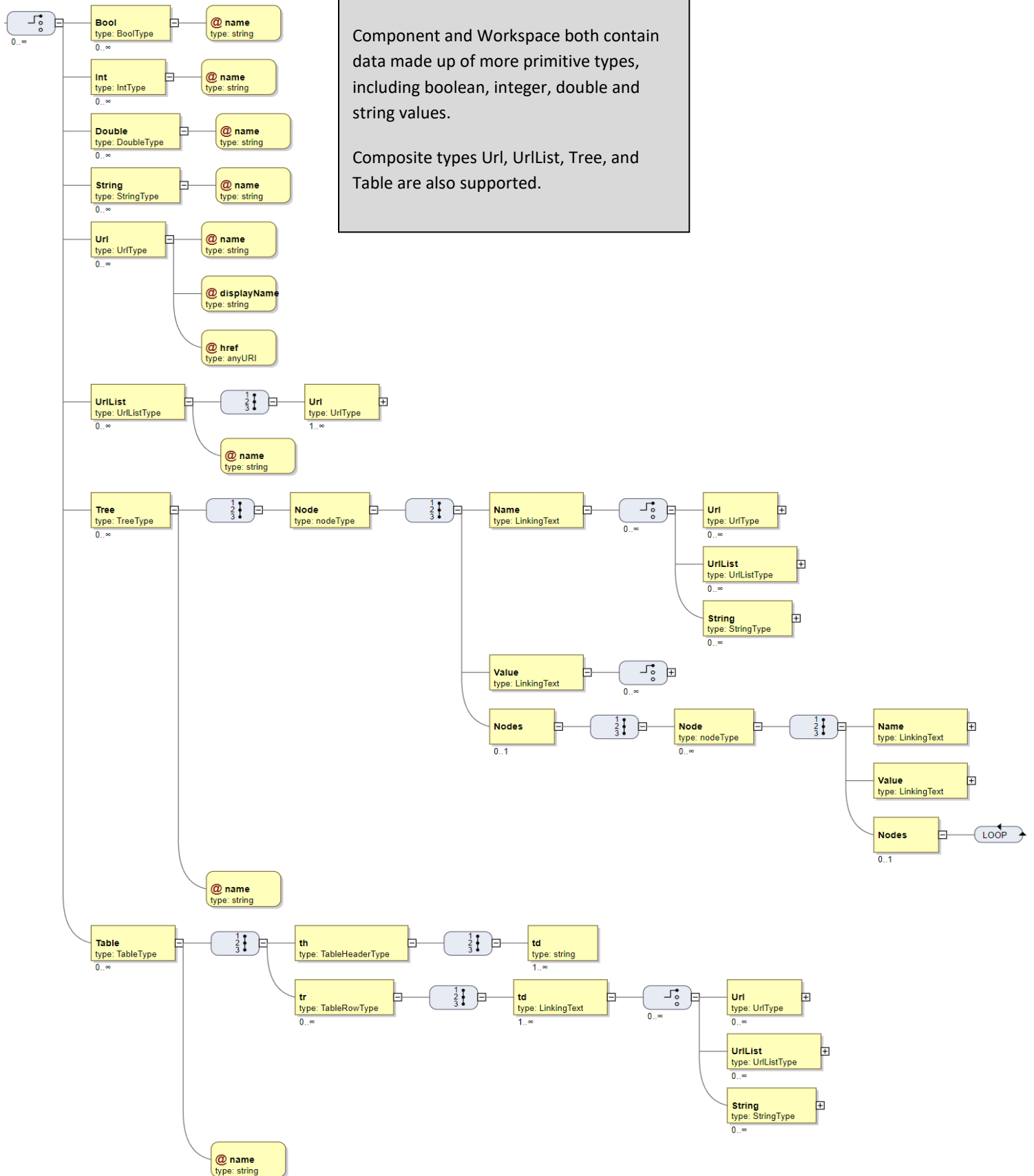
### XML SCHEMA

The **Anark Core External XML Schema.xsd** is .xsd file that describes the XML import validation performed when importing XML with the Anark External XML Data Connector. the guideline that should be used to form valid Anark External XML files. For XML files that deviate from this schema, import errors will occur when attempting to import the file into Anark software products.

Component data is exogenous data that will extend the attribute schema for parts and assemblies, as well as provide a binding mechanism (attributeKey) for associating the data with 3D components when they are imported.

XML files that are imported into Anark software will appear as attributes on the Workspace (represented as the top node in the Anark Workstation Product Structure tree), or depending on the contents of the XML file, extend a 3D Component (part and assembly) attribute schema and bind with components that have a designated attribute that match the designated **attributeKey**. Workspace data and Component data are both published to PDF documents and HTML 5 content items, in different locations within the JSON DOM that is created when the document is first displayed.





## ANARK DATA CONNECTOR LIBRARIES AND EXAMPLE CODE

Anark Data Connector Libraries and Example Code enables the development of custom data connectors for the Anark platform—allowing exogenous data to be imported and optionally associated with CAD geometry for publishing. The exogenous data may be based on in-memory data, a calculation, user input, or external systems such as databases or web services.

The job of the data connector is simply:

- To provide metadata about the attributes that are provided by the connector, such as the display name, datatype, cache expiration interval, etc.
- To provide the attribute data.

It is necessary to uniquely identify parts/assemblies especially when attribute data is extracted from external systems. Most often, this will be accomplished by using the part/assembly name. This task is delegated to the data connector, so the data connector can use any of the information contained within the part/assembly data to generate an identifier that will work with external systems. The information could even include other attributes. The basic prerequisite to pulling data in from an external system is that there must be some way to cross reference the CAD entities with the data in the external system.

## DEVELOPMENT BASICS

### VISUAL STUDIO

Your project type should be “Class Library” and your Visual Studio project target framework must be set to .NET 4.7 (or earlier).

### DEPLOYMENT

Copy your output and any third-party dependencies to **[Anark Workstation Install Path]** \AttributeDataConnectors. This can be performed manually or by an installation program.

For attributes to work consistently across multiple instances of Anark Workstation and Anark Publish, the same versions of the data connectors must be installed on each machine. If a data connector is missing, any attributes that are provided by that data connector will not be available.

### DATATYPE SUPPORT

The following C# datatypes are well supported by Anark: bool, byte, sbyte, char, decimal, double, float, int, uint, long, ulong, short, ushort, string, Color, DateTime, DataTable

### BEST PRACTICES

Keep in mind that Attribute Data Connectors must run in a workstation environment and in an automated server environment. This means that you should not ask for user input. Do not pop up dialogs, etc. User-specified settings such as file paths or usernames should be handled by the connector preferences.



## IMPLEMENTATION

### REQUIREMENTS

The data connector will take the form of a public class that is derived from **AttributeDataConnector**.

A **GUID** must be generated to identify your data connector and pass this to the **AttributeDataConnector** constructor in the form of an **AttributeDataConnectorId**.

### PREFERENCES

You can expose preferences to the user. Examples would be paths to CSV or Excel files, URIs to servers, login names and passwords, etc. Just derive a class from **CoreApplicationSettings** and use the standard **System.Configuration** markup. There is one special .NET attribute called **PreferenceKey** that is used to store the preference in Anark Recipes, and it should be unique to each preference. We recommend using a namespace convention such as **Company.Product.ConnectorName.Preference**.

If you have preferences to expose, override the **CreatePreferences** method; it must return a new instance of your preferences class. If you have no preferences to expose, do not override the **CreatePreferences** method.

### PROVIDING THE ATTRIBUTE METADATA

The constructor of your data connector should create **DynamicAttributeType** objects that contain the required attribute metadata. Each attribute must have a unique id (**DynamicAttributeId**), consisting of a name and a GUID. Supply parameters that best fit your use case, as follows.

#### TO CREATE A BASIC STRING OR NUMERIC ATTRIBUTE

```
public DynamicAttributeType(IAttributeDataConnector dataConnector,
    DynamicAttributeId attributeId, string displayName, Type dataType,
    AttributeOfflineRequirement offlineValueRequirement, long valueCacheInterval,
    object emptyValue, UiWidgetInfo uiWidgetInfo, string shortDisplayName, bool
    preserveOriginalValues, bool isSystemAttribute)
```

**dataConnector:** pass "this"

**attributeId:** create a unique id for the attribute

**displayName:** create a reasonably unique name for the attribute. This will be the attribute name used in export files and in the Anark user interface.

**dataType:** the .NET datatype of the attribute

**offlineValueRequirement:** **Unwanted** (Attribute values will never be serialized to the project) / **Required** (Attribute values will always be serialized to the project) / **Optional** (Attribute values will be serialized to the project depending on whether the user has enabled the "Save Attribute Values" preference.)

**valueCacheInterval:** 0



**emptyValue:** null

**uiWidgetInfo:** null

**shortDisplayName:** null

**preserveOriginalValues:** true

**isSystemAttribute:** false

---

## TO CREATE A BASIC STRING OR NUMERIC ATTRIBUTE WITH CACHING SUPPORT

Same as above, but **valueCacheInterval** specifies the amount of time in milliseconds to cache each attribute value.

---

## TO CREATE A MORE COMPLEX ATTRIBUTE

Same as above, with the following changes:

**emptyValue:** primarily used to provide the schema for empty DataTables

**icon:** not used now, you may pass null

**uiWidgetInfo:** information about the user interface widget that will be used to edit/view the attribute in Anark Workstation. There are 2 pre-defined widgets available: **CommonUiWidget.DataTableView** (a viewer for the .NET DataTable type) and **CommonUiWidget.DateEditor** (a basic date editor.). You can provide your own UI widget also.

---

## MAKING AN ATTRIBUTE AVAILABLE TO THE SYSTEM

For each of your **DynamicAttributeType** objects, you should call **AddSupportedAttribute** with specified **AttributeWiring**. This allows you to define getters, setters, and validation methods. You must specify a **GetValueDelegate** at minimum. To create an attribute that can be edited by the user of Anark Workstation, you must provide both a **SetValueDelegate** and a **RevertValueDelegate**. You may optionally provide a **ValidateValueDelegate** to validate the data entered by the user. The **ValidateValueDelegate** must return **DynamicValidationResult** where **errorText** is null if the attribute value validates correctly and non-null if there is a problem. This text will be shown to the user, so it should be descriptive of the problem so that the user can correct the value.

