ARIS DataProcessing Tool for ArcGIS
User's Manual

4 July 2019

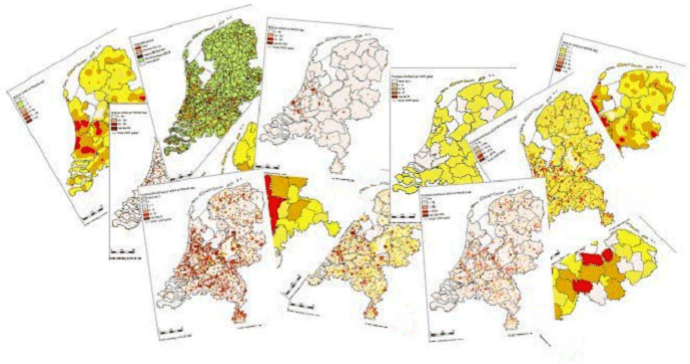
ARIS B.V.
http://www.aris.nl/

Table of contents

1 Introduction

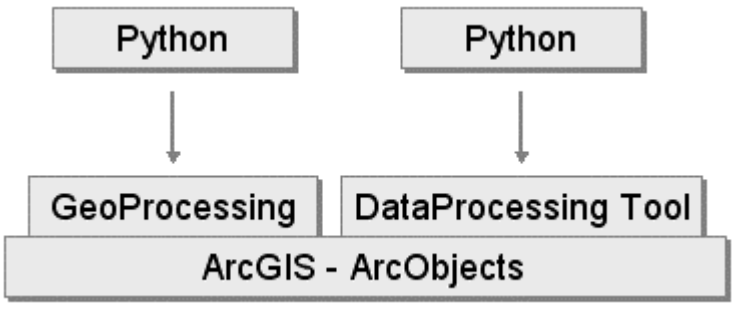
Some years ago the ARIS DataProcessing Tool has been developed by ARIS for the Netherlands Environmental Assessment Agency (PBL, dutch: Planbureau voor de Leefomgeving). The goal of the tool was to provide datamanagers with the possibility to automate processes involving loading data, generating metadata and publishing data easily using methods they already knew from ArcGIS Desktop.

Over the years the amount of data and metadata being used by PBL has grown and management of it became troublesome, error prone and time-consuming.



The idea for the tool originated from the ArcGIS GeoProcessing framework with Python as the scripting language. The ease of using Python and the ability to automate ArcGIS processes with the GeoProcessing framework worked well for geoprocessing but for datamanagement the framework was lacking functionality.

To resolve this the ARIS DataProcessing Tool was developed. The tool is a program library (DLL) which provides a collection of methods to access datasources, layerfiles and mxd-files and get information about these items or change their properties. These automation methods can be called from popular languages like Python, VBScript or JScript. The ARIS DataProcessing Tool implements automation using the COM IDispatch interface, making it possible for interpretative and macro languages to access the underlying ArcObjects functionality. You can use the ARIS DataProcessing Tool in the same way as you use the standard ArcGIS GeoProcessing automation library.



Due to this design the ARIS DataProcessing Tool is not developed for end-users but especially for programmers and data-administrators.

The past years the tool has been used at PBL for the following tasks:

- Loading data to a SDE database and prepare layerfiles and metadata for publication.
- Checking metadata for errors and correcting them automatically when possible.
- Mass mutating of metadata.
- Mass conversion of metadata from one metadata standard to another.
- Build an inventory of the data being used in MXD's within the PBL organization.

Since 2009, the tool was freely available and lots of organizations downloaded the tool. Due to changes in the information infrastructure at PBL, PBL asked ARIS to develop a new tool: the Metadata Processing Tool. This tool uses the FGDB API to manage metadata in FileGeodatabases. At that moment PBL stopped the maintenance of the DataProcessing Tool.

Because many organizations use the ARIS DataProcessing Tool we decided to support the tool in the future at our own costs. This is why the tool is no longer available for free and a small fee is necessary.

This manual only describes the ARIS DataProcessing Tool for ArcGIS 10.2 - 10.7.1. For other ArcGIS versions check the DataProcessing Tool website:
http://www.aris.nl/dataprocessing_arcgis

2 Installation

2.1 System requirements

To use the ARIS DataProcessing Tool the following software must be installed on your computer:
- Windows 7, 8 or 10.
- Microsoft .NET Framework 3.5.
- The appropriate **win32com.client library** for Python.*
- The appropriate ArcGIS and Python version:

DataProcessing Tool	ArcGIS	Python
4.6.2	10.2 / 10.2.1 / 10.2.2	2.7
4.6.3	10.3 / 10.3.1	2.7
4.6.4	10.4 / 10.4.1	2.7
4.6.5	10.5 / 10.5.1	2.7
4.6.6	10.6 / 10.6.1	2.7
4.6.7	10.7 / 10.7.1	2.7

For other ArcGIS versions check the DataProcessing Tool website:
http://www.aris.nl/dataprocessing_arcgis

* You can download the win32com client library from https://github.com/mhammond/pywin32/releases. Be sure to download the proper version according to your Python version.

2.2 Install

Note: Be sure to meet the requirements in the previous paragraph. Especially the **win32com.client library** for Python is important as it is not installed with ArcGIS.

To install the ARIS DataProcessing Tool click the Windows Installer Package *ARISDataProcessingSetup.msi*.

After installation, the following files must be present:
- ARISDataProcessingTool.dll
- ARISDataProcessingToolA.dll
- ARISDataProcessingToolA.exe
- ARISDataProcessingTool.pdf
- ARISDataProcessing.cfg
- arisdataprocessing.py

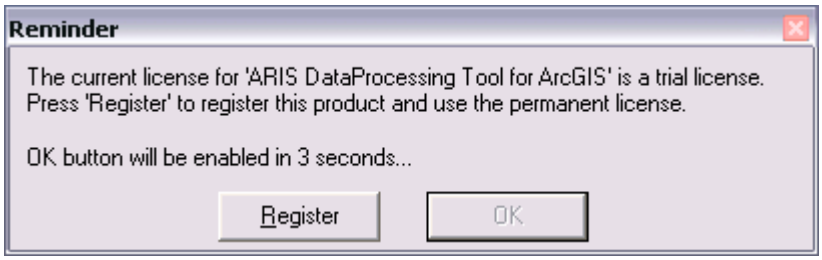
During installation a *arisdataprocessing.pth* file will be created in the Pythons Lib\site-packages directory. This ensures that the wrapper class file *arisdataprocessing.py* will be found when imported in user Python scripts.

In some cases it can be useful to copy the Example directory outside of the "Program Files" to prevent permission errors while using the examples.

3 Registration

3.1 Trial license

The distributed version of the ARIS DataProcessing Tool is an almost full functional version with a trial license. This means it can be used for evaluation purposes for 5 days. In this period the tool will process a maximum of 3 objects per run. After this period, the DataProcessing Tool will be locked until a valid licence key is entered.
While in trial mode, each time an object is retrieved for processing a reminder message will be shown.



Pressing the *Register* button will present you with the following dialog, where you must enter your name and the registration key (If you want to use DataProcessing Tool in trial mode, wait till the *OK* button comes available and press *OK*).

Note that after registering the tool from within ArcGIS, ArcGIS should be restarted for the permament license to be activated.

3.2 Register

If you do not yet have a registration key press the *Buy Now!* button. This will take you to our online store, where you can order this product. Note that you will need the hardware fingerprint of the computer where you want to install the tool, shown in the dialog above. After you complete your purchase, a personal registration key will be sent to you by email. Please store this key in a safe place.

Once you have entered a valid registration key, press *OK*. This key will be stored on your pc. The reminder message will not be shown again.

To register the DataProcessing Tool it is also possible to run ARISDataProcessingToolA.exe. This executable will show the reminder message to register the tool.

After registering the tool from within ArcGIS, ArcGIS should be restarted for the permament license to be activated.

Note: To store your license persistent between sessions it may be necessary to run the registration as administrator. Having administrator privileges may not be sufficient.

3.3 Unregister

When your license is not valid anymore through changes on your PC or if you want to move your license to a new PC, you can obtain a new license (fair use policy) after you unregister the license. Please contact us at helpdesk@aris.nl for assistance with the unregister process.

Send an e-mail to helpdesk@aris.nl with:
- ARIS product name and version
- Original hardware fingerprint (active license, if available)
- Registration name (active license)
- License key (active license)
- Confirmation code (from unregister, contact us for instruction)
- New hardware fingerprint (from register)

If you are entitled to receive a new license key, a new key will be sent to you by e-mail as soon as possible (same day, but might also take some days as this is not an automated process).

3.4 Invalid license

When major changes to your hardware have caused the license to become invalid, you can obtain a new valid license if you provide the necessary information (fair use policy).

To verify that hardware changes are the cause of the invalid license, you need to send a 'Hardware Change Log' file with the request for a new license. To get this Hardware Change Log contact us at helpdesk@aris.nl for assistance.

Send an e-mail to helpdesk@aris.nl with:
- ARIS product name
- Original hardware fingerprint (from license information sent by email)
- Registration name (from license information sent by email)
- License key (from license information sent by email)
- AHCL file (contact us for instruction)
- New hardware fingerprint (from Register/**Enter Key**)

4 Using the DataProcessing Tool

The ARIS DataProcessing Tool is designed to be used with script languages like Python, VBScript and JScript. In this Users Guide we only focus on using the ARIS DataProcessing Tool with Python.

4.1 Running scripts outside ArcGIS

In Python the automation methods of the ARIS DataProcessing Tool can be called directly using the *win32com.client* module.

