ARIS

Users Guide

DataProcessing Tool

11 november 2009

Version 2.2.0.1

ARIS B.V., Utrecht, Netherlands, www.aris.nl
eddy.scheper@aris.nl

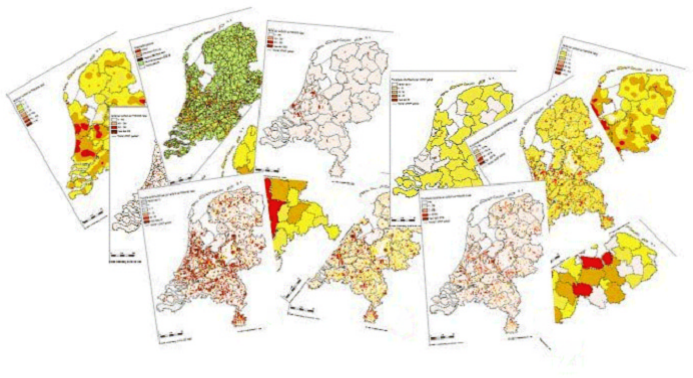
Commissioned by PBL, Bilthoven, Netherlands, www.pbl.nl

Content

1 Introduction

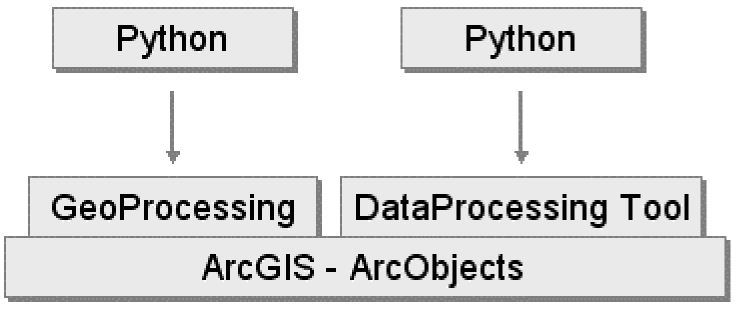
The DataProcessing Tool has been developed by ARIS for the Netherlands Environmental Assessment Agency (PBL, dutch: Planbureau voor de Leefomgeving). The goal of the tool was to provide datamanagers with the possibility to automate processes involving loading data, generating metadata and publishing 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 came from the ArcGIS GeoProcessing framework with Python as the scripting language. The ease of use of 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 DataProcessing Tool is 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 its properties. These automation methods can be called from popular languages like Python, VBScript or JScript. The 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 DataProcessing Tool in the same way as you use the standard ArcGIS GeoProcessing automation library.



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

The past two 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.

2 What's new

In the current version the following modifications are made:

- Several enhancements are added for dealing with ISO metadata.
- Better support for dealing with the Dutch GeoSticker ISO metadata format.
- The methods Metadata.CreateTag and Metadata.DeleteTag are changed; they now also work with tags with a root tag other than "/metadata".
- The method Metadata.CreateTag has an extra argument to pass attributes for the new created tag.
- The method Metadata.CreateTag is changed. Now it is possible to create multiple tags using indices.
- 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.
- The method Metadata.SetValue is changed. When the specified tag does not exists, it will be created. When using a shortcut for an ISO tag and the gmd:MD_Metadata tag does not exists, it will be created with attributes specified in de shortcut template in which the shortcut is defined.
- The methods Metadata.SetAttributes and Metedata.GetAttributes are added for setting and retrieving attributes from tags.
- The method DpDispatch.HasMetadata is changed. When a dbf-file is specified, now this method returns false when the dbf-file is part of a shapefile and returns true when it is a standalone dbf-file with a dbf.xml file.
- The method Metadata.TagCount is added for getting the number of occurrences of a specific tag.
- The method Metadata.SubTagIndices is added for getting information about repeating tags.
- The method Layer.SaveAs is added for saving layers to layerfiles.
- The method DpDispatch.CreateMxd is added for creating mxd files.
- The methods MxdFile.CreateDataFrame and MxdFile.DeleteDataFrame are added for creating and deleting dataframes in mdx files.
- The metadata shortcut files are updated.