You must specify the applicable object types for each supported attribute. These are specified as a **CoreDynamicObjectTypeCollection**. Currently, the object types that you may choose from are: **StandardDynamicObjectTypes.WorkspaceType**, **StandardDynamicObjectTypes.InstanceType**, **StandardDynamicObjectTypes.ComponentType**, **StandardDynamicObjectTypes.OccurrenceType**, **StandardDynamicObjectTypes.GdtType**, **StandardDynamicObjectTypes.ViewType**, and **StandardDynamicObjectTypes.SupplementalGeometryType**.



---

## PROVIDING THE ATTRIBUTE DATA

Override the **InitializeSupportedAttributes** method to perform any heavy weight supported attribute initialization. If the supported attributes have not yet been added, they must be added when this method is called. You would override this if you are adding attributes that are not known at the time of development, such as from a user-specified spreadsheet or database table. To report an error to the user, you should specify the error in by setting **outputArgs.ErrorMessage** to something that the user will understand, such as "Your specified spreadsheet file was not found."

Override the **InitializeData** method if you need to perform any heavy weight initialization. You might override this to load a file or database table into memory. To report an error to the user, you should specify the error in by setting **outputArgs.ErrorMessage** to something that the user will understand, such as "Your specified spreadsheet file was not found."

Your getter method that you specified in the AttributeWiring should return the attribute value. This method will be called any time the attribute value is needed and is not in the cache. When it is called, it will be passed **nativeAndDynamicObjectContext** and **attributeld**.

The nativeAndDynamicObjectContext is used to determine the entity for which we need attribute data. This can be cast to ICadEntity, ICadComponentOccurrence, ICadChildOccurrence, ICadGdtAnnotation, ICadGeometry, INamedView, ICadPart, ICadAssembly, ICadInstance, etc. depending on the types that you support with your CoreDynamicObjectTypeCollection.

The attributeld is only needed if you wish to share getters and setters between multiple attributes.

---

## OTHER ERROR MESSAGES

Use the **Logger** property to report error messages other than what may occur in InitializeData or InitializeSupportedAttributes. When your data connector is running on Anark Workstation, these log messages will normally be directed to the Windows Event Log. When your data connector is running on Anark Publish, these log messages will normally be directed to the database.



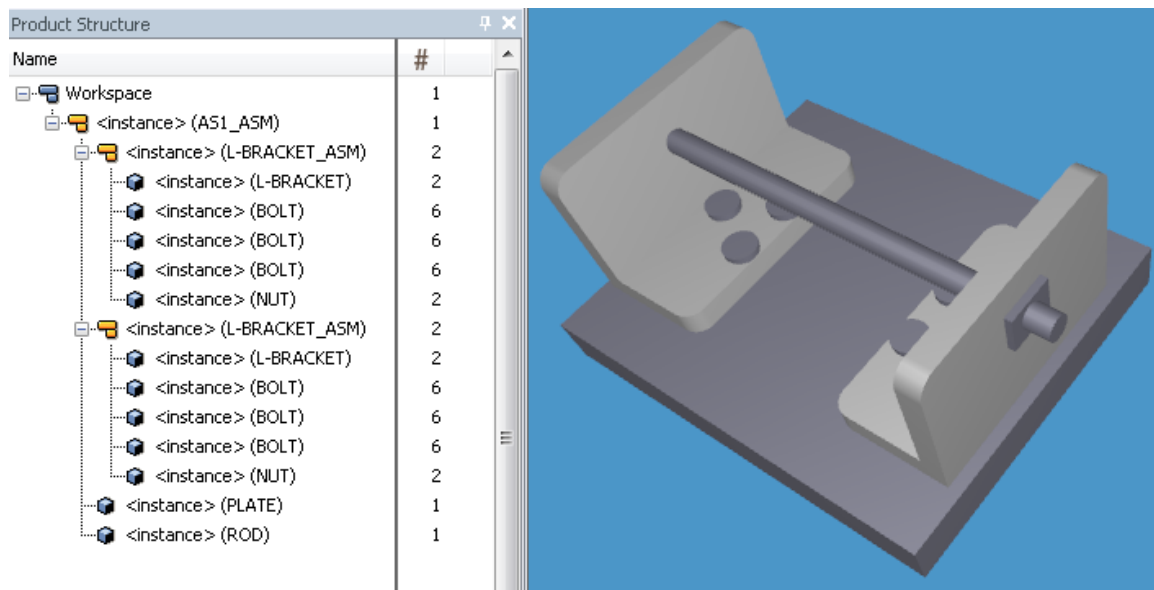
## SAMPLE CODE

## GOOGLE SPREADSHEET EXAMPLE

The provided Google Spreadsheet connector is an example of adding attribute data from external sources. It retrieves data from a Google spreadsheet and adds a “Component Description” attribute to parts and assemblies, using the published name as the relationship key.

	A	B	C	D
1	AS1_ASM	This is the entire assembly		
2	L-BRACKET_ASM	Subassembly		
3	L-BRACKET	L Bracket part		
4	BOLT	Just a normal bolt		
5	NUT	A square nut		
6	PLATE	A very boring plate		
7	ROD	A perfectly cylindrical rod		
8				
9				
10				
11				
12				
13				
14				

Here is the AS1\_PE.xt CAD model as imported in Anark Workstation. This CAD model is included with the Anark SDK to support this example.



It would be undesirable to make a network request every time an attribute is needed, so lazy loading is used to pull down the entire spreadsheet and then all of the values are cached. If the data is fairly static, you may prefer to load the data once in the InitializeData method with a cache interval on the DynamicAttributeType set to long.MaxValue or start an asynchronous request and block your “get”





method until the response arrives to provide better UI responsiveness. You may also store the in-memory data in your data connector rather than use the built-in cache mechanism.

The example code contains detailed comments; please refer to it for further information.

## EDITABLE ATTRIBUTE EXAMPLE

The editable attribute example code demonstrates how to provide editable attributes, with and without validation. The example code contains detailed comments; please refer to these for further information.

## ANARK PUBLISH REST CLIENT EXAMPLE CODE

Anark Publish REST Client Example Code provides IT professionals with platform-independent control over Anark Publish in a variety of existing enterprise workflows using web services. Provides a fully-customizable means to control Anark Publish SOA automated publishing in push architectures.

The Anark Publish SOA provides an automation API with REST (HTTP/HTTPS) web services that can be invoked by any software system—written in virtually any computer language, running on any OS platform. REST interfaces and client example code can be found in the **Anark Core SDK.zip** file under the **\Anark Publish REST Client Application\REST Client Example Code** directory.

The example code shows how to invoke the Anark Publish REST API for all supported API functionality. It depends on the NewtonSoft.JSON library, so be sure that your project includes a reference to that library (it is included in the SDK package) when using this example code. Please see the Execute method in Program.cs as a starting point.



## ANARK PUBLISH CLI APPLICATION

Anark Publish CLI Application allows IT professionals to introduce the Anark Publish into a variety of existing enterprise workflows—with support for common scripting languages such as JavaScript, Perl, and Python, and compiled languages such as C, C++, Java, and Microsoft .NET languages such as C#, on the Windows platform.

The application is a 64-bit Windows .EXE with command-line parameters to control Anark Publish SOA automated publishing—without the need to write web service calls. It provides asynchronous control and feedback and is fully configurable via a simple XML file.

## INSTALLING THE APPLICATION

All resources for the Anark Publish CLI Application can be found in the **Anark Core SDK.zip** file under the **\Anark Publish Client Connector Application** directory. In this directory, you will find the following files:

***ACSCLI.exe***—the command-line-interface (CLI) application invoked in your workflow

***ACSCLI.exe.config***—the ACSCLI application configuration file

Copy these two files together along with all the other DLL (.dll) files to a file folder that is conveniently located with respect to your workflow process application. Before using the ACSCLI, you must first edit the XML configuration file.

## CONFIGURING ACSCLI

The ACSCLI.exe.config file provides information regarding the server name that will be used to call Anark Publish SOA web services, and the protocol that should be used when communicating with Anark Publish (HTTP or HTTPS). Modify the “Server Name” value highlighted below in yellow to match the server name where you are hosting Anark Publish. Modify the “Protocol” value highlighted below in yellow to select your intended protocol. There is also an option to use client certificates and in order to use client certificates set the “UseX509ClientCertificate” value to “true” and specify the certificate issuer name by modifying the “X509Issuename” value.

If SQL Windows authentication is used when installing the Anark Publish, IIS Basic Authentication is enabled and communication with Anark Publish web services will be over HTTPS. Modify the “UserLogin” and “Password” values highlighted below in yellow to specify the Windows login info. “Protocol” value should be set to “https”.

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <configSections>
    <sectionGroup name="applicationSettings"
type="System.Configuration.ApplicationSettingsGroup, System, Version=2.0.0.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089" >
      <section name="Anark.Core.Client.CoreSOAClient.Settings1"
type="System.Configuration.ClientSettingsSection, System,
Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" requirePermission="false"/>
    </sectionGroup>
  </configSections>
  <applicationSettings>
```



```

    <Anark.Core.Client.CoreSOAClient.Settings1>
      <setting name="ServerName" serializeAs="String">
        <value>anark-pc-bravo</value>
      </setting>
      <setting name="Protocol" serializeAs="String">
        <value>http</value>
      </setting>
      <setting name="X509IssuerName" serializeAs="String">
        <value/>
      </setting>
      <setting name="UseX509ClientCertificate" serializeAs="String">
        <value>False</value>
      </Anark.Core.Client.CoreSOAClient.Settings1>
    <Anark.Core.Client.CoreSOAClient.AuthenticationSettings>
      <setting name="UserLogin" serializeAs="String">
        <value>DOMAIN\username</value>
      </setting>
      <setting name="Password" serializeAs="String">
        <value>password</value>
      </setting>
    </Anark.Core.Client.CoreSOAClient.AuthenticationSettings>
  </applicationSettings>
</configuration>

```

## ENCRYPTING USER LOGIN AND PASSWORD IN ACSCLI CONFIGURATION FILE

A batch file called ACSCLI\_EncryptUtil.bat has been provided to perform the encryption tasks. Before running it, add the below section in the ACSCLI.exe.config file above the “applicationSettings” XML element (<applicationSettings>) and below the “configSections” XML element (</configSections>):