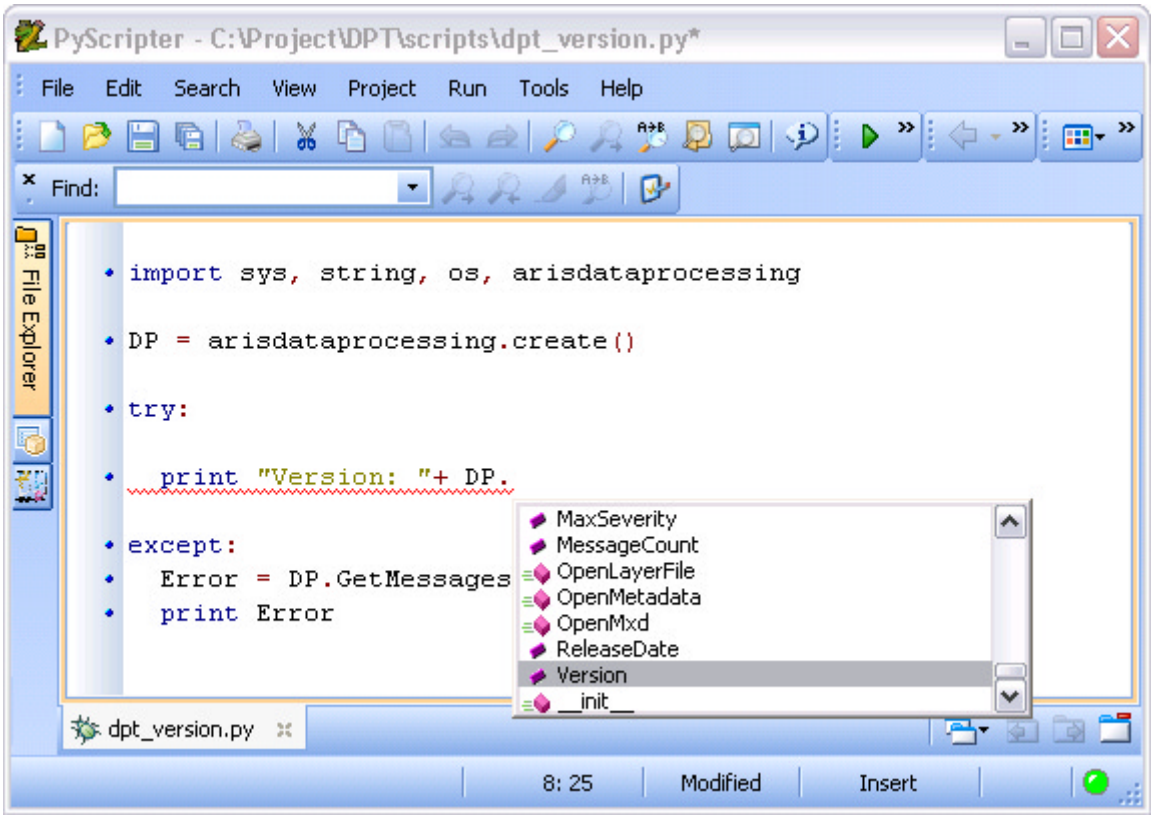
This example retrieves the version of the currently installed ARIS DataProcessing Tool.

```
import sys, string, os, win32com.client

DP = win32com.client.Dispatch("DataProcessing.DpDispatch")

try:

  print "Version: "+ DP.Version

except:
  Error = DP.GetMessages(2)
  print Error
```

Python scripts can be created, edited and run from the IDLE (Python GUI) which comes with ArcGIS. Just rightclick an existing python file and click Edit with IDLE. You can also use other IDE's like PythonWin or PyScripter. When using *PythonWin* or *PyScripter* for creating and editing Python scripts you can use the DataProcessing wrapper class ‘arisdataprocessing’ which enables inline code completion and argument information.
The code completion list can contain undocumented features of the DataProcessing wrapper class. These undocumented features have not been tested thoroughly. In a next version of the DataProcessing tool an undocumented feature might change to documented or to obsolete.

If the following message appears, you will have to install the pywin32 package as described in the Installation chapter:

ImportError: No module named win32com.client



This example gets the version of the currently installed ARIS DataProcessing Tool using the wrapper class.

```
import sys, string, os, arisdataprocessing

DP = arisdataprocessing.create()

try:

  print "Version: "+ DP.Version

except:
  Error = DP.GetMessages(2)
  print Error
```

Note: when running the scripts outside ArcGIS an ArcGIS license is needed. If no ArcGIS license is available the error 'RuntimeError: NotInitialized' may occur.

4.2 Running scripts inside ArcGIS

To run a python script in ArcGIS do the following:

1. In ArcCatalog or in the Catalog window in ArcMap select '*My Toolboxes*'.
2. Rightclick '*My Toolboxes*' and click *New* | *Toolbox*.
3. Give the new toolbox a proper name, for example DataProcessing.tbx.
4. Select the new toolbox.
5. Rightclick the new toolbbox and click *Add* | *Script*.
6. Type a name, label and description for the script and click *Next*.
7. Browse and select the script file you want to run and click *Next*.
8. Click Finish.
9. In ArcCatalog or in the Catalog window in ArcMap doubleclick the script.
10. In the tool window the message "This tool has no parameters." can be ignored as the example scripts do not need any parameters
11. Click OK to run the script.

If the following message appears in the Execution Dialog, you will have to install the pywin32 package as described in the Installation chapter:

ImportError: No module named win32com.client

4.3 Example scripts

All the examples described in this manual can be found in the <installation>\Example\Script directory. The data used in the examples can be found in the <installation>\Example\Data directory.

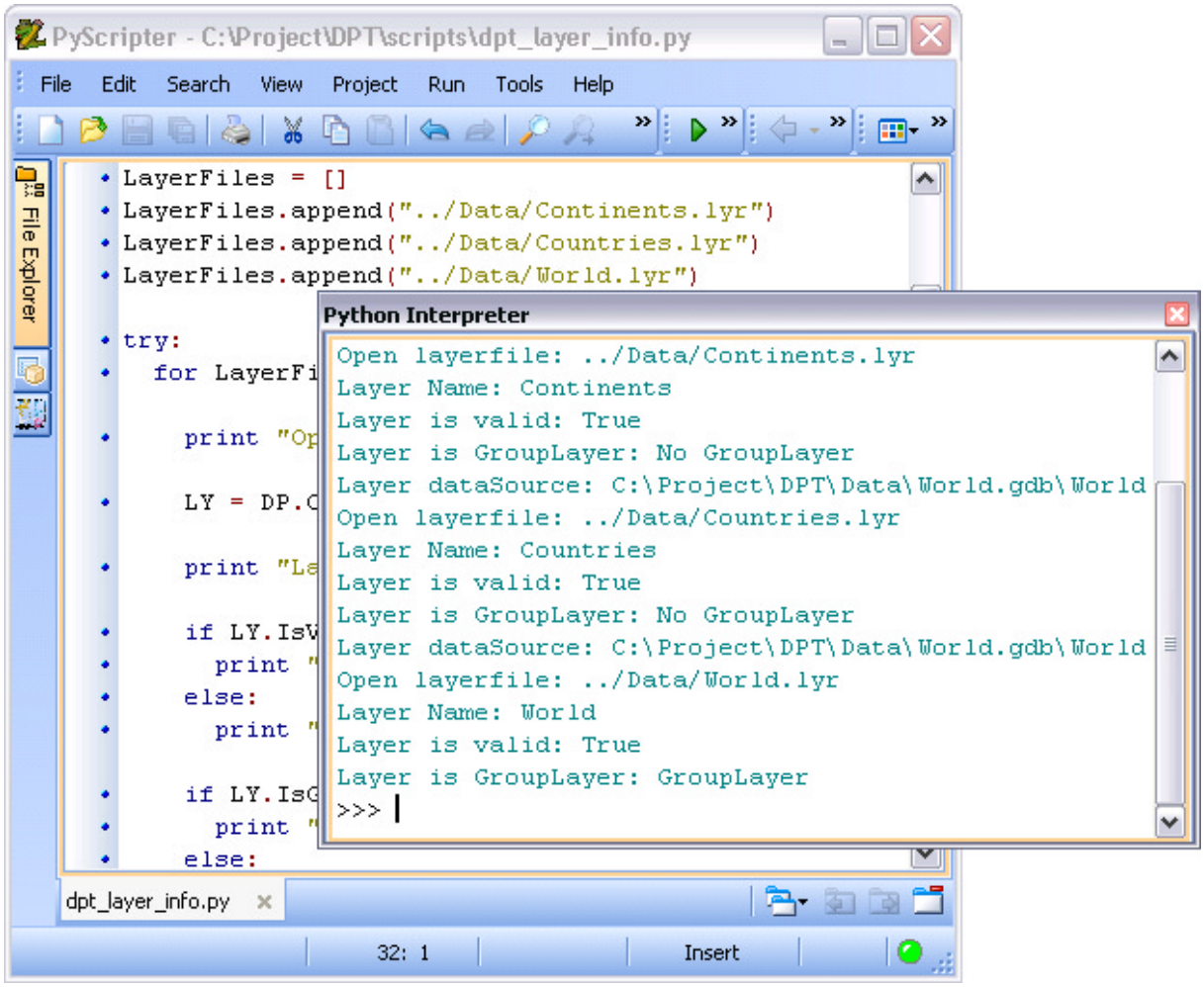
Preferably copy the example directory outside of the "Program Files" <installation> directory to prevent permission errors while using the examples.

4.3.1 Show layerfile info

An example of a practical task is shown by the next script (see also <installation>\Example\Script \dpt_layer_info.py). This script prints some information from some layerfiles.

```
import sys, string, os, arisdataprocessing, arisdataprocessing_util as dpu

DP = arisdataprocessing.create()

LayerFiles = []
LayerFiles.append("../Data/Continents.lyr")
LayerFiles.append("../Data/Countries.lyr")
LayerFiles.append("../Data/World.lyr")

try:
  for LayerFileName in LayerFiles:

    dpu.Msg("Open layerfile: " + LayerFileName)

    LY = DP.OpenLayerFile(LayerFileName)

    dpu.Msg("Layer Name: "+LY.Name)

    if LY.IsValid():
      dpu.Msg("Layer is valid: True")
    else:
      dpu.Msg("Layer is valid: False")

    if LY.IsGroupLayer():
      dpu.Msg("Layer is GroupLayer: GroupLayer")
    else:
      dpu.Msg("Layer is GroupLayer: No GroupLayer")

      if LY.IsValid():
        dpu.Msg("Layer dataSource: "+LY.DataSource)

  dpu.ShowWarnings(DP)
except:
  # Using exception handling ensures you getting a message about a trial or
  # expired license.
  dpu.ShowErrors(DP)
```

Running this script gives the following result:



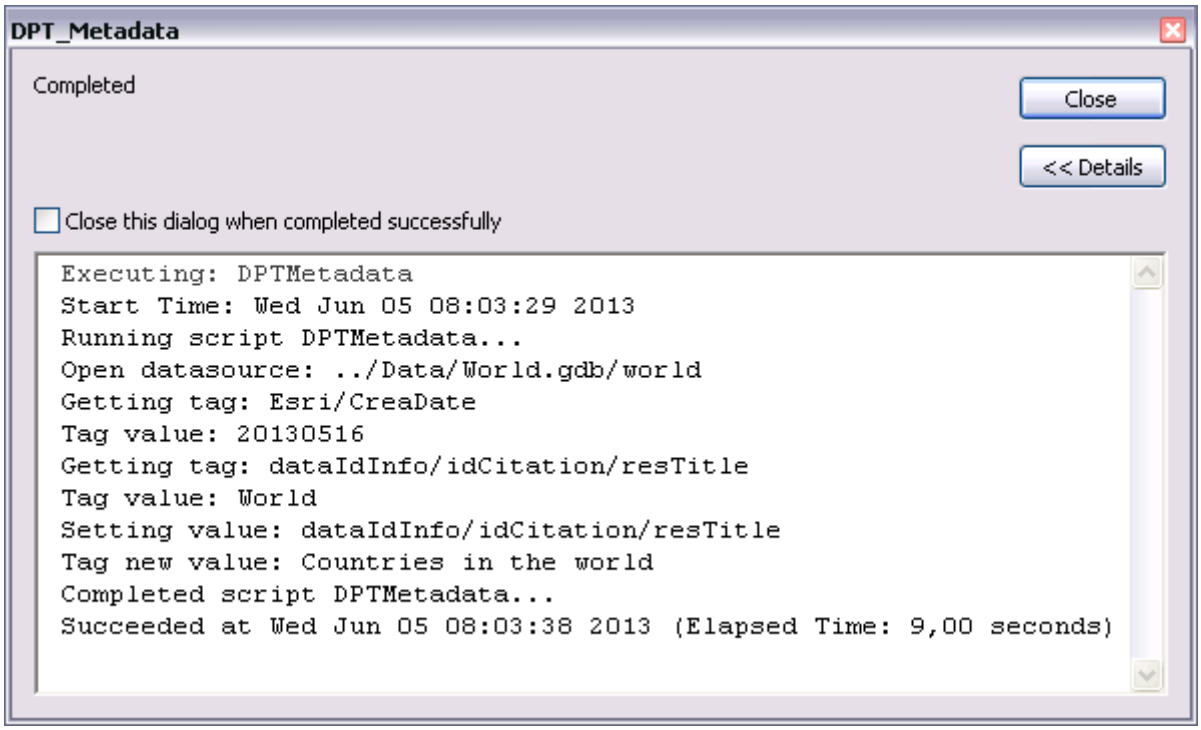
4.3.2 View and modify meta-information in a FeatureClass