3 License Agreement

ARIS and PBL expressly publish the DataProcessing Tool to the public domain as Freeware. This work is free for the taking and cannot be appropriated by a single author. It may be freely used and redistributed and is provided "AS IS" without warranty of any kind. When redistributed ARIS and PBL must be mentioned explicitly as authors of the DataProcessing Tool.

THIS TOOL MAY NOT BE EMBEDDED IN COMMERCIAL PRODUCTS AND YOU MAY NOT ASK PAYMENT FOR THE USE OF IT.

4 Disclaimer

Disclaimer of Warranties

ARIS AND PBL DISCLAIM ALL WARRANTIES OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE MATERIALS ARE PROVIDED "AS IS" AND YOU ASSUME ALL RISK OF USE.

Limitation of Liability

ARIS AND PBL SHALL NOT BE LIABLE TO YOU FOR COSTS OF PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOST PROFITS, LOST SALES, OR BUSINESS EXPENDITURES; INVESTMENTS; BUSINESS COMMITMENTS; LOSS OF ANY GOODWILL, OR FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES ARISING OUT OF OR RELATED TO THIS LICENSE AGREEMENT OR USE OF THE MATERIALS, HOWEVER

CAUSED ON ANY THEORY OF LIABILITY, WHETHER OR NOT ARIS AND PBL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE. THESE LIMITATIONS SHALL APPLY NOTWITHSTANDING ANY FAILURE OF ESSENTIAL PURPOSE OF ANY LIMITED REMEDY.

5 System requirements

To use the DataProcessing Tool the following software must be installed on your computer:

- Windows XP.
- Microsoft .NET Framework 2.0.
- ESRI ArcGIS 9.3 or 9.3.1, with Microsoft .NET support installed.
- Python 2.5, with win32com.client library installed.

6 Installation

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

During the installation the following files will be installed.

In the <installation> directory:

• DataProcessing.dll	*A program file.*
• DataProcessing.tlb	*A program file.*
• dataprocessing.py	*The Python wrapper class file.*
• Users Guide.pdf	*The users guide.*
• LicenseAgreement.rtf	*The license agreement.*

In the <installation>\ShortcutTemplates directory:

• MetadataKeywordsCEN3.xml	*The metadata shortcut template for CEN3.*
• MetadataKeywordsCEN4.xml	*The metadata shortcut template for CEN4.*
• MetadataKeywordsISO.xml	*The metadata shortcut template for ISO19115.*

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

7 Supported datasources

At this moment the DataProcessing Tool only supports the datasources which are frequently used by PBL. These datasources include:

- Coverages
- Shapefiles
- Grids and Images
- Raster Catalogs
- Personal Geodatabase Feature Classes
- Personal Geodatabase Grids and Images
- Personal Geodatabase Raster Catalogs
- File Geodatabase Feature Classes
- File Geodatabase Grids and Images
- SDE Feature Classes (by Oracle Direct Connect)
- SDE Grids and Images (by Oracle Direct Connect)
- SDE Raster Catalogs (by Oracle Direct Connect)
- ArcIMS Image Services
- ArcIMS Feature Services
- XY Event Sources