```

<configProtectedData>
  <providers>
    <add keyContainerName="ACSCLIEncKeys"
      useMachineContainer="true"
      description="Uses RsaCryptoServiceProvider to encrypt and decrypt"
      name="AppEncryptionProvider"
      type="System.Configuration.RsaProtectedConfigurationProvider,
      System.Configuration,
      Version=2.0.0.0,
      Culture=neutral,
      PublicKeyToken=b03f5f7f11d50a3a" />
  </providers>
</configProtectedData>

```

To run ACSCLI\_EncryptUtil.bat, open a command prompt as an administrator (right click command prompt and select “Run as Administrator”).

To encrypt the username and password in the ACSCLI.exe.config file, specify /encrypt and the user who will be running ACSCLI.exe:

```
ACSCLI_EncryptUtil.bat /encrypt DOMAIN\USER
```

To run ACSCLI.exe on a different machine with the encrypted config file, first export the encryption key on the machine where encryption was done by specifying /exportkey:

```
ACSCLI_EncryptUtil.bat /exportkey
```



This will create an XML file (ACSLI\_SecurityKeys.xml). Copy the XML file and “ACSLI\_EncryptUtil.bat” file to the other machine where ACSCLI.exe will be run, and it should be in the same folder. To import the encryption key, specify /importkey and the user who will be running ACSCLI.exe:

```
ACSLI_EncryptUtil.bat /importkey DOMAIN\USER
```

After the key is successfully imported on the machine, ACSCLI.exe will be able to decrypt the username and password correctly. After importing the key, delete the XML file for security reasons.

## USING ACSCLI.EXE

### RUNNING ANARK RECIPES THAT HAVE BEEN DEPLOYED TO ANARK PUBLISH

The ACSCLI.exe utility allows you to send remote messages to Anark Publish to run recipes. This functionality can be used to publish HTML content and PDF documents, including multiple published items from the same recipe (and same import). Recipe operations are queued on the server, so by default the calls to ACSCLI are non-blocking – they will often use a run id (a GUID) that can be used to check on the status of the submitted recipe run. The run id will be created for you if not specified, but usually you will want to create that id so that you can easily check the status of the run later. Run metadata can be specified as well as the priority if Anark Publish is configured to process runs with different priorities.

Some parameters are required, and parameters in square brackets [] are optional. The waituntildone option can be used to make the ACSCLI call into a blocking call that will check the status of the run continuously until the run is finished or the specified timeout is exceeded. When specifying waituntildone, a maximum run time must be specified, and the run will be killed if the wait time exceeds the timeout, and the current status of the run will be output to the console (standard output). If the final run status is not 'RunCompleted' then the log will be automatically retrieved and shown. The total run time in seconds will be shown at the end. The quiet option will suppress any output other than errors.

The following options can be added to any ACSCLI command:

-servername	Overrides the servername and protocol setting specified in the ACSCLI.exe.config file.  Example: -servername <a href="https://myserver">https://myserver</a>
-logfile <logFilePath>	Logs the output to a file.  Example: -logfile <a href="\\server\share\log.txt">\\server\share\log.txt</a>

The recommended way to run recipes is with the **runrecipe** parameter. The setup for this is as follows:

- 1) Create recipe using Anark Workstation
- 2) Deploy the recipe to Anark Publish
- 3) Make sure ACSCLI.exe.config has been configured properly



- 4) Invoke ACSCLI with the **getrecipeinfo** parameter and the name of the recipe to retrieve the options that can be overridden in your recipe
- 5) Using the information from step 4, invoke ACSCLI with the **runrecipe** parameter and specify the options, if any, that you wish to replace.

## GETRECIPEINFO

### Syntax:

```
ACSCLI -getrecipeinfo -recipeName <recipeName>
```

### Example:

```
> .\ACSCLI.exe -servername mattserver2019 -getrecipeinfo -recipeName myrecipe
```

Recipe 'myrecipe' contains the following settings that can be overridden:

```
    actionId='502' actionType='Import' formatId='9debd1f-67d2-4a0d-bfea-274af9b7115e
(XtFileFormat)' optionKey='File' optionValue='\\server\share\file.x_t'

    actionId='504' actionType='Import' formatId='2305092f-48f2-444d-9695-708a7d992d3a
(ExcelFormat)' optionKey='File' optionValue='\\server\share\fileA.xlsx'

    actionId='506' actionType='Import' formatId='2305092f-48f2-444d-9695-708a7d992d3a
(ExcelFormat)' sequenceNumber='2' optionKey='File' optionValue='\\server\share\fileB.xlsx'

    actionId='507' actionType='Export' formatId='1afad9fa-bb17-491f-9d49-2e14ea7b07f3
(MBEWebFormat)' optionKey='File' optionValue='MBEWeb'

    actionId='507' actionType='Export' formatId='1afad9fa-bb17-491f-9d49-2e14ea7b07f3
(MBEWebFormat)' optionKey='TemplateId' optionValue='5d7975858f33c86ff6dbdfa3'

...
```

**actionId** – Is the unique identifier of the action in the recipe. Each action in the recipe has a unique ID that is created in the order in which actions were applied during the creation of the recipe. Different recipes may have different action IDs, so avoid referencing action IDs unless you need a very strong binding to a specific recipe.

**actionType** – This is the action type, for example “Import” would be an import action and “Export” would be a publishing action.

**formatId** – This provides an identifier that specifies the type of import or export action. For example, you could reference this to update the path of a CSV file while avoiding updates of Excel files.

**sequenceNumber** – This is provided when the recipe contains more than one instance of the same action. If there are two Excel import actions, then the sequenceNumber of the first one would be 1 and the sequenceNumber of the second one would be 2. The sequenceNumber value is only shown when the value is greater than one but it is still present and can be referenced. For example, you could use this to update the first and/or second Excel import file path.

**optionKey** – This is the name of an option that can be specified by an action.

**optionValue** – is the value of the option.



## RUNRECIPE

**runRecipe** requires the name of the recipe (the name is assigned when deploying with Anark Workstation). Optionally, you can provide replacement values for certain settings in the recipe. The run will be executed on a modified copy of the recipe and the original recipe will not be modified.

### Syntax:

```
ACSLI -runrecipe [runId] -recipeName <recipeName>
      [-runmetadata <runMetaData>]
      [-runpriority <high | normal | low>]
      [-recipeupdate "[actionId='<actionId1>' [actionType='<actionType1>']
                      [formatId='<formatId1>' [sequenceNumber='<sequenceNumber1>']
                      [optionKey='<optionKey1>' [optionValue='<optionValue1>']
                      replacementOptionValue='<replacementOptionValue1>']"
      [-recipeupdate "[actionId='<actionId2>' [actionType='<actionType2>']
                      [formatId='<formatId2>' [sequenceNumber='<sequenceNumber2>']
                      [optionKey='<optionKey2>' [optionValue='<optionValue2>']
                      replacementOptionValue='<replacementOptionValue2>']"
      ...
      [-waituntildone <maxruntimeinminutes>]
      [-quiet]
      [-contentid <contentId>]
      [-componentid <componentId>]
```

To provide replacement values, specify the **recipeupdate** parameter which provides a way to find and replace settings in the recipe. This parameter can be provided multiple times to replace multiple settings. You must specify at least one of **actionId**, **actionType**, **formatId**, **sequenceNumber**, **optionKey**, **optionValue** (hint: use the information provided by **getrecipeinfo**), and you must specify **replacementOptionValue**. Note that the search is case-sensitive. To specify an array, or list, of values, separate each value with a pipe character (`|`), for example: `replacementOptionValue='file1|file2|file3'`.

To override a specific component on an Anark Collaborate content item, the **contentid** and **componentid** parameters can be specified to identify that component. The recipe being run must already contain an Anark Collaborate publishing action in order to use the functionality, in which the remaining parameters can be specified and overridden. The **contentid** parameter will always override the content id present in the existing publish action, as well as overriding any content id specified through **recipeupdate**.

### ANARK COLLABORATE PUBLISHING TIPS:

When an Anark Collaborate publishing action is added to a recipe (by Anark Workstation), the unique content item id is recorded in the recipe, so the content item can be republished (overwritten). The content item will be overwritten by default when the export action is refreshed or when the whole recipe is executed. The option *ContentId* contains this content item id. To publish a new content item without overwriting the original content item, specify an empty **replacementOptionValue** for this option, as shown in example #3 below. To retrieve the published content item id, wait until the run is complete, then retrieve the run log (use the **-log** parameter), and look for this data:

```
publishedContentInfo: { actionId: 123, target: 'MBEWeb', contentId:
  '5a0b27539056d257465d2735' }
```



#### EXCEL / CSV IMPORT TIPS:

When using the Excel or CSV attribute data connector to produce workspace-level tables, the attribute name is based on the file name (excluding the path of the file). If your PDF or Anark Collaborate template references this table attribute specifically by name, it's important that you preserve the Excel or CSV file name when specifying a recipe update, for example: `\\this\can\change\andthiscannot.xlsx`. It is also possible to create templates that are tolerant of a change in attribute name but it's easier to keep the same file name.

#### PDF PUBLISHING TIPS:

When specifying a recipe update for the option *Anark.Core.CadAdapter.Export.Pdf.EmbeddedFiles*, you must specify where to put the PDF template / 3D content using the text *PDF Template Placeholder*. The content will be embedded in the same order as specified. For example, this will add the `file_to_embed.pdf` content before the templated content:

```
replacementOptionValue='\\server\share\file_to_embed.pdf|PDF Template Placeholder'
```

and this will add content before and after the templated content:

```
replacementOptionValue='\\server\share\file_to_embed.pdf|PDF Template Placeholder|\\server\share\another_file_to_embed.pdf'
```

#### EXAMPLE #1, RUN A RECIPE WITH NO CHANGES:

```
> ACSCLI.exe -runrecipe -recipeName myrecipe -waituntildone 10
```

#### EXAMPLE #2, RUN A RECIPE WITH MODIFICATIONS TO SOME SETTINGS:

```
> ACSCLI.exe -runrecipe -recipeName myrecipe -waituntildone 10 -recipeupdate
"optionValue='\\server\share\originalImport.prt'
replacementOptionValue='\\server\share\newImport.prt'" -recipeupdate
"actionId='505' optionKey='Filename'
replacementOptionValue='\\server\share\newOutput.pdf'"
```

This shows that recipe settings can be replaced using different search criteria. The value `\\server\share\originalImport.prt` will be replaced with the value `\\server\share\newImport.prt`. Additionally, the value of the Filename option on action 505 will be replaced with `\\server\share\newOutput.pdf`.

#### EXAMPLE #3, PUBLISH A NEW CONTENT ITEM TO ANARK COLLABORATE WITHOUT OVERWRITING ORIGINAL PUBLISHED CONTENT ITEM:

```
> ACSCLI.exe -runrecipe -recipeName myrecipe -waituntildone 10 -recipeupdate
"optionKey='ContentId' replacementOptionValue=''" -recipeupdate
"optionKey='ContentName' replacementOptionValue='my content name'"
```

#### EXAMPLE #4, RUN A RECIPE THAT HAS 2 FILES ATTACHED TO A PDF, AND SPECIFY A DIFFERENT ATTACHED FILE FOR ONE OF THE ATTACHMENTS:



```
> ACSCLI.exe -runrecipe -recipeName myrecipe -waituntildone 10 -recipeupdate  
"optionKey='Filename' optionValue='\\server\share\myattachment.xlsx'  
replacementOptionValue='\\server\share\different_file.pdf'"
```

EXAMPLE #5, RUN A RECIPE AND ADD ATTACHMENTS TO THE PDF:

```
> ACSCLI.exe -runrecipe -recipeName myrecipe -waituntildone 10 -recipeupdate  
"optionKey='Anark.Core.CadAdapter.Export.Pdf.Attachments'  
replacementOptionValue='\\server\share\attachment1.pdf|\\server\share\attachment  
2.pdf'"
```

EXAMPLE #6, RUN A RECIPE AND EMBED A PDF FILE IN THE PUBLISHED PDF:

```
> ACSCLI.exe -runrecipe -recipeName myrecipe -waituntildone 10 -recipeupdate  
"optionKey='Anark.Core.CadAdapter.Export.Pdf.EmbeddedFiles'  
replacementOptionValue='\\server\share\file_to_embed.pdf|PDF Template  
Placeholder'"
```

EXAMPLE #7, RUN A RECIPE AND SPECIFY CONTENT METADATA:

```
> ACSCLI.exe -runrecipe -recipeName myrecipe -waituntildone 10 -recipeupdate  
"actionId='509' optionKey='ContentProperties'  
replacementOptionValue='name1,value1|name2,value2|name3,value3'"
```

EXAMPLE #8, RUN A RECIPE AND REPLACE A SPECIFIC COMPONENT

```
> ACSCLI.exe -runrecipe -recipeName myrecipe -waituntildone 10 -contentid  
SOMEEXISTINGCONTENTID -componentid SOMEEXISTINGCOMPONENTID
```

## RECIPE MANAGEMENT

ACSCLI.exe can be used to deploy, retrieve, list, and delete recipes from Anark Publish.

### DEPLOY RECIPE TO ANARK PUBLISH

#### Syntax:

```
ACSCLI -deployrecipe -recipeName <recipeName> -caxfile <caxFile>
```

#### Parameters:

- recipeName is the name that will be used to reference the recipe on Anark Publish
- caxfile is the recipe file (ending with a .cax file extension). This is one of the files that is created when saving a project with Anark Workstation.

#### Example:

```
> ACSCLI.exe -deployrecipe -recipeName "my recipe" -caxfile "c:\path\recipe.cax"
```

### LIST THE RECIPES DEPLOYED TO ANARK PUBLISH





**Syntax:**

```
ACSLI -listrecipes
```

**Example:**

```
> ACSLI.exe -listrecipes
```

## FETCH RECIPE FROM ANARK PUBLISH

**Syntax:**

```
ACSLI -fetchrecipe -recipeName <recipeName> -caxfile <caxFile>
```

Parameters:

- recipeName is the name of the recipe
- caxfile is the recipe file to create or overwrite (ending with a .cax file extension).

**Example:**

```
> ACSLI.exe -fetchrecipe -recipeName "my recipe" -caxfile "c:\path\recipe.cax"
```

## DELETE RECIPE FROM ANARK PUBLISH

This will permanently delete the recipe from the Anark Publish database.

**Syntax:**

```
ACSLI -deleterecipe -recipeName <recipeName>
```

Parameters:

- recipeName is the name of the recipe

**Example:**

```
> ACSLI.exe -deleterecipe -recipeName "my recipe"
```

## GETDOCUMENTPROPERTIES

**getDocumentProperties** requires the full path of a PDF file, and the full path of a CSV file that will be created. The document properties will be read from the PDF and exported to the CSV file.

**Syntax:**



```

ACSCLI -getdocumentproperties [runId] -document <pdf file name>
      -outcsv <csvFile>
      [-runmetadata <runMetaData>]
      [-runpriority <high | normal | low>]
      [-waituntildone <maxruntimeinminutes>]
      [-quiet]

```

**Example:**

```

> ACSCLI.exe -getdocumentproperties -document "\\server\share\file.pdf" -output
  "\\server\share\out.csv" -waituntildone 10

```

## SETDOCUMENTPROPERTIES

**setDocumentProperties** requires the full path of a PDF file, and the property names and values to set. The document properties will be written to the PDF file.

**setDocumentProperties** creates or overwrites preexisting PDF properties. Previously created document properties can be changed by referencing an existing PDF document property name and changing its value. Listing new document properties will not remove the previously created document properties. All previously created document properties can only be removed manually inside of the PDF.

You cannot edit Adobe standard properties "CreationDate," "ModDate," "Producer," and "Trapped." If these standard properties are added via Anark, they will be added to custom properties with a "--Text" suffix.

**Syntax:**

```

ACSCLI -setdocumentproperties [runId] -document <pdf file name>
      [-property "name='<name1>' value='<value1>'" ]
      [-property "name='<name2>' value='<value2>'" ]
      ...
      [-runmetadata <runMetaData>]
      [-runpriority <high | normal | low>]
      [-waituntildone <maxruntimeinminutes>]
      [-quiet]

```

**Example:**

```

> ACSCLI.exe -setdocumentproperties -document "\\server\share\file.pdf"
  -property "name='Release Status' value='Released'" -waituntildone 10

```

## KILL A RUN

**Syntax:**

```

ACSCLI -kill <runId>

```

**Example:**

```

> ACSCLI -kill 9AA3AE42-95D0-4a69-B1D1-FFD227BC37DE -logfile \\share\out\Run.log

```

## GET THE STATUS OF A RUN

**Syntax:**

```
ACSLI -status <runId>
```

**Example:**

```
> ACSCLI.exe -status 9AA3AE42-95D0-4a69-B1D1-FFD227BC37DE -logfile  
"\\share\out\status.log"
```

## GET RUN METADATA

This can be used to retrieve information about a run, such as associated metadata, performance information, etc.

**Syntax:**

```
ACSLI -getrun <runId>
```

**Example:**

```
> ACSCLI.exe -getrun 9AA3AE42-95D0-4a69-B1D1-FFD227BC37DE
```

## GET THE RUN LOG DATA

**Syntax:**

```
ACSLI -log <runId>
```

**Example:**

```
> ACSCLI -log 9AA3AE42-95D0-4a69-B1D1-FFD227BC37DE -logFile  
"\\share\out\error.log"
```

The -logFile parameter is optional – when performing troubleshooting on the command line, this can be omitted to show the log output directly in the command window.

## WATERMARK A PDF

The ACSCLI.exe utility allows you to add a watermark to a PDF using Anark Publish. A run id (GUID) needs to be specified as well as the location of the PDF file and the watermark text. Other parameters are optional.

**Syntax:**

```
ACSLI -watermarkpdf [runId] -cadfile <inputPDFFileLocation> -txt <watermarkText>  
[-font <textFontName default:Arial>]  
[-fontsize <fontSize default:72>]  
[-txtalign <textAlignment default:center (left|center|right)>]  
[-txtcolor <textColor default:0,0,0 (values 0 to 255 in order of  
red,green,blue)>]  
[-rot <textRotationInDegrees default:0.0>]  
[-hdist <horizontalDistance default:0.0>]  
[-halign <horizontalAlignment default:center (left|center|right)>]  
[-vdist <verticalDistance default:0.0>]
```



```

[-valign <verticalAlignment default:center (top|center|bottom)>]
[-startpage <startPageIndex default:0>]
[-endpage <endPageIndex default:maxintvalue>]
[-pagesubset <pageSubset default:all (odd|even|all)>]
[-scale <scaleRelativeToTargetPage default:1.0 (0.0 to 1.0 (100%))>]
[-opacity <opacity default:1.0 (0.0 to 1.0 (100%))>]
[-percentagevals <usePercentageForUnits default:true>]
[-showonscreen <showonScreenDisplay default:true>]
[-showonprint <showWhenPrinting default:true>]
[-fixedprint <constantPositionAndSize default:false>]
[-zOrderTop <putWatermarkInForeground default: true>]
[-runmetadata <runMetaData>]
[-runpriority <high | normal | low>]
[-waituntildone <maxruntimeinminutes>]
[-quiet]

```

runId	Run identifier GUID/UUID (optional)
inputPDFFileLocation	Location of the PDF file
watermarkText	Watermark text
textFontName	Watermark text font (Default is Arial)
fontSize	Watermark text font size in point (Default is 72)
textAlignment	Watermark text alignment - specify either left, center, or right (Default is center)
textColor	Watermark text color - R, G, B values separated by a comma, range is from 0 to 255 (Default is black (0,0,0))
textRotationInDegrees	Watermark counter-clockwise rotation in degrees (Default is 0.0)
horizontalDistance	Watermark position - horizontal offset (Default is 0.0)
horizontalAlignment	Watermark position - horizontal alignment, specify either left, center, or right (Default is center)
verticalDistance	Watermark position - vertical offset (Default is 0.0)
verticalAlignment	Watermark position - vertical alignment, specify either top, center, or bottom (Default is center)
startPageIndex	Starting page index to which watermark should be applied (Default is 0)
endPageIndex	Ending page index to which watermark should be applied (Default is max integer value)
pageSubset	Subset of the page range to which watermark should be applied – specify either all, even or odd (Default is all)
scaleRelativeToTargetPage	Scale factor to be used when adding the watermark, with 1.0 meaning 100% (Default is 1.0)
opacity	Opacity to be used when adding the watermark, with 0.0 meaning fully transparent and 1.0 meaning fully opaque (Default is 1.0)
usePercentageForUnits	If this is true, horizontal and vertical offset position represent percentages of the page dimensions. Otherwise in user units (Default is true)
showonScreenDisplay	Specifies whether the watermark should be visible on screen (Default is true)
showWhenPrinting	Specifies whether the watermark should be printed (Default is true)
constantPositionAndSize	Specifies whether the watermark maintain their size and position regardless of the dimensions of the target media. This option should be set to true for putting watermark over 3D (Default is false). See additional fixedprint notes below.
putWatermarkInForeground	True to put the watermark in the foreground, false to put the watermark in the background.



runMetaData	Use this to specify any user metadata for this run
runpriority	Priority of the run – Normal, Low or High. Server should be configured to process runs with different priorities. (Default is Normal).
maxruntimeinminutes	Maximum runtime before cancelling the run.

Using parameter -fixedPrint true:

The Watermark will appear in front of 3D space but any color in the pdf that is not "burnt in" by the JavaScript will show above the watermark. This includes background color of form fields and text inside of a form field.

Using parameter -fixedPrint false:

The watermark will appear in front of all form fields but will not appear in font of 3D space.

The recommended practice when using the parameter -watermarkPdf is to make two separate watermark calls. If both calls are being run from the same batch file, a TIMEOUT or SLEEP command will need to be used or both calls will be run in parallel; the first call will hold the PDF and the second call will fail because it can't find the PDF.

#### Example:

```
ACSLCLI -watermarkpdf 5c4a72d0-081d-457f-96cc-0cca6e9ffc21 -cadfile
\\share\in\tdp.pdf -txt Released -font Arial -fontsize 144 -txtalign center
-txtcolor 255,0,0 -rot 90 -hdist 0.2 -halign right -vdist 0.3 -valign top
-startpage 0 -endpage 5 -pagesubset even -scale 1.0 -opacity 1.0 -percentagevals
true -showonscreen true -showonprint true -fixedprint true -runmetaData "john
doe;add wm" -logfile C:\temp\acscli.log -runpriority normal
```

TIMEOUT /t 20

```
ACSLCLI -watermarkpdf 5c4a72d0-081d-457f-96cc-0cca6e9ffc21 -cadfile
\\share\in\tdp.pdf -txt Released -font Arial -fontsize 144 -txtalign center
-txtcolor 255,0,0 -rot 90 -hdist 0.2 -halign right -vdist 0.3 -valign top
-startpage 0 -endpage 5 -pagesubset even -scale 1.0 -opacity 1.0 -percentagevals
true -showonscreen true -showonprint true -fixedprint false -runmetaData "john
doe;add wm" -logfile C:\temp\acscli.log -runpriority normal
```

## REMOVE WATERMARK FROM A PDF

The ACSCLI.exe utility allows you to remove watermarks from a PDF using Anark Publish. A run id (GUID) and the location of the PDF file needs to be specified. Other parameters, run metadata, run priority and log file are optional.

#### Syntax:

```
ACSLCLI -removepdfwatermark [runId] -cadfile <inputPDFFileLocation>
[-runmetaData <runMetaData>]
[-runpriority <high | normal | low>]
```

#### Example:



```
> ACSCLI.exe -removepdfwatermark fc803e47-c363-4217-b21c-d653f70405bf -cadFile
"\\share\in\tdp.pdf" -runmetadata "john doe;tdp, remove wm" -logfile
"C:\temp\acscli.log" -runpriority low
```

## ASSEMBLE A PDF

The ACSCLI.exe utility allows you to add attachments to a PDF using Anark Publish. A run id (GUID), location of the PDF file and the file paths for attachments separated by a semicolon if there are multiple files needs to be specified. Other parameters, run metadata, run priority and log file are optional.

### Syntax:

```
ACSCLI -assemblepdf [runId] -cadFile <inputPDFFileLocation>
      -attachFiles <file1;file2;...>
      [-runmetaData <runMetaData>]
      [-runpriority <high | normal | low>]
```

### Example:

```
> ACSCLI.exe -assemblePDF fc803e47-c363-4217-b21c-d653f70405bf -cadFile
\\share\in\tdp.pdf -attachfiles
\\share\in\attach\2d.pdf;\\share\in\attach\model.stp -runMetaData "john doe;tdp
prerelease; add attach" -logFile C:\temp\acscli.log -runpriority low
```

## DISASSEMBLE A PDF

The ACSCLI.exe utility allows you to save and/or delete the attachments stored in a PDF using Anark Publish. A run id (GUID), location of the PDF file and either the output folder where attachments needs to be saved or the delete parameter needs to be specified. Other parameters, run metadata, run priority and log file are optional.

### Syntax:

```
ACSCLI -disassemblepdf [runId] -cadfile <inputPDFFileLocation>
      [-outfolder <folderLocationToSaveFiles>]
      [-delete <deleteAttachments>]
      [-runmetadata <runMetaData>]
      [-runpriority <high | normal | low>]
```

### Example (Save attachments):

```
> ACSCLI.exe -disassemblePDF fc803e47-c363-4217-b21c-d653f70405bf -cadFile
"\\share\in\tdp.pdf" -outFolder "\\share\out\attachments" -runMetaData "john
doe;tdp prerelease; save" -logFile "C:\temp\acscli.log" -runpriority low
```

### Example (Delete attachments):

```
> ACSCLI.exe -disassemblePDF fc803e47-c363-4217-b21c-d653f70405bf -cadFile
"\\share\in\tdp.pdf" -delete true -runMetaData "john doe;tdp prerelease; delete"
-logFile "C:\temp\acscli.log" -runpriority low
```

### Example (Save and Delete attachments):

```
> ACSCLI.exe -disassemblePDF fc803e47-c363-4217-b21c-d653f70405bf -cadFile
"\\share\in\tdp.pdf" -outFolder "\\share\out\attachments" -delete true -
```



```
runMetaData "john doe;tdp prerelease; save and delete" -logFile
"C:\temp\acscli.log" -runpriority low
```

## GET COMMENTS FROM A PDF

The ACSCLI.exe utility allows you to retrieve comments from a PDF and output to a CSV file using Anark Publish. A run id (GUID) and location of the PDF file needs to be specified. Other parameters, output CSV file, run metadata, run priority and log file are optional. If the output CSV parameter is not specified, CSV will be written to the same folder as the input PDF file, the file name will be same as the input PDF file name. If the comment is a file attachment type, file attachment will be saved to the same folder as the output CSV file.

### Syntax:

```
ACSCLI -getpdfcomments [runId] -cadFile <inputPDFFileLocation>
[-outcsv <csvFile>]
[-runmetaData <runMetaData>]
[-runpriority <high | normal | low>]
```

### Example:

```
> ACSCLI.exe -getpdfcomments fc803e47-c363-4217-b21c-d653f70405bf -cadFile
"\\share\in\tdp.pdf" -outcsv "\\share\out\comments.csv" -runMetaData "john
doe;tdp prerelease;" -logFile "C:\temp\acscli.log" -runpriority high
```

## GET FIELDS FROM A PDF

The ACSCLI.exe utility allows you to retrieve fields from a PDF and output to a CSV file using Anark Publish. A run id (GUID) and location of the PDF file needs to be specified. Other parameters, output CSV file, run metadata, run priority and log file are optional. If the output CSV parameter is not specified, CSV will be written with the same name and to the same folder as the input PDF file.

### Syntax:

```
ACSCLI -getpdffields [runId] -cadFile <inputPDFFileLocation>
[-outcsv <csvFile>]
[-runmetaData <runMetaData>]
[-runpriority <high | normal | low>]
```

### Example:

```
ACSCLI -getpdffields fc803e47-c363-4217-b21c-d653f70405bf -cadFile
"\\share\in\tdp.pdf" -outcsv "\\share\out\fields.csv" -runMetaData "john doe;tdp
prerelease;" -logFile "C:\temp\acscli.log" -runpriority low
```

## ALL POSSIBLE ACSCLI.EXE RETURN CODES

ACSCLI.EXE can return any of the following return codes. The numeric return code value is available on the Windows command line with the %ERRORLEVEL% variable (e.g. *echo %ERRORLEVEL%*).



Return Code Description	Return Code Value
<b>Success</b>	0
<b>Error</b>	-1
<b>IncorrectArgs</b>	-2
<b>InvalidRecipeId</b>	-3
<b>InvalidCadLocation</b>	-4
<b>InvalidRunId</b>	-5
<b>ExceptionThrown</b>	-6
<b>RecipeCannotBeRetrievedFromDatabase</b>	-7
<b>RecipesNotValid</b>	-8
<b>RecipeContainsNoOrMoreThanOneImportAction</b>	-9
<b>RecipeContainsNoOrMoreThanOneExportAction</b>	-10
<b>RecipeImportCommandModificationError</b>	-11
<b>RecipeExportCommandModificationError</b>	-12
<b>RecipesMissingTheSpecifiedConnector</b>	-13
<b>RecipeConnectorModificationError</b>	-14
<b>RecipeCannotBeSaved</b>	-15
<b>AutomationProcessPathCannotBeObtained</b>	-17
<b>AutomationProcessCannotBeStarted</b>	-18
<b>InvalidServerName</b>	-19
<b>ServerConnectionError</b>	-20
<b>InvalidLogFile</b>	-21
<b>InvalidProtocol</b>	-22
<b>ProcessDoesNotExist</b>	-23
<b>FormatOptionNotFound</b>	-24
<b>NoWaterMarkTextSpecified</b>	-25
<b>InvalidRunPriority</b>	-26
<b>InvalidRecipeName</b>	-27
<b>RunCompleted</b>	1
<b>RunQueued</b>	2
<b>RunAccepted</b>	3
<b>RunInProgress</b>	4
<b>RunCompletedWithError</b>	5
<b>RunCompletedWithWarning</b>	6
<b>RunKilled</b>	7





<b>RunError</b>	<b>8</b>
-----------------	----------



## ANARK PUBLISH RESTFUL API

Anark Publish RESTful API is a modern interface for Anark Publish that is built into your Anark Publish install. Anark Publish RESTful API allows IT professionals to introduce the Anark Publish into a variety of existing enterprise workflows—with support for common scripting languages such as JavaScript, Perl, and Python, and compiled languages such as C, C++, Java, and Microsoft .NET languages such as C#, allowing interoperability with Windows, Linux, and other platforms.

There is an online help reference for the API at [http\[s\]://YOURSERVERNAME/coreservice/help](http[s]://YOURSERVERNAME/coreservice/help).

## USING ANARK RESTFUL API

### RUNNING ANARK RECIPES THAT HAVE BEEN DEPLOYED TO ANARK PUBLISH

Anark Publish RESTful API allows you to send remote messages to Anark Publish to run recipes. This functionality can be used to publish HTML content items and PDF documents, including multiple items from the same recipe (and same import). Recipe operations are queued on the server, so by default the RESTful API calls are non-blocking – they will often use a run id (a GUID) that can be used to check on the status of the submitted recipe run. The run id will be created for you if not specified, but usually you will want to create that id so that you can easily check the status of the run later. Run metadata can be specified as

The recommended way to run recipes is with a **POST** Run. The setup for this is as follows:

- 1) Create recipe using Anark Workstation
- 2) Deploy the recipe to Anark Publish
- 3) Perform a **GET** <http://YOURSERVERNAME/coreservice/api/Recipes?recipeName=RECIPENAME> to retrieve the options that can be overridden in your recipe
- 4) Perform a **POST** <http://YOURSERVERNAME/coreservice/api/Recipes> and specify the options, if any, that you wish to replace using the information from step 3.

### GET API/RECIPES?RECIPENAME={RECIPENAME}

#### URI Example:

**GET** <http://YOURSERVERNAME/coreservice/api/Recipes?recipeName=RECIPENAME>

Recipe “RECIPENAME” contains the following settings that can be overridden:

```
[
  {
    "ActionId": 5XX,
    "ActionType": "Import",
    "FormatId": "ACCC8D7D-7F97-4450-BA01-8699381A4FB5 (UnigraphicsFileFormat)",
    "SequenceNumber": 1,
    "SettingName": "File",
    "SettingValue": "\\server\\share\\mypart.prt",
```



```
    },  
    {  
      "ActionId": 5XX,  
      "ActionType": "Export",  
      "FormatId": "cde4b88c-b23e-4ccf-932c-0856f398ccf8 (PdfFileFormat)",  
      "SequenceNumber": 1,  
      "SettingName": "File",  
      "SettingValue": "\\server\\share\\output.pdf",  
    },  
    ...  
  ]
```

## GET API/RECIPES

### URI Example:

**GET** http://YOURSERVERNAME/coreservice/api/Recipes

```
[  
  {  
    "Name": "recipe1"  
  },  
  {  
    "Name": "recipe2"  
  }  
]
```

## GET API/RECIPES/RECIPENAME

### URI Example:

**GET** http://YOURSERVERNAME/coreservice/api/Recipes/recipeName

```
{  
  "Contents": "encoded recipe contents"  
}
```

## PUT API/RECIPES/RECIPENAME

### URI Example:

**PUT** http://YOURSERVERNAME/coreservice/api/Recipes/recipeName

```
{
```



```
"Contents": "encoded recipe contents"
}
```

## DELETEAPI/RECIPES/RECIPENAME

### URI Example:

**DELETE** http://YOURSERVERNAME/coreservice/api/Recipes/recipeName

## POST API/RECIPES

### ANARK COLLABORATE PUBLISHING TIPS:

When an Anark Collaborate publishing action is added to a recipe (in Anark Workstation), the unique content item id is recorded in the recipe so the content can be republished (overwritten) and so its metadata properties can be edited. The content item will be overwritten by default when the export action is refreshed or when the whole recipe is executed again. The option *ContentId* contains this content item id. To publish a new content item with an existing recipe without overwriting the original content item, specify an empty **replacementOptionValue** for this option. To retrieve the published content item id, wait until the run is complete, then retrieve the run log (use the **GET** Log call on pg. 45) **and look for this data:**

```
publishedContentInfo: {actionId: 5XX, target: 'Collaborate', contentId:
'5a0b27539056d257465d2735'}
```

If you don't know the run id or if you are unable to retrieve the logs for a specific run, the Anark Collaborate content item id can also be retrieved directly from existing content in one of the two following ways.

### When opening content directly (from ACW, from PLM, or with a content hyperlink):

The content item id is displayed in the browser URL. The URL is of the format:  
http://{ANARK\_COLLABORATE\_SERVER}/content/{content ID}/

The content item ID can be copied directly from the URL and used in subsequent requests.

### When opening content from Anark Collaborate App:

When accessing content from the Anark Collaborate App, the content opens in an iFrame. As such, the URL is the location of the background page, not the location of the iFrame content. The content item id of the content open in the iFrame can be retrieved by opening the browser console and entering the following string. The response will be the content item id:

```
$('#captureAppFrame').src.substring($('#captureAppFrame').src.indexOf('content')+8,$('#captureAppFrame').src.indexOf('content')+32)
```

### EXCEL / CSV IMPORT TIPS:



When using the Excel or CSV attribute data connector to produce workspace-level tables, the attribute name is based on the file name (excluding the path of the file). If your PDF or Anark Collaborate template references this table attribute specifically by name, it's important that you preserve the Excel or CSV file name when specifying a recipe update, for example: `\\\\this\\can\\change\\andthiscannot.xlsx`. It is also possible to create templates that are tolerant of a change in attribute name but it's easier to standardize the file name.

#### PDF PUBLISHING TIPS:

When specifying a recipe update for the option *Anark.Core.CadAdapter.Export.Pdf.EmbeddedFiles*, you must specify where to put the PDF template / 3D content using the text *PDF Template Placeholder*. The content will be embedded in the same order as specified. For example, this will add the `file_to_embed.pdf` content before the templated content:

```
"ReplacementValue": "\\server\\share\\file_to_embed.pdf|PDF Template Placeholder",
```

and this will add content before and after the templated content:

```
"ReplacementValue": "\\server\\share\\file_to_embed.pdf|PDF Template Placeholder|\\server\\share\\another_file_to_embed.pdf",
```

Note: unlike using ACSCLI to create an Anark Publish run, the Rest API needs all commands listed below and is order-dependent. You must list the **RunId**, **ActionId**, **ActionType**, **FormatId**, **SequenceNumber**, **SettingValue**, **SettingName**, and the **ReplacementValue**. Refer to the response from **GET Recipe** to fill values for **ActionId**, **SettingName**, and **SettingValue**. To specify an array, or list, of values, separate each value with a pipe character (|). For example: `"ReplacementValue": "file1|file2|file3"`.

#### Other Parameters:

- **RecipeName** – The name of the recipe that has been deployed to Anark Publish
- **RunMetaData** - Use this to specify any user metadata for this run (OPTIONAL)
- **RunPriority** – Priority of the run – Normal, Low or High. Server should be configured to process runs with different priorities. (OPTIONAL)
- **RunId** – Unique GUID for the run. Can be randomly generated. (OPTIONAL for JSON requests, REQUIRED for XML requests)
- **ActionId** – Is the unique identifier of the action in the recipe. Each action in the recipe has a unique ID that is created in the order in which actions were applied during the creation of the recipe. Different recipes may have different action IDs, so avoid referencing specific action IDs unless you need a very strong binding to a specific recipe. Use the value of -1 to mean "any action Id."
- **ActionType** – This is the action type, for example "Import" would be an import action and "Export" would be a publishing action. Use an empty string ("") to mean "any action type."



- **FormatId** – This provides an identifier that specifies the type of import or export action. For example, you could reference this to update the path of a CSV file while avoiding updates of Excel files. Use an empty string ("" ) to mean “any import or export format.”
- **SequenceNumber** – This is provided when the recipe contains more than one instance of the same action. If there are two Excel import actions, then the sequenceNumber of the first one would be 1 and the sequenceNumber of the second one would be 2. The sequenceNumber value is only shown when the value is greater than one but it is still present and can be referenced. For example, you could use this to update the first and/or second Excel import file path. Use a value of -1 to mean “any sequence number.”
- **SettingName** – Refers to the name of an option that can be specified by an action. Use an empty string ("" ) to mean “any setting name.”
- **SettingValue** – Value of the associated option. Use an empty string ("" ) to mean “any setting value.”
- **ReplacementValue** – Value to replace the **SettingValue**

**JSON Example:**

```
{
  "RunId": "",
  "RecipeName": "YOURRECIPEHERE",
  "RunMetaData": "",
  "RunPriority": "",
  "RecipeUpdates": [
    {
      "ActionId": 5XX,
      "ActionType": "Export",
      "FormatId": "cde4b88c-b23e-4ccf-932c-0856f398ccf8 (PdfFileFormat)",
      "SequenceNumber": 1,
      "SettingName": "File",
      "SettingValue": "\\ServerLocation\\originalPath\\originalPdfOutput.pdf",
      "ReplacementValue": "\\replacementServerLocation\\replacementPath\\replacementPdfOutput.pdf"
    },
    {
      "ActionId": 5XX,
      "ActionType": "",
      "FormatId": "",
      "SequenceNumber": -1,
      "SettingName": "ContentProperties",
      "SettingValue": "name1,value1|name2,value2|name3,value3",
      "ReplacementValue": "nameX,valueX|nameY,valueY|nameZ,valueZ"
    }
  ]
}
```