The next example shows how to get meta-information from the specified FeatureClass in a filegeodatabase (see also <installation>\Example\Script\dpt_metadata.py).

```
import sys, string, os, arcpy, arisdataprocessing, arisdataprocessing_util as
dpu

DP = arisdataprocessing.create()

try:

  # Set datasource.
  Datasource = "../Data/World.gdb/world"

  # Check for metadata.
  if DP.HasMetadata(Datasource):

    # Open metadata.
    dpu.Msg("Open datasource: " + Datasource)
    MD = DP.OpenMetadata(Datasource)

    # Set metadata tag.
    Tag = "Esri/CreaDate"
```

```
      dpu.Msg("Getting tag: " + Tag)

      # Get metadata info.
      Value = MD.GetValue(Tag)
      dpu.Msg("Tag value: " + Value)

      # Set metadata tag.
      Tag = "dataIdInfo/idCitation/resTitle"
      dpu.Msg("Getting tag: " + Tag)

      # Get metadata info.
      Value = MD.GetValue(Tag)
      dpu.Msg("Tag value: " + Value)

      dpu.Msg("Setting value: " + Tag)
      if Value == "World":
        MD.SetValue(Tag, "Countries in the world")
      else:
        MD.SetValue(Tag, "World")

      # Get metadata info.
      Value = MD.GetValue(Tag)
      dpu.Msg("Tag new value: " + Value)
    else:
      dpu.Msg("No metadata available.")

    dpu.ShowWarnings(DP)
  except:
    # Using exception handling ensures you getting a message about a trial or
    # expired license.
    dpu.ShowErrors(DP)
```

After adding this script to a Toolbox in ArcMap and running it in ArcMap it generates the following result.



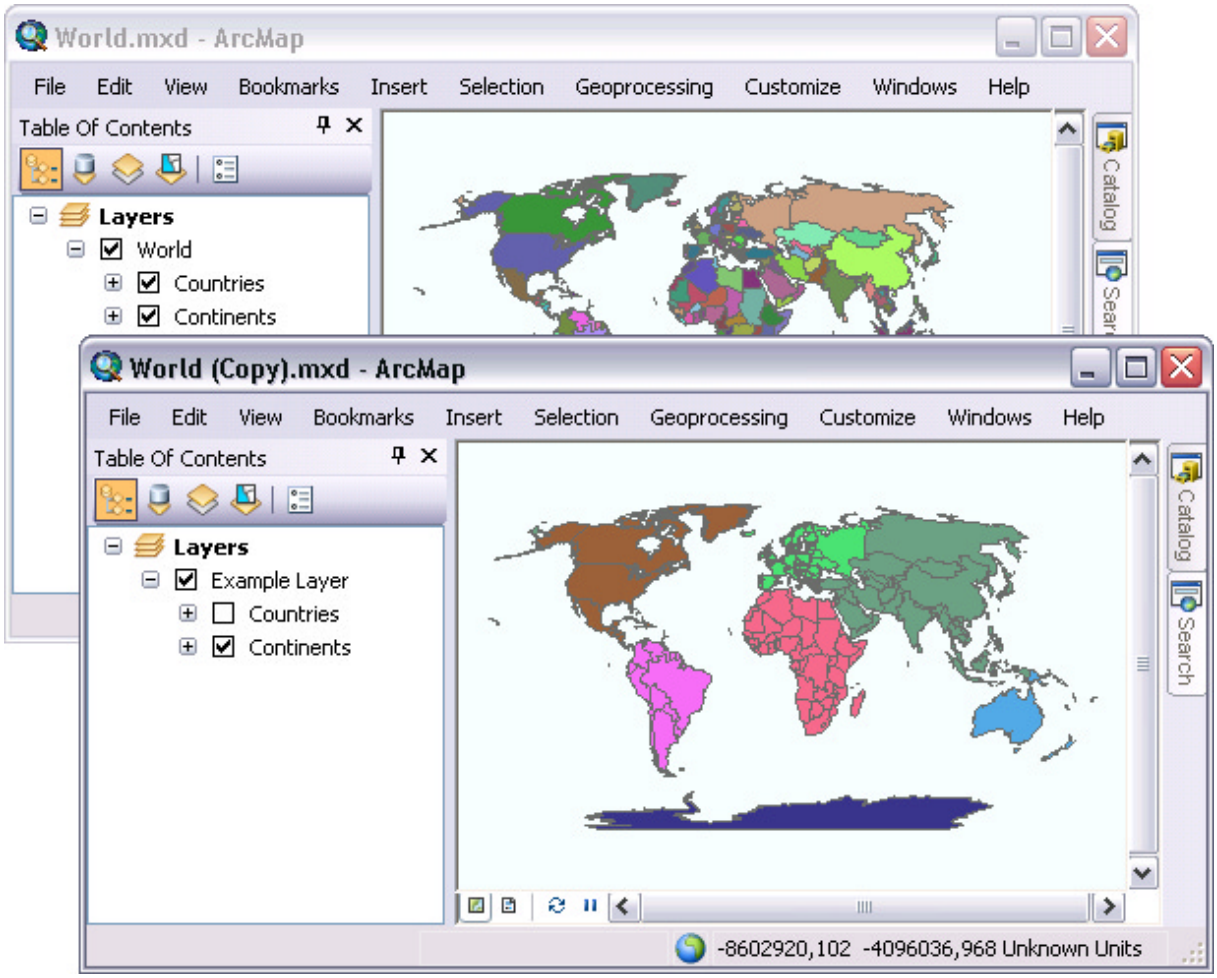
The above examples show how the ARIS DataProcessing Tool can be used to perform several tasks for managing geodata.

4.3.3 Modify mxd file

The next example shows how to change the name and visibility of layers in a mxd file (see also <installation>\Example\Script\dpt_mxdfile.py).

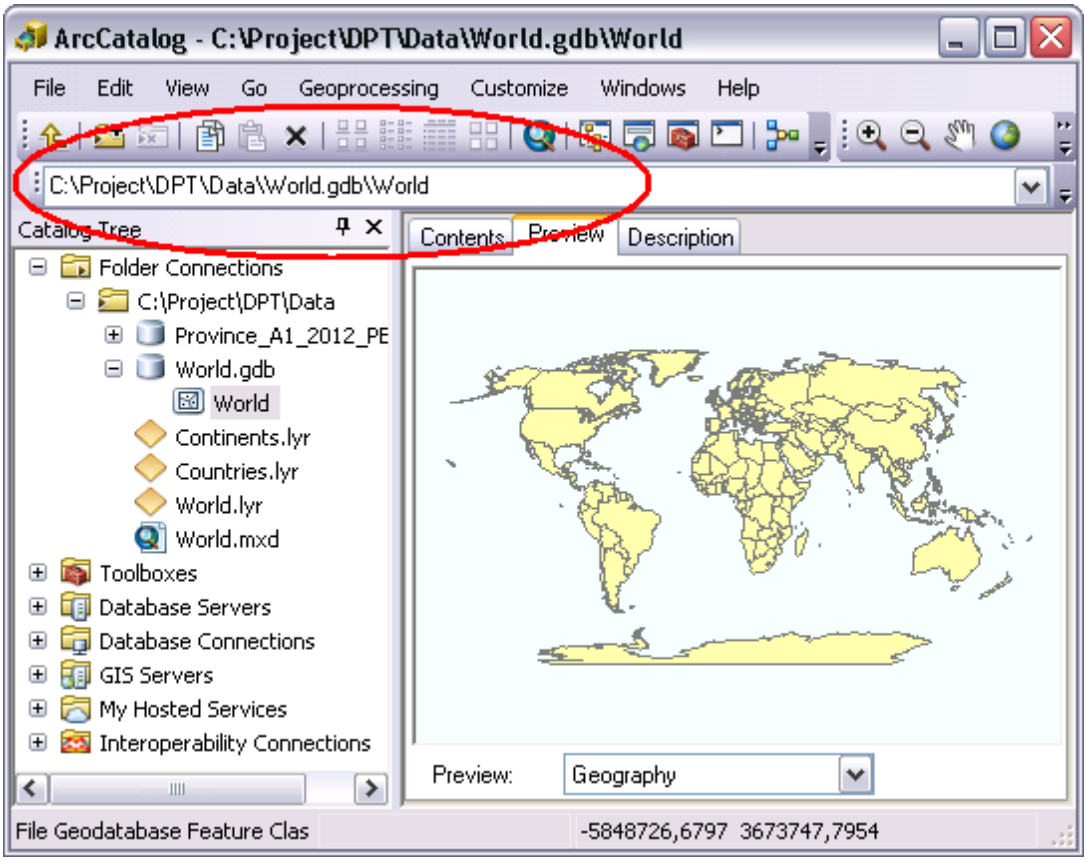
```
import sys, string, os, shutil, arcpy, arisdataprocessing,
arisdataprocessing_util as dpu

DP = arisdataprocessing.create()

try:

  # Set datasources.
  Datasource = os.path.abspath("../Data/World.mxd")
  DatasourceCopy = os.path.abspath("../Data/World (Copy).mxd")

  # Copy the datasource if exists.
  if os.path.exists(Datasource):
    dpu.Msg("Copying " + Datasource + " to " + DatasourceCopy + "...")
    shutil.copy(Datasource, DatasourceCopy)

  if os.path.exists(DatasourceCopy):

    # Open the mxd file
    dpu.Msg("Open " + DatasourceCopy + "...")
    Mxd = DP.OpenMxd(DatasourceCopy)

    # Get the dataframe
    DF = Mxd.OpenDataFrame(0)

    # Change the name of the first layer in the dataframe.
    LY = DF.OpenLayer(0)
    dpu.Msg("Changing layer name...")
    LY.Name = "Example Layer"

    # Change the visibility of the first sublayer
    dpu.Msg("Changing layer visibility...")
    if LY.IsGroupLayer:
      SLY = LY.OpenSubLayer(0)
      SLY.Visible = False

    dpu.Msg("Open both mxd files in ArcMap and compare them to see the
changes.")

  dpu.ShowWarnings(DP)
except:
  # Using exception handling ensures you getting a message about a trial or
  # expired license.
  dpu.ShowErrors(DP)
```

The image beneath shows the original and the copied and changed mxd file.



4.4 Datasources and ArcCatalog paths

The ARIS DataProcessing Tool provides several functions for accessing and manipulating datasources.
In almost all these cases an ArcCatalog path must be used for referencing the datasource. The ArcCatalog path is the path reported in the ArcCatalog Location toolbar. The ARIS DataProcessing Tool uses this path to find the datasource.