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

8 Getting started with scripts

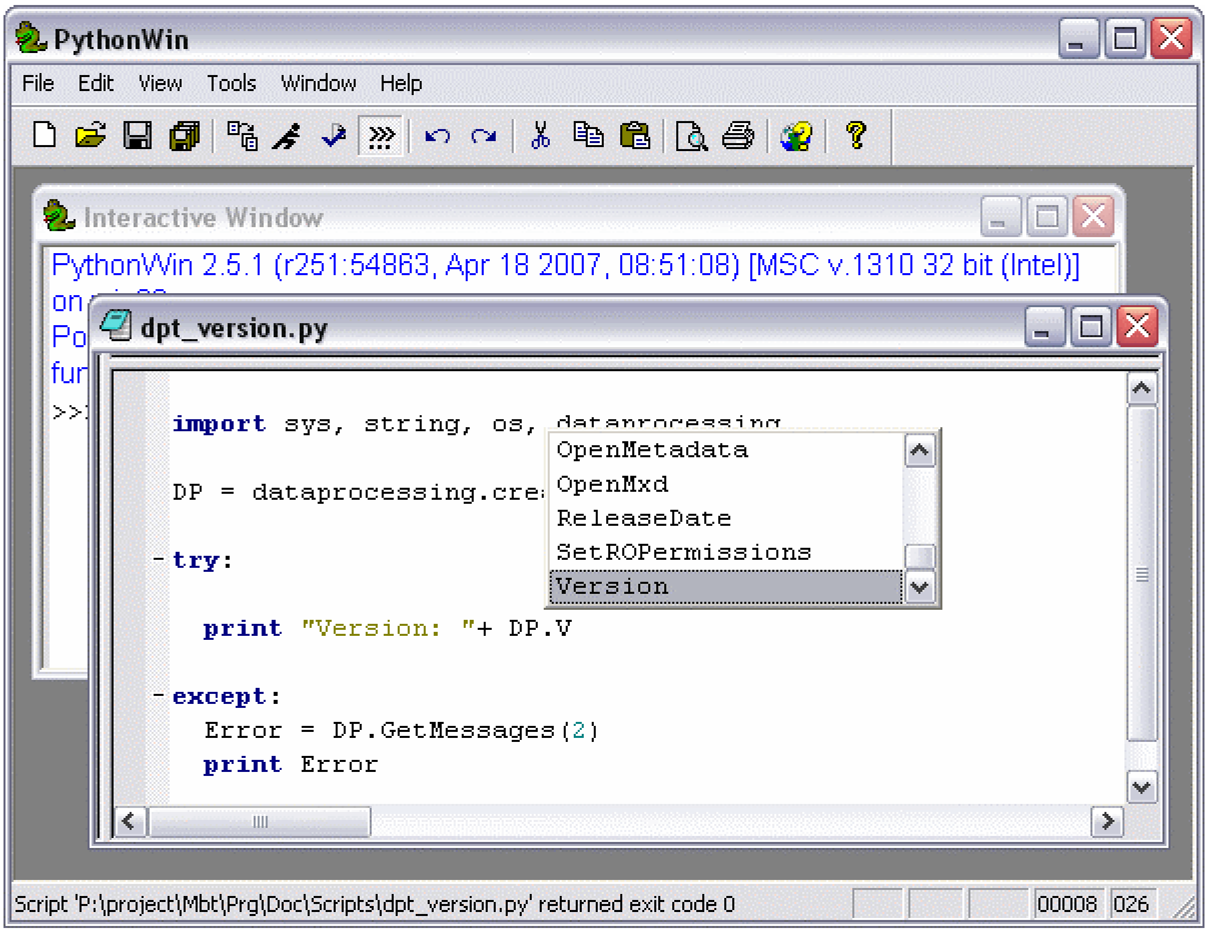
The DataProcessing Tool is designed to be used with script languages like Python, VBScript and JScript. In this Users Guide we only focus on using the DataProcessing Tool with Python.

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

This example retrieves the version of the currently installed 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
```

When using *PythonWin* for creating and editing Python scripts you can use the DataProcessing wrapper class ‘dataprocessing’ which enables inline code completion and argument information.



```
PythonWin 2.5.1 (r251:54863, Apr 18 2007, 08:51:08) [MSC v.1310 32 bit (Intel)]
>>>
```

```
import sys, string, os, dataprocessing

DP = dataprocessing.cre

try:

  print "Version: "+ DP.V

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

```
OpenMetadata
OpenMxd
ReleaseDate
SetROPermissions
Version
```

```
Script 'P:\project\Mbt\Prg\Doc\Scripts\dpt_version.py' returned exit code 0    00008 026
```