```

{
  "ActionId": 5XX,
  "ActionType": "",
  "FormatId": "",
  "SequenceNumber": -1,
  "SettingName": "Anark.Core.CadAdapter.Export.Pdf.EmbeddedFiles",
  "SettingValue": "\\server\\share\\file_to_embed.pdf|PDF Template Placeholder",
  "ReplacementValue": "PDF Template Placeholder|\\server\\share\\file_to_embed.pdf"
}
]
}

```

## POST ANARK COLLABORATE LOGIN

When using a Web Service testing client (such as Postman), you will need to **POST** an HTTP login request before retrieving or changing Anark Collaborate content item properties. The **POST** request will return a cookie that must be added in the header of any subsequent **GET** or **PUT** requests. The Body for the **POST** request should be of the “x-www-form-urlencoded” content type.

### URI Example:

**POST** http://{ANARK\_COLLABORATE\_SERVER}/api/auth/login

The **POST** request must contain two keys in the body:

- **login:** Base64 encoded Anark Collaborate user name
- **password:** Base64 encoded Anark Collaborate password

When the request is sent successfully (status code: ‘200 OK’), you will get a cookie in return. If you are using Postman, the cookie will be shown in the “Cookies” tab of the response. Otherwise, the cookie will be stored by the PLM system or by the browser when a user is logged in to Anark Collaborate.

Note: the ANARK\_COLLABORATE\_SERVER (where Anark Collaborate is accessed) is different than the SERVERNAME of the Anark Publish host machine (where Anark Publish is installed).

## GET ANARK COLLABORATE CONTENTPROPERTIES

FOR PDF DOCUMENTS:

Name	Description	Additional Information
<b>runMetaData</b>	Run metadata string to search for, it will return all the runs that contains the specified string in metadata (OPTIONAL)	Define this parameter in the request <b>URI</b>
<b>runId</b>	Run identifier (GUID) (OPTIONAL)	Define this parameter in the request <b>URI</b>



<b>runPriority</b>	Priority of the run – Normal, Low or High. Server should be configured to process runs with different priorities. (OPTIONAL)	Define this parameter in the request <b>URI</b>
<b>documentFilename</b>	Location of the PDF file to read	Define this parameter in the request <b>URI</b>
<b>outputFilename</b>	The CSV output file path and file name for the document properties	Define this parameter in the request <b>URI</b>

**URI Example:**

**GET** `http://SERVERNAME/coresevice/api/DocumentProperties?runId=&runMetaData=&runPriority=&documentFilename=\\server\folder\file.pdf&outputFilename=\\server\folder\file.csv`

## FOR ANARK COLLABORATE CONTENT ITEMS:

You can get all the properties from a certain content item or you can get a specific property value. Before sending the **GET** request, make sure that you have an authorization cookie. See the [POST Anark Collaborate Login](#) for more information on cookies.

**URI Example (all properties):**

**GET** `http://ANARK_COLLABORATE_SERVER/api/contents/{content ID}/properties`

**URI Example (specific property):**

**GET** `http://ANARK_COLLABORATE_SERVER/api/contents/{content ID}/properties/{property name}`

**POST API/DOCUMENTPROPERTIES**

Post api/DocumentProperties creates or overwrites preexisting PDF properties. Previously created document properties can be changed by referencing an existing PDF document property name and changing its value. Listing new document properties will not remove the previously created document properties. All previously created document properties can only be removed manually inside of the PDF.

You cannot edit Adobe standard properties “CreationDate,” “ModDate,” “Producer,” and “Trapped.” If these standard properties are added via Anark, they will be added to custom properties with a “--Text” suffix.

**Other Parameters:**

- **RunId** – Unique GUID for the run. Can be randomly generated. (OPTIONAL for JSON, REQUIRED for XML)
- **RecipeName** – The name of the recipe that has been deployed to Anark Publish.





- **RunMetaData** - Use this to specify any user metadata for this run. (OPTIONAL)
- **RunPriority** – Priority of the run – Normal, Low or High. Server should be configured to process runs with different priorities. (OPTIONAL)
- **DocumentFileName** – Full PDF file path that the user would like to add custom Adobe PDF properties to.
- **Name** – Name of user defined custom property.
- **Value** – Value of user defined custom property.

**JSON Example:**

```
{
  "RunId": "",
  "RecipeName": " YOURRECIPENAMEHERE ",
  "RunMetaData": "",
  "RunPriority": "normal",
  "DocumentFilename": "\\Server\Folder Path\File.Pdf",
  "Properties": [
    {
      "Name": "Custom Property Name 1",
      "Value": "Value 1"
    },
    {
      "Name": "Custom Property Name 2",
      "Value": "Value 2"
    },
    {
      "Name": "Custom Property Name 3",
      "Value": "Value 3"
    }
  ]
}
```

**PUT PROPERTIES**

Anark Collaborate content item properties can be added or altered using a **PUT** request. Previously created content item properties can be changed by referencing an existing Anark Collaborate content item id and giving a new value for the specified property. Listing new content item properties will not remove the previously created content item properties. Before sending the **PUT** request, make sure that you have an authorization cookie. See the [POST Anark Collaborate Login](#) for more information on cookies.

**URI Example:**

**PUT** `http://{ANARK_COLLABORATE_SERVER}/api/contents/{content ID}/properties`



In the body of the request, list the content item properties you want to add or change as raw JSON in the following format. A status of "204 No Content" indicates success.

```
{"property key": "new property value", "property key 2": "new property value 2"}
```

### GET API/LOGS/{ID}

Get the logs associated with the automation run.

Name	Description	Additional Information
<b>id</b>	Run identifier (GUID)	Define this parameter in the request <b>URI</b>

#### URI Example:

```
GET http://SERVERNAME/coreservice/api/Log/E6D4629E-CC23-4D32-8FDE-4BF1095D26AE
```

## API/RUNS

### RETRIEVE RUNS BY METADATA

Making this GET call will get the automation runs filtered by the specified run metadata and run status from Anark Publish.

Name	Description	Additional Information
<b>runMetaData</b>	Run metadata string to search for, it will return all the runs that contains the specified string in metadata	Define this parameter in the request <b>URI</b>
<b>runStatus</b>	Run status (OPTIONAL)	Define this parameter in the request <b>URI</b>

#### URI Example:

```
GET http://SERVERNAME/coreservice/api/Runs?runMetaData=SampleData&runStatus=
```

### RETRIEVE RUNS BY NUMBEROFRUNS

Making this GET call will get the last N automation runs.

Name	Description	Additional Information
------	-------------	------------------------



**numberOfRuns** Run metadata string to search for, it will return all the runs that contains the specified string in metadata Define this parameter in the request **URI**

**URI Example:**

**GET** `http://SERVERNAME/coreservice/api/Runs?numberOfRuns=100`

**RETRIEVE RUN BY GUID**

Get the automation run with a specific GUID.

Name	Description	Additional Information
<b>id</b>	Run identifier (GUID)	Define this parameter in the request <b>URI</b>

**URI Example:**

**GET** `http://SERVERNAME/coreservice/api/Runs/E6D4629E-CC23-4D32-8FDE-4BF1095D26AE`

**KILL A RUN**

Kill the automation run with a specific GUID.

Name	Description	Additional Information
<b>id</b>	Run identifier (GUID)	Define this parameter in the request <b>URI</b>

**URI Example:**

**PUT** `http://SERVERNAME/coreservice/api/Runs/E6D4629E-CC23-4D32-8FDE-4BF1095D26AE`

**POST API/WATERMARK**

The Anark Publish RESTful API allows you to add a watermark to a PDF using Anark Publish. Unlike using ACSCLI to create a watermark, the RESTful API needs all commands listed below.

**Parameters:**

- **PdfFileLocation** – Location of the PDF file.
- **RunId** – Unique GUID for the run. Can be randomly generated. (OPTIONAL for JSON, REQUIRED for XML)
- **RunMetaData** – Use this to specify user metadata for this run. (OPTIONAL)



- **RunPriority** – Priority of the run – Normal, Low or High. Server should be configured to process runs with different priorities. (OPTIONAL)
- **PdfWatermarkTextParams** (REQUIRED):
  - **srcText** – Watermark text.
  - **textAlign** – Watermark text align.
    - JSON: Specify 0 for left, 1 for center, and 2 for right.
    - XML: horizontal alignment specify kPDHorizLeft for left, kPDHorizCenter for center, and kPDHorizRight for right.
  - **fontName** – Watermark text font name.
  - **fontSize** – Watermark text font size in points.
  - **color** – Watermark text color - R, G, B values separated by a comma, range is from 0 to 255 (Default is black (0,0,0)). You may also specify common color names such as red, blue, etc.
- **PdfAddWatermarkParams** (REQUIRED):
  - **targetRange** – (OPTIONAL). If this is not provided, then all pages will be watermarked. When it is provided, it must be fully specified with values for startPage, endPage, and pageSpec, as follows:
    - **startPage** – Starting page index to which watermark should be applied. The first page index is 0.
    - **endPage** – Ending page index to which watermark should be applied. The first page index is 0. When the last page number is unknown in advance, specify a large number, for example 2147483647 (the largest 32-bit integer value) as a stand-in for the actual last page number.
    - **pageSpec** – Subset of the page range to which watermark should be applied
      - JSON: specify -3 for all pages, -4 for odd pages only, and -5 for even pages only.
      - XML: specify kPDAllPages for all pages, kPDOddPagesOnly for odd pages only, and kPDEvenPagesOnly for even pages only.
  - **fixedPrint** – Specifies whether the watermark maintain their size and position regardless of the dimensions of the target media. This option should be set to true for putting watermark over 3D
  - **zOrderTop** – Set to true to add the watermark to the foreground, set to false to add the watermark to the background.
  - **showOnScreen** – Specifies whether the watermark should be viewed on screen
  - **showOnPrint** – Specifies whether the watermark should be viewed on print
  - **horizAlign** – Watermark position
    - JSON: horizontal alignment, specify 0 for left, 1 for center, and 2 for right
    - XML: horizontal alignment, specify kPDHorizLeft for left, kPDHorizCenter for center, and kPDHorizRight for right
  - **vertAlign** – Watermark position
    - JSON: vertical alignment, specify 0 for top, 1 for center, and 2 for bottom
    - XML: vertical alignment, specify kPDVertTop for top, kPDVertCenter for center, and kPDVertBottom for bottom
  - **horizValue** – Watermark position - horizontal offset
    - With percentageVals set to “True”, horizValue will override your horizAlign settings
  - **vertValue** – Watermark position - vertical offset
    - With percentageVals set to “True”, vertValue will override your vertAlign settings
  - **percentageVals** – If this is true, horizontal and vertical offset position represent percentages of the page dimensions. Otherwise in user units
  - **scale** – Scale factor to be used when adding the watermark, with 1.0 meaning 100%



- **rotation** – Watermark counter-clockwise rotation in degrees
- **opacity** – Opacity to be used when adding the watermark, with 0.0 meaning fully transparent and 1.0 meaning fully opaque
- **wtrmrkDrawOption** – Defines the type of entity that will contain the watermark. This can have an effect with different PDF viewer implementations, such as Apple Preview.
  - JSON:
    - 0 - attempt to create the watermark as an annotation, however if these conditions (fixedPrint is true, zOrderTop is true) are not met then it will be drawn as a Form XObject.
    - 1 - attempt to create the watermark as a Form XObject, however if these conditions (fixedPrint is false, zOrderTop is false) are not met then it will be drawn as an annotation.
    - 2 - draw watermark as an Annotation, even if the option zOrderTop is false.
    - 3 - Draw watermark as a Form XObject, even if fixedPrint is true. This is the recommended value.
  - XML:
    - kPDWatermarkTryAnnotation - attempt to create the watermark as an annotation, however if these conditions (fixedPrint is true, zOrderTop is true) are not met then it will be drawn as a Form XObject.
    - kPDWatermarkTryFormXObject - attempt to create the watermark as a Form XObject, however if these conditions (fixedPrint is false, zOrderTop is false) are not met then it will be drawn as an annotation.
    - kPDWatermarkAnnotation - draw watermark as an Annotation, even if the option zOrderTop is false.
    - kPDWatermarkFormXObject - Draw watermark as a Form XObject, even if fixedPrint is true. This is the recommended value.

#### URI Example:

**POST** http://SERVERNAME/coreservice/api/Watermark

#### JSON Example:

```
{
  "PdfFileLocation": "\\server\path\File.pdf",
  "RunId": "63347e1f-dcbe-4ed7-a3cc-0ab885ab41e7",
  "RunMetaData": "",
  "RunPriority": "Normal",
  "PdfWatermarkTextParams": {
    "srcText": "Your watermark text here",
    "textAlign": 0,
    "fontName": "Times New Roman",
    "fontSize": 20,
    "color": "0,0,255"
  },
  "PdfAddWatermarkParams": {
    "targetRange": {
```



```

    "startPage": 0,
    "endPage": 1,
    "pageSpec": -3
  },
  "fixedPrint": true,
  "zOrderTop": true,
  "showOnScreen": true,
  "showOnPrint": true,
  "horizAlign": 0,
  "vertAlign": 0,
  "horizValue": 0,
  "vertValue": 0,
  "percentageVals": true,
  "scale": 1.0,
  "rotation": 0,
  "opacity": 1.0,
  "wtrmrkDrawOption": 3
}
}

```

If a Template requires a watermark, make sure that the font that is being used in the Adobe Form Fields can also be used to make a watermark. Watermarks can be made with much fewer fonts than the Adobe Form Fields

When creating a watermark in a 3D PDF, “fixedPrint” set to “true” WILL NOT appear ABOVE any objects that are not “burnt in” to the PDF by the 3D PDF JS. Using parameter “fixedPrint” set to “true”:

The Watermark will appear in front of 3D space but any color in the pdf that is not "burnt in" by the JavaScript will show above the watermark. This includes background color of form fields and text inside of a form field.

Using parameter “fixedPrint” set to “false”:

The watermark will appear in front of all form fields but will not appear in front of 3D space.

The recommended practice when using the parameter api/Watermark is to make two separate watermark calls. One using parameter “fixedPrint” set to “true” and another using parameter “fixedPrint” set to “false”.



## DELETE API/WATERMARK

Using the DELETE api/Watermark?pdfLocation={pdfLocation}&runId={runId}&runMetaData={runMetaData}&runPriority={runPriority} call will remove all watermarks that have been applied by Anark software to a PDF.

Name	Description	Additional Information
<b>pdfLocation</b>	Location of the PDF file	Define this parameter in the request <b>URI</b>
<b>runMetaData</b>	Run metadata string to search for, it will return all the runs that contains the specified string in metadata (OPTIONAL)	Define this parameter in the request <b>URI</b>
<b>runId</b>	Run identifier (GUID) (OPTIONAL)	Define this parameter in the request <b>URI</b>
<b>runPriority</b>	Priority of the run – Normal, Low or High. Server should be configured to process runs with different priorities. (OPTIONAL)	Define this parameter in the request <b>URI</b>

### URI Example:

**DELETE** `http://SERVERNAME/coreservice/api/Watermark?pdfLocation=\\server\folder\file.csv&runId=&runMetaData=sampleData&runPriority=normal`

## POST API/ATTACHMENTS

**POST** `api/Attachments?pdfLocation={pdfLocation}&attachments={attachments}&runId={runId}&runMetaData={runMetaData}&runPriority={runPriority}` will add file attachments to the PDF asynchronously. HTTP response contains the location header URI to query the run status.

Name	Description	Additional Information
<b>pdfLocation</b>	Location of the PDF file	Define this parameter in the request <b>URI</b>
<b>attachments</b>	File paths separated by a semicolon of files to attach	Define this parameter in the request <b>URI</b>
<b>runId</b>	Run identifier (GUID) (Optional)	Define this parameter in the request <b>URI</b>
<b>runMetaData</b>	Any user data that needs to be added to the run (Optional)	Define this parameter in the request <b>URI</b>
<b>runPriority</b>	Priority of the run – Normal, Low, or High. Server should be configured to process runs with different priorities. (OPTIONAL)	Define this parameter in the request <b>URL</b>

### URI Example:

**POST** `http://SERVERNAME/coreservice/api/Attachments?pdfLocation=\\server\folder\file.pdf&attachments=\\server\folder\model.stp&runId=&runMetaData=&runPriority=`



## DELETE API/ATTACHMENTS

**DELETE** `api/Attachments?pdfLocation={pdfLocation}&folderLocation={folderLocation}&runId={runId}&runMetaData={runMetaData}&runPriority={runPriority}` will delete and optionally save all the PDF attachments to the specified folder asynchronously. HTTP response contains the location header URI to query the run status.

### URI Example:

**DELETE** `http://SERVERNAME/coreservice/api/Attachments?pdfLocation=\\server\folder\file.pdf&attachments=\\server\folder\model.stp&runId=&runMetaData=&runPriority=`

Name	Description	Additional Information
<b>pdfLocation</b>	Location of the PDF file	Define this parameter in the request <b>URI</b>
<b>folderLocation</b>	Folder location to save attached files (Optional)	Define this parameter in the request <b>URI</b>
<b>runId</b>	Run identifier (GUID) (Optional)	Define this parameter in the request <b>URI</b>
<b>runMetaData</b>	Any user data that needs to be added to the run (Optional)	Define this parameter in the request <b>URI</b>
<b>runPriority</b>	Priority of the run – Normal, Low, or High. Server should be configured to process runs with different priorities. (OPTIONAL)	Define this parameter in the request <b>URI</b>

## GET API/ATTACHMENTS

**GET** `api/Attachments?pdfLocation={pdfLocation}&folderLocation={folderLocation}&runId={runId}&runMetaData={runMetaData}&runPriority={runPriority}` will delete and optionally save all the PDF attachments to the specified folder asynchronously. HTTP response contains the location header URI to query the run status.

Name	Description	Additional Information
<b>pdfLocation</b>	Location of the PDF file	Define this parameter in the request <b>URI</b>
<b>folderLocation</b>	Folder location to save attached files	Define this parameter in the request <b>URI</b>
<b>runId</b>	Run identifier (GUID) (Optional)	Define this parameter in the request <b>URI</b>
<b>runMetaData</b>	Any user data that needs to be added to the run (Optional)	Define this parameter in the request <b>URI</b>





<b>runPriority</b>	Priority of the run – Normal, Low, or High. Server should be configured to process runs with different priorities. (OPTIONAL)	Define this parameter in the request <b>URI</b>
--------------------	---	---

**URI Example:**

**GET** http://SERVERNAME/coreservice/api/Attachments?pdfLocation=\\server\folder\file.pdf  
&attachments=\\server\folder\model.stp&runId=&runMetaData=&runPriority=

**GET API/COMMENTS**

**GET** api/Comments?pdfLocation={pdfLocation}&csvLocation={csvLocation}&runId={runId}  
&runMetaData={runMetaData}&runPriority={runPriority} will get comments from the PDF asynchronously. HTTP response contains the location header URI to query the run status.

Name	Description	Additional Information
<b>pdfLocation</b>	Location of the PDF file	Define this parameter in the request <b>URI</b>
<b>runId</b>	Run identifier (GUID) (Optional)	Define this parameter in the request <b>URI</b>
<b>csvLocation</b>	Output CSV file location (Optional)	Define this parameter in the request <b>URI</b>
<b>runMetaData</b>	Any user data that needs to be added to the run (Optional)	Define this parameter in the request <b>URI</b>
<b>runPriority</b>	Priority of the run – Normal, Low, or High. Server should be configured to process runs with different priorities. (OPTIONAL)	Define this parameter in the request <b>URI</b>

**URI Example:**

**GET** http://SERVERNAME/coreservice/api/Comments?pdfLocation=\\server\folder\file.pdf  
&csvLocation=\\server\folder\file.csv&runId=&runMetaData=&runPriority=

**GET API/FIELDS**

**GET** api/Fields?pdfLocation={pdfLocation}&csvLocation={csvLocation}&runId={runId}  
&runMetaData={runMetaData}&runPriority={runPriority} will Get fields from the PDF asynchronously. HTTP response contains the location header URI to query the run status.

Name	Description	Additional Information
<b>pdfLocation</b>	Location of the PDF file	Define this parameter in the request <b>URL</b>
<b>csvLocation</b>	Output CSV file location (Optional)	Define this parameter in the request <b>URL</b>
<b>runId</b>	Run identifier (GUID) (OPTIONAL)	Define this parameter in the request <b>URL</b>



<b>runMetaData</b>	Run metadata string to search for, it will return all the runs that contains the specified string in metadata (OPTIONAL)	Define this parameter in the request <b>URL</b>
<b>runPriority</b>	Priority of the run – Normal, Low or High. Server should be configured to process runs with different priorities. (OPTIONAL)	Define this parameter in the request <b>URL</b>

**URI Example:**

**GET** `http://SERVERNAME/coreservice/api/Fields?pdfLocation=\\server\folder\file.pdf  
&csvLocation=\\server\folder\file.csv&runId=&runMetaData=&runPriority=`



## UPDATERECIPE.EXE COMMAND LINE APPLICATION

UpdateRecipe.exe allows IT professionals to script recipe updates in CAX files. It allows certain settings in CAX files to be examined and updated, as well as allowing imports and exports to be cloned or deleted. It will produce new CAX files containing the changes.

## INSTRUCTIONS FOR DEPLOYMENT

Under the “Tools” folder in the Anark SDK package, copy UpdateRecipe.exe and UpdateRecipe.exe.config into your Anark Workstation installation folder (for example, C:\Program Files\Anark\Anark Workstation). Please be sure to unblock the file first (properties -> unblock) if necessary.

## INSTRUCTIONS FOR USE

Open a command window (cmd.exe or PowerShell) and execute UpdateRecipe.exe to see example usage including required command line syntax and the available options.



## ANARK PUBLISH LOG VIEWER

The Anark Publish Log Viewer provides IT professionals with logs from the runs of your Anark Publish. The log viewer offers the choice of how many runs previous should be displayed, filtering based on server, run status, and metadata, and a convenient selection tool to simplify copying the table into Excel.

## INSTRUCTIONS FOR DEPLOYMENT

Under the “Anark Publish Log Viewer” folder in the Anark SDK package, copy Default.htm into the root directory of your web server.

## INSTRUCTIONS FOR USE

Open your web browser, append “/Default.htm” to your web server’s root URL, and navigate to the result (ex. [http\[s\]:// YOURSERVERNAME/Default.htm](http[s]://YOURSERVERNAME/Default.htm)).