The ArcCatalog path for a shapefile is simply the path to the folder containing the shapefile and the shapefile's name, including its .shp extension. A shapefile containing roads located in the folder C:\GeoData would have an ArcCatalog path of "C:\GeoData\roads.shp".

Feature classes in a personal geodatabase reside in an Access database file, and enterprise geodatabase feature classes are found in a Relational Database Management System (RDBMS). The ArcCatalog path to a personal geodatabase has the disk location of the Access file. A feature class name is simply added to that path if it is standalone, resulting, for example, in a path of "C:\GeoData\Data.mdb\rivers".

Instead of a path to an Access file, paths to data in an enterprise geodatabase contain the location of the file defining the database connection. The default location for this information is Database Connections in ArcCatalog, so a typical path to a standalone feature class in an enterprise geodatabase may appear as "Database Connections\Connection to GeoData.sde\reed.roads".

To retrieve the correct ArcCatalog path of your datasource use the Location toolbar in ArcCatalog to check the dataset or workspace path.

In case of a datasource in an SDE geodatabase an alternative format can be used. In addition to using a ArcCatalog path you can use a string with the connection properties of the datasource. This string must have the following format: <server>|<instance>|<username>|<password>|<featuredataset name>|<dataset name>. The fields <server> and <featuredataset name> may be left empty.
An example of a connection string to a dataset in an "Oracle Direct Connect" SDE geodatabase is:
"|sde:Oracle10g:/;LOCAL=prod_ihc_svr01|dbuser|xxx|NL.werken|NL.adep_2005"

4.5 Metadata

When accessing metadata from datasources XPath tags (also known as XSL Patterns) are used to identify the information items (i.e. the xml nodes) in the metadata. An important point when using XPath tags is that they are **case sensitive**.

Some valid XPath tags examples are:

- Gets the creation date of the datasource.

```
/metadata/Esri/CreaDate
```

- Gets the creation date of the datasource. Omitting the forward slash (/) at the beginning of the XPath the tool will interprete the XPath tag as relative to the default root node <metadata>.

```
Esri/CreaDate
```

- Gets the language of the metadata with namespaces.

```
gmd:MD_Metadata/gmd:language/gco:CharacterString
```

When accessing metadata which uses namespace prefixes (by example ISO19115) the prefixes does not need to be specified in the XPath tags if a default namespace is defined.

If the metadata contains xml nodes which can be occur more then once, indexes can be used to specify subsequent nodes. Beware, the index is **zero-based**.

Some examples with indexes are:

- Gets the alias name of the 4th attribute of the datasource.

```
eainfo/detailed/attr[3]/attalias
```

- Gets the creation date of the datasource. This is also a valid xml tag. The index 0 stands for the first xml node and is usually omitted.

```
Esri/CreaDate[0]
```

By using XPath tags the ARIS DataProcessing Tool is **not** bound to a specific metadata profile (CEN3, CEN4, ISO19115 etc.). The ARIS DataProcessing Tool is **profile independent**.

The ARIS DataProcessing Tool can access metadata from all supported datasources, except webservices. Also accessing metadatawebservices is not supported.

For accessing the metadata of a datasource the metadata should be saved at the standard ESRI location, i.e. in the personal geodatabase, in ArcSDE, in a shp.xml file etc.

In addition to the ESRI datasources the ARIS DataProcessing Tool also supports single files and folders with metadata. In case of single files (for example Word documents, PDF documents, Excel spreadsheets, etc.) the metadata should be in a separate file called <filename>.doc.xml, <filename>.pdf.xml, <filename>.xls.xml etc.. In case of folders the metadata should be in a file called metadata.xml within the folder.

4.6 Using metadata shortcuts

To make working with metadata a lot easier the ARIS DataProcessing Tool supports using shortcuts instead of XPath tags. Shortcuts are names, always preceded by a %, which are defined in a so-called **shortcuttemplate file**. This template file has the same structure as the metadata files you want to use. Instead of metadata information it has shortcut names on those places you want to access.

Suppose we have metadata with the following profile:

```
<?xml version="1.0"?>
<metadata xml:lang="en">
```

```
  <Esri>
    <MetaID>{EF3E94EF-BE94-4466-9B38-74495B832928}</MetaID>
    <CreaDate>20030613</CreaDate>
    <CreaTime>16270800</CreaTime>
  </Esri>
  ...
</metadata>
```

To define the shortcuts %FileID and %CreationDate create the following shortcuttemplate file:

```
<?xml version="1.0"?>
<metadata xml:lang="en">
  <Esri>
    <MetaID>%FileID<MetaID>
    <CreaDate>%CreationDate</CreaDate>
    <CreaTime></CreaTime>
  </Esri>
  ...
</metadata>
```

Save this shortcuttemplate file on your system and add a reference to the configuration file ARISDataProcessingTool.cfg in the ARIS DataProcessing Tool installation directory. For example:

```
<?xml version="1.0" encoding="utf-8" ?>
<Config>
  <ShortcutTemplates>
    <File>ShortcutTemplate\MetadataKeywordsCEN4.xml</File>
    <File>C:\MetadataTemplates\MetadataKeywordsMyProfile.xml</File>
  </ShortcutTemplates>
</Config>
```

For accessing the creation date of a datasource you can now use `%CreationDate` instead of `Esri/CreaDate`. Especially when tags have more subtags and become long or when they are cryptic, using shortcuts is a lot easier and will result in less errors when accessing metadata.

Important: When creating multiple shortcut templates be sure that all shortcut names are unique.

The following script is an example how to access the metadata of a datasource in a filegeodatabase (see also also <installation>\Example\Script\dpt_metadata_iso.py).

```
import sys, string, os, arcpy, arisdataprocessing, arisdataprocessing_util as dpu

DP = arisdataprocessing.create()

try:

  # Set datasource.
  Datasource = "../Data/Provinces.gdb/Provinces"

  # Check for metadata.
  if DP.HasMetadata(Datasource):

    # Open metadata.
    dpu.Msg("Open datasource: " + Datasource)
    MD = DP.OpenMetadata(Datasource)

    #-------------------------------------------------------
    # Get metadata by Tag.
    #-------------------------------------------------------

    # Set metadata ISO MD tag.
    MDTag = "gmd:MD_Metadata"
```

```
    # Set metadata ISO IdentificationInfo tag.
    IdentTag = MDTag + "/gmd:identificationInfo/gmd:MD_DataIdentification"

    # Set metadata ISO Citation tag.
    CitTag = IdentTag + "/gmd:citation/gmd:CI_Citation"

    # Set metadata ISO Title tag.
    TitleTag = CitTag + "/gmd:title/gco:CharacterString"
    dpu.Msg("Getting Title...")

    # Get metadata info.
    Value = MD.GetValue(TitleTag)
    dpu.Msg("Title by Tag: " + Value)

    #---------------------------------------------------------
    # Get metadata by Shortcut.
    #---------------------------------------------------------

    # Set metadata ISO Title shortcut.
    TitleShortcut = "%ISO.title"

    # Get metadata info.
    Value = MD.GetValue(TitleShortcut)
    dpu.Msg("Title by Shortcut: " + Value)

  else:
    dpu.Msg("No metadata available.")

  dpu.ShowWarnings(DP)
except:
  # Using exception handling ensures you getting a message about a trial or
  # expired license.
  dpu.ShowErrors(DP)
```

After adding this script to a Toolbox in ArcMap and running it in ArcMap it generates the following result.

As with normal tags, indexes can be used with shortcuts too. You define the shortcut in the usual way:

```
<?xml version="1.0"?>
<metadata xml:lang="en">
  <eainfo>
```

```
    <detailed>
      <attr>
        <attalias>%AliasName<attalias>
  ...
</metadata>
```

For accessing the aliasname of the 3rd field you can use the following shortcut:

```
%AliasName[2][0]
```

This shortcut will be translated into this tag for accessing the metadata:

```
eainfo/detailed/attr[2]/attalias[0]
```

In order to know at what subtag level the index should be applied, dummy 0 indexes should be added at the end.

5 Supported datasources

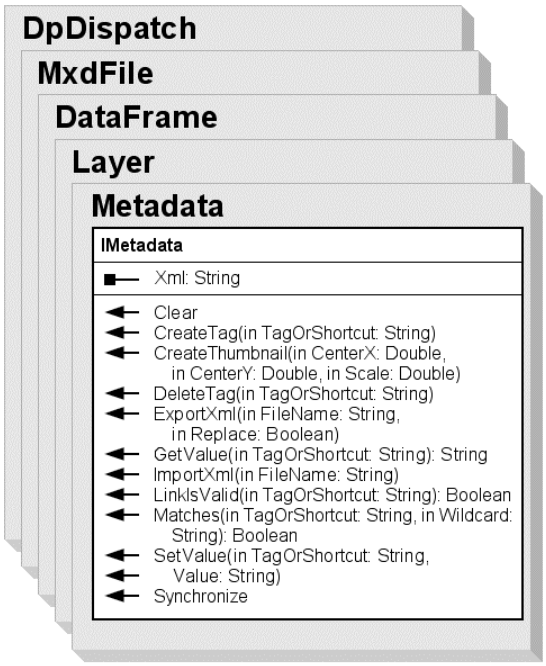
At this moment the ARIS DataProcessing Tool supports the following datasources:

- ArcInfo Coverages
- Shapefiles
- Raster datasets
- Raster Catalogs
- Personal Geodatabase Feature Classes
- Personal Geodatabase Raster datasets
- Personal Geodatabase Raster Catalogs
- File Geodatabase Feature Classes
- File Geodatabase Raster datasets
- SDE Feature Classes (by Oracle Direct Connect)
- SDE Raster datasets (by Oracle Direct Connect)
- SDE Raster Catalogs (by Oracle Direct Connect)
- ArcIMS Image Services
- ArcIMS Feature Services
- XY Event Sources
- Tables
- Single files (for example Word documents, PDF documents, Excel spreadsheets, etc.)
- Folders

Support for using non-'Oracle Direct Connect' SDE connection types was not tested recently and **may not work**!

6 Object model

The purpose of the ARIS DataProcessing Tool is to retrieve and change metadata from datasources, and the contents of layerfiles and mxdfiles.



To perform these tasks the ARIS DataProcessing Tool consists of the following objects or classes:

- DpDistpatch

  This is the main object of the ARIS DataProcessing Tool and the starting point to retrieve references to the other objects. The methods to get these references are:
  - CreateMetadata
  - OpenMetadata
  - CreatelayerFile
  - OpenLayerFile
  - CreateMxd
  - OpenMxd

  It also facilitates some general functions concerning version information and error management.