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

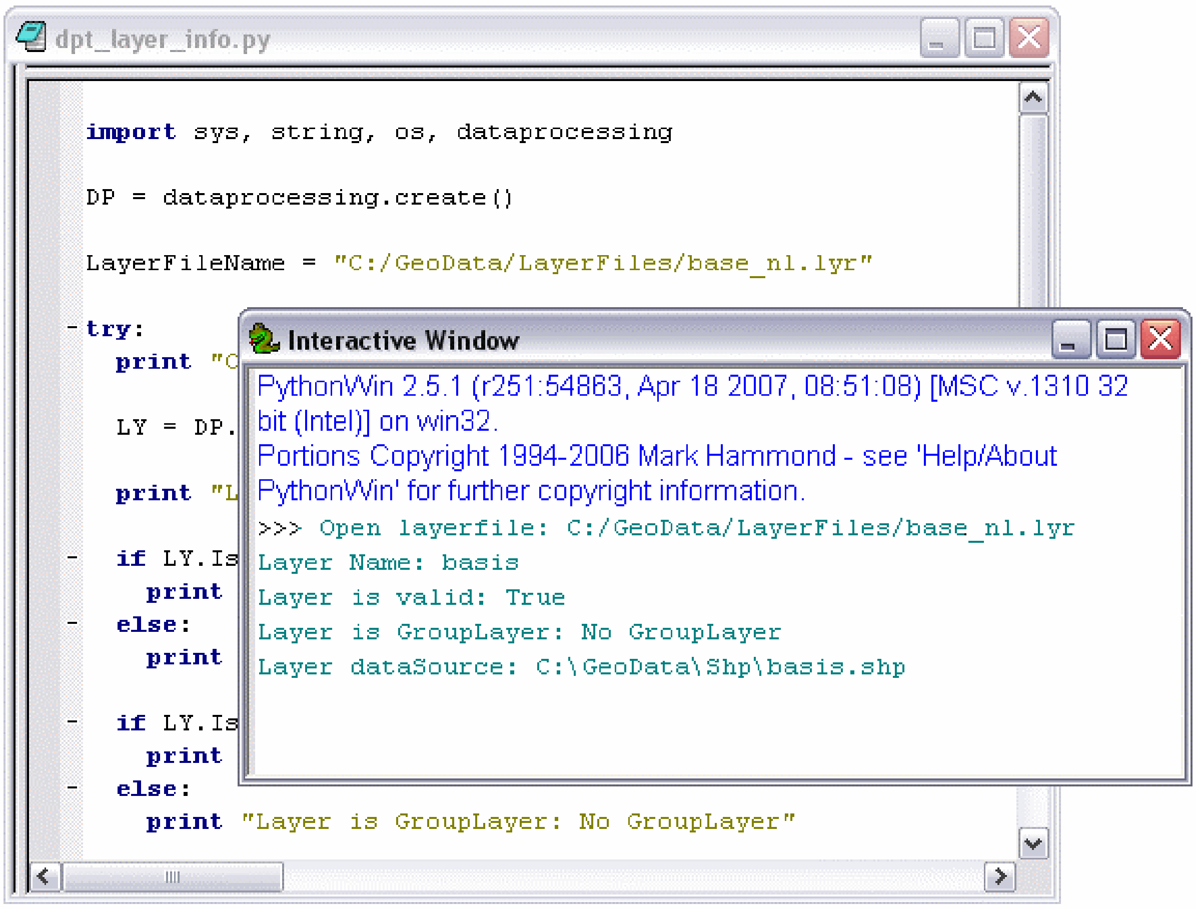
```
import sys, string, os, dataprocessing

DP = dataprocessing.create()
```

```
try:

  print "Version: "+ DP.Version

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

An example of a more practical task is shown by the next script. This script prints some information from a layerfile.