- Metadata

  This object provides several methods to access the metadata of datasources and datafiles. Some of these methods are:
  - CreateTag
  - GetValue
  - SetValue
  - Matches
  - Xml
  - ExportThumbnail
  - ExportXml
  - ImportXml

- MxdFile

This object provides methods to access the dataframes (i.e. maps) in a mxdfile. These methods are:
- CreateDataFrame
- NrOfDataFrames
- OpenDataFrame
- ListDataFrames

- DataFrame

  This object provides methods to access the dataframe properties and the layers in the dataframe. Some of these methods are:
  - Name
  - NrOfLayers
  - OpenLayer
  - CreateLayer

- Layer

  This object provides methods to access the layer properties. Some of these methods are:
  - Name
  - IsValid
  - IsGroupLayer
  - Visible
  - MinScale
  - MaxScale
  - WhereClause

  It also provides methods to access sublayers in the layer. This only works with GroupLayers. Some of these methods are:
  - NrOfSubLayers
  - OpenSubLayer
  - CreateSubLayer
  - ListSubLayers

  The ARIS DataProcessing Tool provides several ways to retrieve a reference to a Layer object. These are:
  - DpDispatch.CreateLayerFile and DpDispatch.OpenLayerFile
  - DataFrame.CreateLayer and DataFrame.OpenLayer
  - Layer.CreateSubLayer and Layer.OpenSubLayer

6.1 DpDispatch Object

Provides access to the properties/methods of the DataProcessing DpDispatch object.

Members

	Description
CreateLayerFile	Creates a new layerfile.
CreateMetadata	Creates metadata for a dataset or datafile.
CreateMxd	Creates a new mxd file.
GetMessage	Get the return message by index.
GetMessages	Get all the return messages.
GetTagFromShortcut	Returns the xml-tag of a shortcut.
HasMetadata	Returns if a datasource has metadata.
MaxSeverity	The maximum returned severity.
MessageCount	The number of returned messages.
OpenLayerFile	Opens a LayerFile.
OpenMetadata	Opens the metadata of a dataset or datafile.
OpenMxd	Opens a mxd file.
ReleaseDate	Returns the current DataProcessing release date.
Version	Returns the current DataProcessing version.

CreateLayerFile Method

Creates a new layerfile from a datasource.

Syntax
variable = *object*.**CreateLayerFile (***LayerFileName, DataSource* **)**

The **CreateLayerFile** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **DpDispatch** object.
variable	A reference to a Layer object.
LayerFileName	Required. The filename of the LayerFile.
DataSource	Required. The name of the DataSource.

Remarks
The DataSource must be a valid ArcCatalogPath.

CreateMetadata Method

Creates the metadata for a dataset or datafile, if no metadata exists.

Syntax
variable = *object*.**CreateMetadata (***DataSource,*[*InitalXml*] **)**

The **CreateMetadata** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **DpDispatch** object.
variable	A reference to a Metadata object.
DataSource	Required. The name of the DataSource.

InitialXml	Optional. The initial xml of the created metadata.

Remarks
The DataSource must be a valid ArcCatalogPath.

The InitialXml string must be valid xml. In no InitialXml string is specified the created metadata wil be initialized with the following xml: *<?xml version="1.0"?><metadata></metadata>*.

Also a ConnectionInfo string can be used. A ConnectionInfo string is a '|' delimited list of SDE connection properties. The following properties **must** be used: server, instance, user, password, featuredatasetname and datasetname. The field featuredatasetname may be left empty.

If the metadata already exists no error message will be generated.

This method cannot be used using **webservices**.

CreateMxd Method

Creates a new mxd file.

Syntax
variable = *object*.**CreateMxd (***MxdFileName* **)**

The **CreateMxd** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **DpDispatch** object.
variable	A reference to a MxdFile object.
MxdFileName	Required. The name of the Mxd.

Remarks
This method creates a mxd file with 1 dataframe.

GetMessage Method

Get the return message by index.

Syntax
object.**GetMessage (***Index* **)**

The **GetMessage** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **DpDispatch** object.
Index	Required. A Integer that represents the index.

Return value
String

GetMessages Method

Get all return messages.

Syntax
object.**GetMessages ([***Severity***])**

The **GetMessages** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **DpDispatch** object.
Severity	Optional. A Variant that represents the severity.

Return value
String

Remarks
The following severity levels can be used:
0 Informative messages
1 Warning messages
2 Error messages

If no severity is given, all messages are returned.

GetTagFromShortcut Method

Returns the xml-tag of a shortcut.

Syntax
object.**GetTagFromShortcut (** *Shortcut* **)**

The **GetTagFromShortcut** method syntax has the following object qualifier and arguments:

Part	Description
Object	An object placeholder that evaluates to a **DpDispatch** object.
Shortcut	Required. The shortcut.

Return value
String

HasMetadata Method

Returns if a datasource has metadata.

Syntax
object.**HasMetadata (***DataSource* **)**

The **HasMetadata** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **DpDispatch** object.
DataSource	Required. The name of the DataSource.

Return value
Boolean

Remarks
The DataSource must be a valid ArcCatalogPath.

Also a ConnectionInfo string can be used. A ConnectionInfo string is a '|' delimited list of SDE connection properties. The following properties must be used: server, instance, user, password, featuredatasetname and datasetname. The field featuredatasetname may be left empty.

MaxSeverity Property

The maximum returned severity.

 Read only (Includes Get_MaxSeverity)

Syntax
variable = *object*.**MaxSeverity**

The *object* placeholder represents a **DpDispatch** object.

Return value
Integer

MessageCount Property

The number of returned messages.

 Read only (Includes Get_MessageCount)

Syntax
variable = *object*.**MessageCount**

The *object* placeholder represents a **DpDispatch** object.

Return value
Integer

OpenLayerFile Method

Opens an existing LayerFile.

Syntax
variable = *object*.**OpenLayerFile (** *LayerFileName* **)**

The **OpenLayerFile** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **DpDispatch** object.
variable	A reference to a Layer object.
LayerFileName	Required. The filename of the LayerFile.

Remarks
After opening a layerfile you can use **DataSource** and **DataType** to get or set layerfile properties.

OpenMetadata Method

Opens the metadata of a dataset or datafile.

Syntax
variable = *object*.**OpenMetadata (** *DataSource* **)**

The **OpenMetadata** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **DpDispatch** object.

variable	A reference to a Metadata object.
DataSource	Required. The name of the DataSource.

Remarks
The DataSource must be a valid ArcCatalogPath.

Also a ConnectionInfo string can be used. A ConnectionInfo string is a '|' delimited list of SDE connection properties. The following properties **must** be used: server, instance, user, password, featuredatasetname and datasetname. The field featuredatasetname may be left empty.

This method generates an error when the datasource in not valid, does not exist or has no metadata.

OpenMxd Method

Opens a mxd file.

Syntax
variable = *object*.**OpenMxd (***MxdFileName* **)**

The **OpenMxd** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **DpDispatch** object.
variable	A reference to a MxdFile object.
MxdFileName	Required. The name of the Mxd.

Remarks
After opening a mxdfile you can access the dataframes and layers in the mxd file.

ReleaseDate Property

Returns the current DataProcessing release date.

Read only (Includes Get_ReleaseDate)

Syntax
variable = *object*.**ReleaseDate**

The *object* placeholder represents a **DpDispatch** object.

Return value
String

Version Property

Returns the current DataProcessing version.

Read only (Includes Get_Version)

Syntax
variable = *object*.**Version**

The *object* placeholder represents a **DpDispatch** object.

Return value
String

6.2 Metadata Object

Provides access to members that control the metadata of a dataset or datafile.

Members

	Description
Clear	Clears the metadata.
CreateTag	Creates a new xml-tag in the metadata.
CreateThumbnail	Creates or updates the thumbnail in the metadata.
DeleteAttribute	Deletes an xml-attribute.
DeleteTag	Deletes a xml-tag.
ExportThumbnail	Exports the thumbnail in the metadata to a png file.
ExportXml	Exports the metadata to an external file.
GetAttributes	Returns the attributes of a xml-tag or shortcut.
GetAttributeValue	Returns the value of an xml-attribute.
GetValue	Returns the value of a xml-tag or shortcut.
ImportXml	Imports the metadata from an external file.
LinkIsValid	Returns if the value (url or file) is available or exists.
Matches	Returns if text in xml-tag matches wildcard.
SetAttributes	Sets the attributes of a xml-tag or shortcut.
SetAttributeValue	Sets the value of an xml-attribute.
SetValue	Sets the value of a xml-tag or shortcut.
SubTagIndices	Returns information about repeating xml-subtags.
Synchronize	Synchronizes the metadata with the data.
TagCount	Returns the number of occurrences of a xml-tag.
TagExists	Returns if a xml-tag or shortcut exists.
Xml	Returns the metadata as Xml.

Clear Method

Clears the metadata of the data.

Syntax
object.**Clear (** [*InitalXml*] **)**

The **Clear** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **Metadata** object.
InitialXml	Optional. The initial xml of the created metadata.

Remarks
Existing metadata is replaced by an empty metadata xml structure.

The InitialXml string must be valid xml. In no InitialXml string is specified the metadata xml structure wil be initialized with the following xml: *<?xml version="1.0"?><metadata></metadata>*.

CreateTag Method

Creates a new xml-tag in the metadata if the tag not already exists.

Syntax
object.**CreateTag (** *TagOrShortcut, [Attributes]* **)**

The **CreateTag** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **Metadata** object.
TagOrShortcut	Required. The tag or shortcut.
Attributes	Optional: Attributes for the tag.

Remarks
The xml-tag must have the format <tagname>/<subtagname>/… If no root "/" is specified the default roottag "/metadata" is used.

If a shortcut is used, this shortcut must always start with a %.

All subtags of the full xml-tag are created when they do not exist.

When attributes are specified these are set for the last subtag.

When creating ISO tags using shortcuts and the ISO tag gmd:MD_Metadata is created, the proper attributes are retrieved from the shortcut template in which the shortcut is defined.

You can use shortcuts with indices to specify indexed tags. These indices can reflect more than one level. For example the shortcut %PocName[1][0] will be translated into
point_of_contacts_role/point_of_contact[1]/organisation_name[0]

Note: The first subtag has index 0.

When using indices new tags will be added or inserted. When indices are used for the last subtag, the new subtag is added when the index is greater than the number of existing subtags. Otherwise the sew subtag is inserted according to the index value.
When indices are used for intermediate subtags, new subtags are only added if they do not already exist.
If an index is greater than the number of existing subtags, additional subtags are created to match the index.

If no indices are specified the new tag will be added as the last subtag.

For example, suppose the following xml (schematic):

```
<metadata>
  <a>
    <b>
      <c>
      </c>
    </b>
</metadata>
```

The following calls can be made:

```
CreateTag('/a/b/c/d','title="d"')
CreateTag('/a/b/c/e','title="e1"')
CreateTag('/a/b/c/e[0]','title="e2"')
CreateTag'/a/b/c/e[3]','title="e3"')
CreateTag'/a/b/c[2]/f','title="f1"')
CreateTag'/a/b/c[0]/f[1]','title="f2"')
CreateTag'/a/b/d/g','title="g1"')
```

This gives the following result:

```
<metadata>
  <a>
    <b>
      <c>
```

```
        <d title=="d">
        </d>
        <e title=="e2">
        </e>
        <e title=="e1">
        </e>
        <e />
        <e title=="e3">
        </e>
        <f />
        <f title=="f2">
        </f>
      </c>
      <c />
      <c>
        <f title=="f1">
        </f>
      </c>
      <d>
        <g title=="g1">
        </g>
      </d>
    </b>
</metadata>
```

This method cannot be used using **webservices**.

CreateThumbnail Method

Creates or updates the thumbnail in the metadata.

Syntax
object.**CreateThumbnail (** *CenterX, CenterY, Scale* **)**

The **CreateThumbnail** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **Metadata** object.
CenterX	Required. A Double that represents the x-coordinate of the center of the maps extent.
CenterY	Required. A Double that represents the y-coordinate of the center of the maps extent.
Scale	Required. A Double that represents the mapscale.

Remarks
This method will create a thumbnail for the metadata datasource. The CenterX, CenterY and Scale settings can be used to zoom to a point using the specified Scale. When Scale is negative the CenterX and CenterY will be ignored and no zooming will be taken place.

In order to create a thumbnail metadata should already been created.

This method can only be used using **geo datasources** and cannot be used using **webservices**.

DeleteAttribute Method

Deletes an existing xml-attribute in a xml-tag from the metadata.

Syntax

object.**DeleteAttribute (*TagOrShortcut, AttributeName*)**

The **DeleteAttribute** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **Metadata** object.
TagOrShortcut	Required. The tag or shortcut.
AttributeName	Required. The name of the attribute.

Remarks
The xml-tag must have the format <tagname>/<subtagname>/…

If a shortcut is used, this shortcut must always start with a %.

You can use shortcuts with indices to specify indexed tags. These indices can reflect more than one level. For example the shortcut %PocName[1][0] will be translated into
point_of_contacts_role/point_of_contact[1]/organisation_name[0]

The attributename is case-sensitive.

This method can not be used using **webservices**.

DeleteTag Method

Deletes an existing xml-tag from the metadata.

Syntax
object.**DeleteTag (*TagOrShortcut*)**

The **DeleteTag** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **Metadata** object.
TagOrShortcut	Required. The tag or shortcut.

Remarks
The xml-tag must have the format <tagname>/<subtagname>/…

If a shortcut is used, this shortcut must always start with a %.

You can use shortcuts with indices to specify indexed tags. These indices can reflect more than one level. For example the shortcut %PocName[1][0] will be translated into
point_of_contacts_role/point_of_contact[1]/organisation_name[0]

This method can not be used using **webservices**.

ExportThumbnail Method

Exports the thumbnail in the metadata to a png file.

Syntax
object.**ExportThumbnail (*FileName, Replace*)**

The **ExportThumbnail** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **Metadata** object.
FileName	Required. The name of the png file.

Replace	Required. A Boolean that indicates if an existing png file will be replaced.

ExportXml Method

Exports the metadata to an external xml file.

Syntax
object.**ExportXml (*FileName, Replace, [TagOrShortcut]*)**

The **ExportXml** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **Metadata** object.
FileName	Required. The name of the xml file.
Replace	Required. A Boolean that indicates if an existing xml file will be replaced.
TagOrShortcut	Optional. The tag or shortcut that will be selected to be the root element of the xml file.

Remarks
The xml-tag (TagOrShortcut) must have the format <tagname>/<subtagname>/…

If a shortcut is used, this shortcut must always start with a %.

You can use shortcuts with indices to specify indexed tags. These indices can reflect more than one level. For example the shortcut %PocName[1][0] will be translated into
point_of_contacts_role/point_of_contact[1]/organisation_name[0]

The tag in TagOrShortcut must exist in the metadata of the object. If in doubt create an export file first without the optional TagOrShortcut to check the file for possible valid tags.

The GetAttributes Method

Returns the attributes of a xml-tag or shortcut.

Syntax
object.**GetAttributes (*TagOrShortcut*)**

The **GetAttributes** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **Metadata** object.
TagOrShortcut	Required. The tag or shortcut.

Return value
String

Remarks
The xml-tag must have the format <tagname>/<subtagname>/…

If a shortcut is used, this shortcut must always start with a %.

You can use shortcuts with indices to specify indexed tags. These indices can reflect more than one level. For example the shortcut %PocName[1][0] will be translated into
point_of_contacts_role/point_of_contact[1]/organisation_name[0]

Returns an empty string if the tag has no attributes.

GetAttributeValue Method

Returns the value of an attribute in a xml-tag or shortcut.

Syntax
object.**GetAttributeValue (***TagOrShortcut, AttributeName* **)**

The **GetAttributeValue** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **Metadata** object.
TagOrShortcut	Required. The tag or shortcut.
AttributeName	Required. The name of the attribute.

Return value
String

Remarks
The xml-tag must have the format <tagname>/<subtagname>/…

If a shortcut is used, this shortcut must always start with a %.

You can use shortcuts with indices to specify indexed tags. These indices can reflect more than one level. For example the shortcut %PocName[1][0] will be translated into
point_of_contacts_role/point_of_contact[1]/organisation_name[0]

The attributename is case-sensitive.

Throws an exception if the attribute does not exist.

GetValue Method

Returns the value of a xml-tag or shortcut.

Syntax
object.**GetValue (***TagOrShortcut* **)**

The **GetValue** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **Metadata** object.
TagOrShortcut	Required. The tag or shortcut.

Return value
String

Remarks
The xml-tag must have the format <tagname>/<subtagname>/…

If a shortcut is used, this shortcut must always start with a %.

You can use shortcuts with indices to specify indexed tags. These indices can reflect more than one level. For example the shortcut %PocName[1][0] will be translated into
point_of_contacts_role/point_of_contact[1]/organisation_name[0]

ImportXml Method

Imports the metadata from an external xml file.

Syntax
object.**ImportXml (*FileName*)**

The **ImportXml** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **Metadata** object.
FileName	Required. The name of the xml file.

Remarks
This method can not be used using **webservices**.
The external xml file will be imported as it is. No changes will be made to its content.

LinkIsValid Method

Returns True if the value (url or file) of the xml-tag or shortcut is available or exists.

Syntax
object.**LinkIsValid (*TagOrShortcut*)**

The **LinkIsValid** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **Metadata** object.
TagOrShortcut	Required. The xml-tag of shortcut.

Return value
Boolean

Remarks
The xml-tag must have the format <tagname>/<subtagname>/…

If a shortcut is used, this shortcut must always start with a %.

You can use shortcuts with indices to specify indexed tags. These indices can reflect more than one level. For example the shortcut %PocName[1][0] will be translated into
point_of_contacts_role/point_of_contact[1]/organisation_name[0]

Matches Method

Returns if the text in the xml-tag or shortcut matches the wildcard.

Syntax
object.**Matches (*TagOrShortcut, Wildcard*)**

The **Matches** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **Metadata** object.
TagOrShortcut	Required. The xml-tag of shortcut to be searched through.
Wildcard	Required. The wildcard text.

Return value
Boolean

Remarks
The xml-tag must have the format <tagname>/<subtagname>/…

If a shortcut is used, this shortcut must always start with a %.

You can use shortcuts with indices to specify indexed tags. These indices can reflect more than one level. For example the shortcut %PocName[1][0] will be translated into
point_of_contacts_role/point_of_contact[1]/organisation_name[0]

Returns True if the text of the xml-tag matches the specified wildcard text. You can use a '*' and a '?' as wildcard symbol.

SetAttributes Method

Sets the attributes of a xml-tag of shortcut.

Syntax
object.**SetAttributes (***TagOrShortcut, Attributes* **)**

The **SetAttributes** method syntax has the following object qualifier and arguments:

Part	Description
Object	An object placeholder that evaluates to a **Metadata** object.
TagOrShortcut	Required. The xml-tag of shortcut.
Attributes	Required. The attributes to be set.

Remarks
The attributes mus be in the format 'name1="value1" name2="value2"'.
For example:

```
'gco:nilReason="withheld"'
```

The xml-tag must have the format <tagname>/<subtagname>/… If no root "/" is specified the default roottag "/metadata" is used.

If a shortcut is used, this shortcut must always start with a %.

You can use shortcuts with indices to specify indexed tags. These indices can reflect more than one level. For example the shortcut %PocName[1][0] will be translated into
point_of_contacts_role/point_of_contact[1]/organisation_name[0]

This method can not be used using **webservices**.

SetAttributeValue Method

Sets the value of an attribute in a xml-tag of shortcut.

Syntax
object.**SetAttributeValue (***TagOrShortcut, Attributes, Value* **)**

The **SetAttributeValue** method syntax has the following object qualifier and arguments:

Part	Description
Object	An object placeholder that evaluates to a **Metadata** object.
TagOrShortcut	Required. The xml-tag of shortcut.
AttributeName	Required. The name of the attribute.
Value	Required. The text to be set.

Remarks
The xml-tag must have the format <tagname>/<subtagname>/… If no root "/" is specified the default roottag "/metadata" is used.

If a shortcut is used, this shortcut must always start with a %.

You can use shortcuts with indices to specify indexed tags. These indices can reflect more than one level. For example the shortcut %PocName[1][0] will be translated into
point_of_contacts_role/point_of_contact[1]/organisation_name[0]

The attributename is case-sensitive.

The xml-attribute will be created if it does not exist.

This method can not be used using **webservices**.

SetValue Method

Sets the value of a xml-tag of shortcut.

Syntax
object.**SetValue (***TagOrShortcut, Value* **)**

The **SetValue** method syntax has the following object qualifier and arguments:

Part	Description
Object	An object placeholder that evaluates to a **Metadata** object.
TagOrShortcut	Required. The xml-tag of shortcut.
Value	Required. The text to be set.

Remarks
The xml-tag must have the format <tagname>/<subtagname>/… If no root "/" is specified the default roottag "/metadata" is used.

If a shortcut is used, this shortcut must always start with a %.

The xml-tag will be created if it does not exist, including all subtags of the full xml-tag if they do not exist.

When setting the value of a ISO tag using shortcuts and the ISO tag gmd:MD_Metadata is created, the proper attributes are retrieved from the shortcut template in which the shortcut is defined.

You can use shortcuts with indices to specify indexed tags. These indices can reflect more than one level. For example the shortcut %PocName[1][0] will be translated into
point_of_contacts_role/point_of_contact[1]/organisation_name[0]

This method can not be used using **webservices**.

SubTagIndices Method

Returns information about repeating xml-subtags.

Syntax
object.**SubTagIndices (** *TagOrShortcut* **)**

The **SubTagIndices** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **Metadata** object.
TagOrShortcut	Required. The tag or shortcut.

Return value
String

Remarks
The result of this method shows if subtags of a xml-tag exists and if they occur more than ones within their parent tags.

For example, suppose the following xml (schematic):

```
<metadata>
  <a>
    <b>
      <c>
      </c>
    </b>
    <b>
      <c>
      </c>
    </b>
  </a>
</metadata>
```

Then the call `SubTagIndices("a/b/c/d")` returns the string `[0][%d][0][]` in which means

[0]	this subtag occurs at most 1 time within it parent.
[%d]	this subtag occurs more than ones within it parent.
[]	this subtag does not exist.

In this example the result shows that subtag ‘a’ occurs at most 1 time, subtag ‘b’ is repeating, subtag ‘c’ occurs also at most 1 time within its parent and subtag ‘d’ does not exist.

Note: The root tag will **never** be shown in the result string.

So, `SubTagIndices("/metadata/a/b/c/d")` will give the same result string `[0][%d][0][]`.

The xml-tag must have the format <tagname>/<subtagname>/… and indices are not allowed.

If a shortcut is used, this shortcut must always start with a %.

Synchronize Method

Synchronizes the metadata with the data.

Syntax
object.**Synchronize**

The *object* placeholder represents a **Metadata** object.

Remarks
The Synchronize method can only be used using **geo datasources** and cannot be used using **webservices**.

TagCount Method

Returns the number of occurrences of a xml-tag.

Syntax
object.**TagCount (** *TagOrShortcut* **)**

The **TagCount** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **Metadata** object.
TagOrShortcut	Required. The tag or shortcut.

Return value
Integer

Remarks
This method returns the number of occurrences of a xml-tag. For example, suppose the following xml (schematic):

```
<metadata>
  <a>
    <b>
      <c>
      </c>
    </b>
    <b>
      <c>
      </c>
    </b>
  </a>
</metadata>
```

TagCount gives the following result:

TagCount("a")	1
TagCount("a/b")	2
TagCount("a/b/c")	2
TagCount("/metadata/a/b/c")	2
TagCount("a/b/c/d")	0

The xml-tag must have the format <tagname>/<subtagname>/… and indices are not allowed.

If a shortcut is used, this shortcut must always start with a %.

TagExists Method

Returns if the xml-tag or shortcut exists.

Syntax
object.**TagExists (** *TagOrShortcut* **)**

The **TagExists** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **Metadata** object.
TagOrShortcut	Required. The tag or shortcut.

Return value
Boolean

Remarks
If a shortcut is used, this shortcut must always start with a %.

Xml Property

Returns the metadata as Xml.

Read only (Includes Get_Xml)

Syntax

variable = *object*.**Xml**

The *object* placeholder represents a **Metadata** object.

Return value
String

6.3 MxdFile Object

Provides access to members that control a mxdfile.

Members

	Description
CreateDataFrame	Creates a new dataframe in the mxdfile.
DeleteDataFrame	Deletes a dataframe from the mxdfile.
ListDataFrames	Returns all dataframes in the mxdfile.
NrOfDataFrames	Returns the number of dataframes in the mxdfile.
OpenDataFrame	Opens a dataframe.

CreateDataFrame Method

Creates a new dataframe in the mxdfile.

Syntax
variable = *object*.**CreateDataFrame (*Name, Index*)**

The **CreateDataFrame** method syntax has the following object qualifier and arguments:

Part	Description
Object	An object placeholder that evaluates to a **MxdFile** object.
variable	A reference to a DataFrame object.
Name	Required. The name of the new dataframe.
Index	Required. A Integer that represents the place where the new dataframe is inserted.

Remarks
The first dataframe in a MxdFile has index 0.

DeleteDataFrame Method

Deletes a dataframe in the mxdfile.

Syntax
object.**DeleteDataFrame (*Index*)**

The **DeleteDataFrame** method syntax has the following object qualifier and arguments:

Part	Description
Object	An object placeholder that evaluates to a **MxdFile** object.
Index	Required. A Integer that represents the index of the dataframe to be deleted.

Remarks
The first dataframe in a MxdFile has index 0.

ListDataFrames Method

Returns a list with the names of all dataframes in the mxdfile.

Syntax
object.**ListDataFrames**

The *object* placeholder represents a **MxdFile** object.

Return value
String

Remarks
The names of the dataframes in the list are separated by a '|'.

NrOfDataFrames Property

The number of dataframes in the mxdfile.

Read only (Includes Get_NrOfDataFrames)

Syntax
variable = *object*.**NrOfDataFrames**

The *object* placeholder represents a **MxdFile** object.

Return value
Integer

OpenDataFrame Method

Opens a dataframe in the mxdfile.

Syntax
variable = *object*.**OpenDataFrame (** *Index* **)**

The **OpenDataFrame** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **MxdFile** object.
variable	A reference to a DataFrame object.
Index	Required. A Integer that represents the index of the dataframe to be opened.

Remarks

The first dataframe in a MxdFile has index 0.

6.4 DataFrame Object

Provides access to members that control a dataframe which is a reference to a map in ArcMap.

Members

	Description
CreateLayer	Creates a new layer in the dataframe.
DeleteLayer	Deletes a layer.
Index	Returns or sets the index of the dataframe.
ListLayers	Returns the layers in the dataframe.
Name	Returns or sets the name of the dataframe.
NrOfLayers	Returns the number of layers in the dataframe.
OpenLayer	Opens a layer.

CreateLayer Method

Creates a new layer in the dataframe.

Syntax
variable = *object*.**CreateLayer (***Name, DataSource, Index***)**

The **CreateLayer** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **DataFrame** object.
variable	A reference to a Layer object.
Name	Required. The name of the Layer.
DataSource	Required. The name of the DataSource.
Index	Required. A Integer that represents the index of the Layer.

Remarks
The DataSource must be a valid ArcCatalogPath. When the DataSource is an empty string a **grouplayer** will be created,

The Index is the place at which the new Layer is to be inserted. The first Layer in a DataFrame has index 0.

DeleteLayer Method

Deletes a layer in the dataframe.

Syntax
object.**DeleteLayer (***Index***)**

The **DeleteLayer** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **DataFrame** object.
Index	Required. A Integer that represents the index of the Layer.

Remarks
The first Layer in a DataFrame has index 0.

Index Property

The index of the datasource.

 Read/Write (Includes Get_Index/Set_Index)

Syntax
object.**Index** = [*value*]
variable = *object*.**Index**

Part	Description
object	An object expression that evaluates to a **DataFrame** object.
value	A Integer that determines the Index.

Return value
Integer

Remarks
The first Layer in a DataFrame has index 0.

When setting the Index of a DataFrame the DataFrame will be moved to the specified place.

ListLayers Method

Returns a list with the names of all layers in the dataframe.

Syntax
object.**ListLayers**

The *object* placeholder represents a **DataFrame** object.

Return value
String

Remarks
The names of the layers in the list are separated by a '|'. No sublayers are reported.

Name Property

The name of the dataframe.

 Read/Write (Includes Get_Name/Set_Name)

Syntax
object.**Name** = [*value*]
variable = *object*.**Name**

Part	Description
object	An object expression that evaluates to a **DataFrame** object.
value	A String that determines the Name.

Return value
String

NrOfLayers Property

The number of layers in the dataframe.

Read only (Includes Get_NrOfLayers)

Syntax
variable = *object*.**NrOfLayers**

The *object* placeholder represents a **DataFrame** object.

Return value
Integer

OpenLayer Method

Opens a layer in the dataframe.

Syntax
variable = *object*.**OpenLayer (*Index*)**

The **OpenLayer** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **DataFrame** object.
variable	A reference to a Layer object.
Index	Required. A Integer that represents the index of the layer to be opened.

Remarks
The first layer in a dataframe has index 0.

6.5 Layer Object

Provides access to members that control a Layer.

There are two kinds of layers: DataLayers and GroupLayers. A DataLayer is a layer which represents a DataSource. A GroupLayer has no associated DataSource but is a layer which can contain other layers, so called SubLayers. A SubLayer can be a DataLayer of a GroupLayer.

Members

	Description
ConnectionInfo	Returns or sets the connection information of the datasource of the layer.
CreateSubLayer	Creates a new sublayer in a grouplayer.
DataSource	Returns or sets the datasource of the layer.
DataType	Returns the type of datasource of the layer.
DeleteSubLayer	Deletes a sublayer from a grouplayer.
Index	Returns or sets the index of the layer.
ImportSymbology	Imports the classification and symbology from a layer in a layerfile.
IsGroupLayer	Returns if the layer is a grouplayer.
IsValid	Returns if the datasource in the layer is valid.
ListSubLayers	Returns the sublayers of a grouplayer.
MaxScale	Returns or sets the maxscale of the layer.
MinScale	Returns or sets the minscale of the layer.
Name	Returns or sets the name of the layer.
NrOfSubLayers	Returns the number of sublayers of a grouplayer.

OpenSubLayer	Opens a sublayer.
SaveAs	Saves a layer to a layerfile.
Visible	Returns or sets if the layer is visible.
WhereClause	Returns or sets the whereclause of the layer.

ConnectionInfo Property

The Connection Information of the datasource of a layer.

Read/Write (Includes Get_ConnectionInfo/Set_ConnectionInfo)

Syntax
object.**ConnectionInfo** = [*value*]
variable = *object*.**ConnectionInfo**

Part	Description
object	An object expression that evaluates to a **Layer** object.
value	A String that determines the Connection Information.

Return value
String

Remarks
The ConnectionInfo represents the connection properties of a datasource of a layer.

The ConnectionInfo string is a '|' delimited list of connection properties.

ConnectionInfo gets or sets the connection properties of the following datasources:
- SDE datasources

When retrieving the ConnectionInfo the following properties are returned: server, instance, user, password, featuredatasetname and datasetname. The returned password field is always empty and if the dataset is not a member of a featuredataset this field is empty too.

An empty string is returned when used on a layer with no supported datasource.

When setting the ConnectionInfo the following properties **must** be used: server, instance, user, password, featuredatasetname and datasetname. The field featuredatasetname may be left empty.

CreateSubLayer Method

Creates a new sublayer in a grouplayer.

Syntax
variable = *object*.**CreateSubLayer (*Name, DataSource, Index*)**

The **CreateSubLayer** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **Layer** object.
variable	A reference to a Layer object.
Name	Required. The name of the Layer.
DataSource	Required. The name of the DataSource.
Index	Required. A Integer that represents the index of the Layer.

Remarks
To use this method the current layer must be a **grouplayer**.

The DataSource must be a valid ArcCatalogPath. When the DataSource is an empty string a **grouplayer** will be created,

The Index is the place at which the new sublayer is to be inserted. The first sublayer in a grouplayer has index 0.

DataSource Property

The dataSource of the layer.

 Read/Write (Includes Get_DataSource/Set_DataSource)

Syntax
object.**DataSource** = [*value*]
variable = *object*.**DataSource**

Part	Description
object	An object expression that evaluates to a **Layer** object.
value	A String that determines the DataSource.

Return value
String

Remarks
The DataSource represents a valid ArcCatalogPath.

DataType Property

The DataType of the layer.

 Read only (Includes Get_DataType)

Syntax
variable = *object*.**DataType**

The *object* placeholder represents a **Layer** object.

Return value
String

Remarks
Returned datatype strings are:
- Arc Feature Class
- Shapefile Feature Class
- Raster Dataset
- Raster Catalog
- Personal Geodatabase Feature Class
- Personal Geodatabase Raster Dataset
- Personal Geodatabase Raster Catalog
- File Geodatabase Feature Class
- File Geodatabase Raster Dataset
- SDE Feature Class
- SDE Raster
- SDE Raster Catalog
- ArcIMS Image Service
- ArcIMS Feature Class
- ArcIMS Feature Class Group

- XY Event Source

DeleteSubLayer Method

Deletes a sublayer of a grouplayer.

Syntax
object.**DeleteSubLayer (** *Index* **)**

The **DeleteSubLayer** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **Layer** object.
Index	Required. A Integer that represents the index of the sublayer.

Remarks
To use this method the current layer must be a grouplayer.

The first sublayer in a grouplayer has index 0.

ImportSymbology Method

Imports the classification and symbology from a layer in a layerfile.

Syntax
object.**ImportSymbology (** *LayerFileName, [LayerName]***)**

The **ImportSymbology** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **Layer** object.
LayerFileName	Required. The name of the layerfile.
LayerName	Optional. If specified, the name of the layer to copy from, if omitted the first layer will be used.

Remarks
Classification and symbology can only be copied when both layers are from the same type: vector or raster.

Index Property

The index of the layer in a dataframe of grouplayer.

Read/Write (Includes Get_Index/Set_Index)

Syntax
object.**Index** = [*value*]
variable = *object*.**Index**

Part	Description
object	An object expression that evaluates to a **Layer** object.
value	A Integer that determines the Index.

Return value
Integer

Remarks
The first layer in a dataframe or sublayer in a grouplayer has index 0.

When setting the Index of a Layer the Layer will be moved to the specified place.

The Index of the base layer in a LayerFile cannot be set.

IsGroupLayer Method

Returns whether the layer is a grouplayer.

Syntax
object.**IsGroupLayer**

The *object* placeholder represents a **Layer** object.

Return value

Boolean

IsValid Method

Returns whether the datasource of the layer is available or exists.

Syntax
object.**IsValid**

The *object* placeholder represents a **Layer** object.

Return value
Boolean

ListSubLayers Method

Returns a list with the names of all sublayers in the grouplayer.

Syntax
object.**ListSubLayers**

The *object* placeholder represents a **Layer** object.

Return value
String

Remarks
To use this method the current layer must be a **grouplayer**.

The names of the sublayers in the list are separated by a '|'.

MaxScale Property

The maximum scale (representative fraction) at which the layer will display.

 Read/Write (Includes Get_MaxScale/Set_MaxScale)

Syntax
object.**MaxScale** = [*value*]
variable = *object*.**MaxScale**

Part	Description
object	An object expression that evaluates to a **Layer** object.
value	A Double that determines the MaxScale.

Return value
Double

Remarks
Specifies the maximum scale at which the layer will be displayed. This means that if you zoom in beyond this scale, the layer will not display. For example, specify 500 to have the layer not display when zoomed in beyond 1:500.

MinScale Property

The minimum scale (representative fraction) at which the layer will display.

 Read/Write (Includes Get_MinScale/Set_MinScale)

Syntax
object.**MinScale** = [*value*]
variable = *object*.**MinScale**

Part	Description
object	An object expression that evaluates to a **Layer** object.
value	A Double that determines the MinScale.

Return value
Double

Remarks
Specifies the minimum scale at which the layer will be displayed. This means that if you zoom out beyond this scale, the layer will not display. For example, specify 1000 to have the layer not display when zoomed out beyond 1:1000.

Name Property

The name of the layer.

 Read/Write (Includes Get_Name/Set_Name)

Syntax
object.**Name** = [*value*]
variable = *object*.**Name**

Part	Description
object	An object expression that evaluates to a **Layer** object.
value	A String that determines the Name.

Return value
String

NrOfSubLayers Property

The number of sublayers of a grouplayer.

 Read only (Includes Get_NrOfSubLayers)

Syntax
variable = *object*.**NrOfSubLayers**

The *object* placeholder represents a **Layer** object.

Return value
Integer

Remarks
To use this method the current layer must be a **grouplayer**.

OpenSubLayer Method

Opens a sublayer of a grouplayer.

Syntax
variable = *object*.**OpenSubLayer (***Index***)**

The **OpenSubLayer** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **Layer** object.
variable	A reference to a Layer object.
Index	Required. A Integer that represents the index of the sublayer to be opened.

Remarks
To use this method the current layer must be a **grouplayer**.

The first sublayer in a grouplayer has index 0.

SaveAs Method

Saves a layer to a layerfile.

Syntax
variable = *object*.**SaveAs (***LayerFileName***)**

The **SaveAs** method syntax has the following object qualifier and arguments:

Part	Description
object	An object placeholder that evaluates to a **Layer** object.
variable	A reference to a Layer object.
LayerFileName	Required. The filename of the LayerFile.

Return value
A referente to the layer in the new created layerfile.

Remarks
The layerfile should not already exist.

Visible Property

Indicates if the layer is visible.

 Read/Write (Includes Get_Visible/Set_Visible)

Syntax
object.**Visible** = [*value*]
variable = *object*.**Visible**

Part	Description
object	An object expression that evaluates to a **Layer** object.
value	A Boolean that determines the Visible state.

Return value
Boolean

WhereClause Property

The definition query expression for the layer.

 Read/Write (Includes Get_WhereClause/Set_WhereClause)

Syntax
object.**WhereClause** = [*value*]
variable = *object*.**WhereClause**

Part	Description
Object	An object expression that evaluates to a **Layer** object.
Value	A String that determines the WhereClause.

Return value
String

Remarks
Use the WhereClause property to read or set the definition query for an existing layer just like you would in the Definition Query tab of the layer's properties dialog.

7 Version History

7.1 Version 4.4

In version 4.4.1 the following modifications are made:
- This version only supports ArcGIS 10.1
- The method Metadata.TagExists is added for checking if a tag exists.
- The method Metadata.SetAttributeValue is changed to support attributes with prefixes.
- The method Metadata.SetAttributes is changed to support attributes with prefixes.
- The method Metadata.CreateTag is changed to support inserting tags using indices. CreateTag will now always create a tag.
- Minor bugfixes.
- Minor changes in the user's manual.

In version 4.4.2 the following modifications are made:
- The tool is upgraded for ArcGIS 10.2 and 10.2.1
- This version only supports ArcGIS 10.2 and 10.2.1

7.2 Version 4.5

Version 4.5.1, 4.5.2, 4.5.3, 4.5.4 and 4.5.5 are identical except for the support of the ArcGIS version.
- Version 4.5.1 supports ArcGIS 10.1 SP1.
- Version 4.5.2 supports ArcGIS 10.2.x
- Version 4.5.3 supports ArcGIS 10.3.x
- Version 4.5.4 supports ArcGIS 10.4.x
- Version 4.5.5 supports ArcGIS 10.5.x

In the above versions the following modification is made:
- The method Metadata.ExportXml now accepts an optional parameter for specifying a root tag.

7.3 Version 4.6

Version 4.6.2, 4.6.3, 4.6.4, 4.6.5, 4.6.6 and 4.6.7 are identical except for the support of the ArcGIS version.
- Version 4.6.2 supports ArcGIS 10.2.x
- Version 4.6.3 supports ArcGIS 10.3.x
- Version 4.6.4 supports ArcGIS 10.4.x
- Version 4.6.5 supports ArcGIS 10.5.x
- Version 4.6.6 supports ArcGIS 10.6.x
- Version 4.6.7 supports ArcGIS 10.7 / 10.7.1

In the above versions the following modification is made:
- Warning during install when the required pywin32 package / win32com.client is not installed.

Appendix A. License Agreement

ARIS Software License Agreement for ARIS DataProcessing Tool for ArcGIS

This is a license agreement and not an agreement for sale. This license agreement (hereinafter referred to as AGREEMENT) is between the end user (hereinafter referred to as LICENSEE) and ARIS b.v., The Netherlands (hereinafter referred to as ARIS), and gives the LICENSEE certain limited rights to use the proprietary ARIS software, examples, on-line and hardcopy documentation and updates (if applicable), hereinafter referred to as PRODUCT. All rights not specifically granted in this AGREEMENT are reserved to ARIS.

Ownership and grant of license

ARIS and its third party licensor(s) retain exclusive rights, title, and ownership of the copy of the PRODUCT licensed under this AGREEMENT and hereby grant to LICENSEE a personal, non-exclusive, non-transferable license to use the PRODUCT based on the terms and conditions of this AGREEMENT. From the date of receipt, the LICENSEE shall agree to make reasonable efforts to protect the PRODUCT from unauthorized use, reproduction, distribution, or publication.

Copyright

The PRODUCT is owned by ARIS and partly by its third party licensor(s) and is protected by Dutch copyright laws and subject to international laws, treaties, and/or conventions.
The LICENSEE agrees not to export the PRODUCT into a country that does not have copyright laws that will protect ARIS's proprietary rights.

Permitted uses

The LICENSEE may use the number of copies of the PRODUCT for which license fees have been paid on computer system(s) and/or specific computer network(s) for the LICENSEE's own internal use.

The LICENSEE may install the number of copies of the PRODUCT for which license or update fees have been paid onto permanent storage device(s) on computer system(s) and/or specific computer network(s).

The LICENSEE may make one (1) copy of the PRODUCT for archival purposes only, during the term of this AGREEMENT, unless the right to make additional copies has been granted by ARIS to the LICENSEE in writing.

The LICENSEE may use parts of the documentation in other documents for LICENSEE's own internal use only with the purpose of using or encouraging to use the PRODUCT.

Uses not permitted

The LICENSEE shall not sell, rent, lease, assign, timeshare, or transfer, in whole or in part, or provide unlicensed third parties access to prior or present versions of the PRODUCT, any updates, or the LICENSEE's rights under this AGREEMENT.

The LICENSEE shall not reverse, engineer, decompile, or disassemble the PRODUCT, or make any attempt to alter the license number and other license information shown in the about box.

The LICENSEE shall not remove or obscure any ARIS copyright or trademark notices.

The LICENSEE shall not make additional copies of the PRODUCT beyond what is laid down in the "permitted uses" section of this AGREEMENT.

Term

The license granted by this AGREEMENT shall commence upon LICENSEE's receipt of the PRODUCT and shall continue until such time as:· the LICENSEE elects to discontinue the use of the PRODUCT;
· ARIS terminates the agreement due to the LICENSEE's material breach of this AGREEMENT.
Upon termination of this AGREEMENT in either instance, LICENSEE shall return to ARIS the PRODUCT and any whole or partial copies in any form. The parties hereby agree that all provisions operating to protect the rights of ARIS shall remain in force, should breach occur.

Limited Warranty

ARIS warrants that the media upon which the PRODUCT is provided will be free from defects in materials and workmanship under normal use and service for a period of ninety (90) days from the date of receipt.

Except for the above express limited warranties, the PRODUCT is provided "as is", without warranty of any kind, either express or implied, including, but not limited to, the implied warranty of merchantability and fitness for a particular purpose.

Exclusive Remedy and Limitation of Liability

During the warranty period, ARIS's entire liability and the LICENSEE's exclusive remedy shall be the return of the license fee paid for the PRODUCT that does not meet ARIS's limited warranty and that is returned to ARIS or its dealers with a copy of the LICENSEE's proof of payment.

ARIS shall not be liable for indirect, special, incidental, or consequential damages related to LICENSEE's use of the PRODUCT, even if ARIS is advised of the possibility of such damage.

Waivers

No failure or delay by ARIS in enforcing any right or remedy under this AGREEMENT shall be construed as a waiver of any future or other exercise of such right or remedy by ARIS.

Order of Precedence

Any conflict and/or inconsistency between the terms of this AGREEMENT and any purchase order, or other terms shall be resolved in favour of the terms expressed in this AGREEMENT, subject to Dutch law, unless agreed otherwise.

Governing Law

This AGREEMENT is governed by the laws of the Netherlands without references to conflict of laws principles.

Entire Agreement

The parties agree that this constitutes the sole and entire agreement of the parties as to the matter set forth herein and supersedes any previous agreements, understandings, and arrangements between the parties relating hereto and is effective, valid, and binding upon the parties.

ARIS is a registered trademark, the Netherlands.