```
import sys, string, os, dataprocessing

DP = dataprocessing.create()

LayerFileName = "C:/GeoData/LayerFiles/base_nl.lyr"

try:
  print "Open layerfile: " + LayerFileName

  LY = DP.OpenLayerFile(LayerFileName)

  print "Layer Name: "+LY.Name

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

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

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

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

Running this script gives the following result:



```
import sys, string, os, dataprocessing

DP = dataprocessing.create()

LayerFileName = "C:/GeoData/LayerFiles/base_nl.lyr"

try:
  print "C
  
  LY = DP.
  
  print "L
  
  if LY.Is
    print
  else:
    print
    
  if LY.Is
    print
  else:
    print "Layer is GroupLayer: No GroupLayer"
```

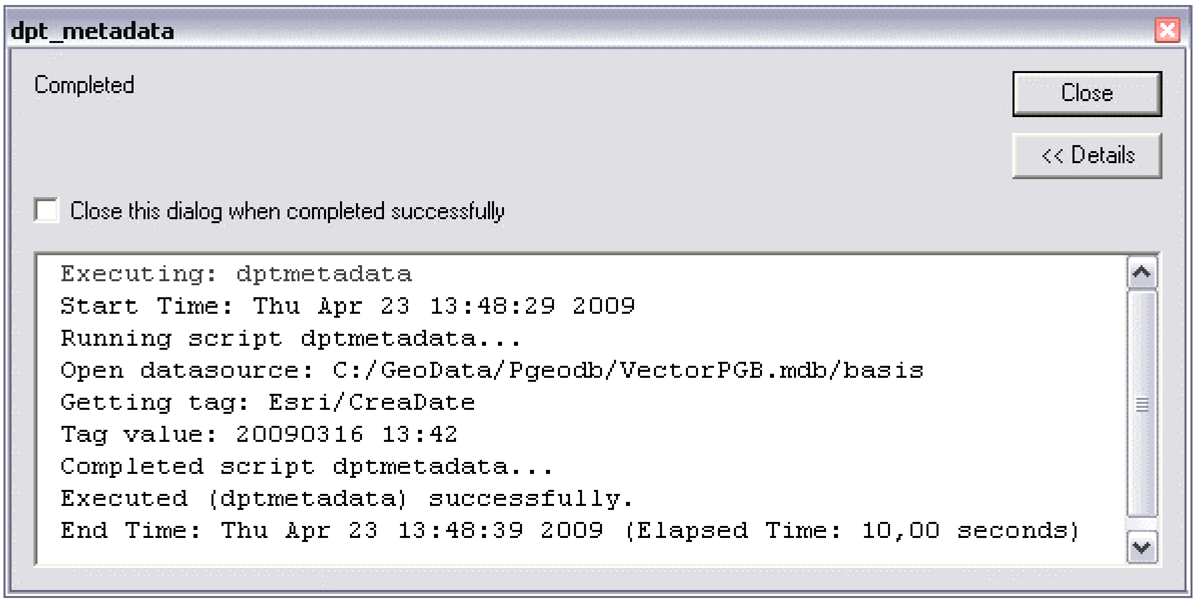
```
PythonWin 2.5.1 (r251:54863, Apr 18 2007, 08:51:08) [MSC v.1310 32
bit (Intel)] on win32.
Portions Copyright 1994-2006 Mark Hammond - see 'Help/About
PythonWin' for further copyright information.
>>> Open layerfile: C:/GeoData/LayerFiles/base_nl.lyr
Layer Name: basis
Layer is valid: True
Layer is GroupLayer: No GroupLayer
Layer dataSource: C:\GeoData\Shp\basis.shp
```

Except from running scripts from PythonWin it is possible to run scripts from within ArcGIS. The next example shows how to get meta-information from the specified FeatureClass in a personal geodatabase.

```
import sys, string, os, arcgisscripting, dataprocessing

GP = arcgisscripting.create()

DP = dataprocessing.create()

try:

  # Set datasource.
  Datasource = "C:/GeoData/Pgeodb/VectorPGB.mdb/basis"

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

    # Open metadata.
    GP.AddMessage("Open datasource: " + Datasource)
    MD = DP.OpenMetadata(Datasource)

    # Set metadata tag.
    Tag = "Esri/CreaDate"
    GP.AddMessage("Getting tag: " + Tag)

    # Get metadata info.
    Value = MD.GetValue(Tag)
    GP.AddMessage("Tag value: " + Value)
```

```
        else:
          GP.AddMessage("No metadata available.")

      except:
        Error = DP.GetMessages(2)
        GP.AddError(Error)
```

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

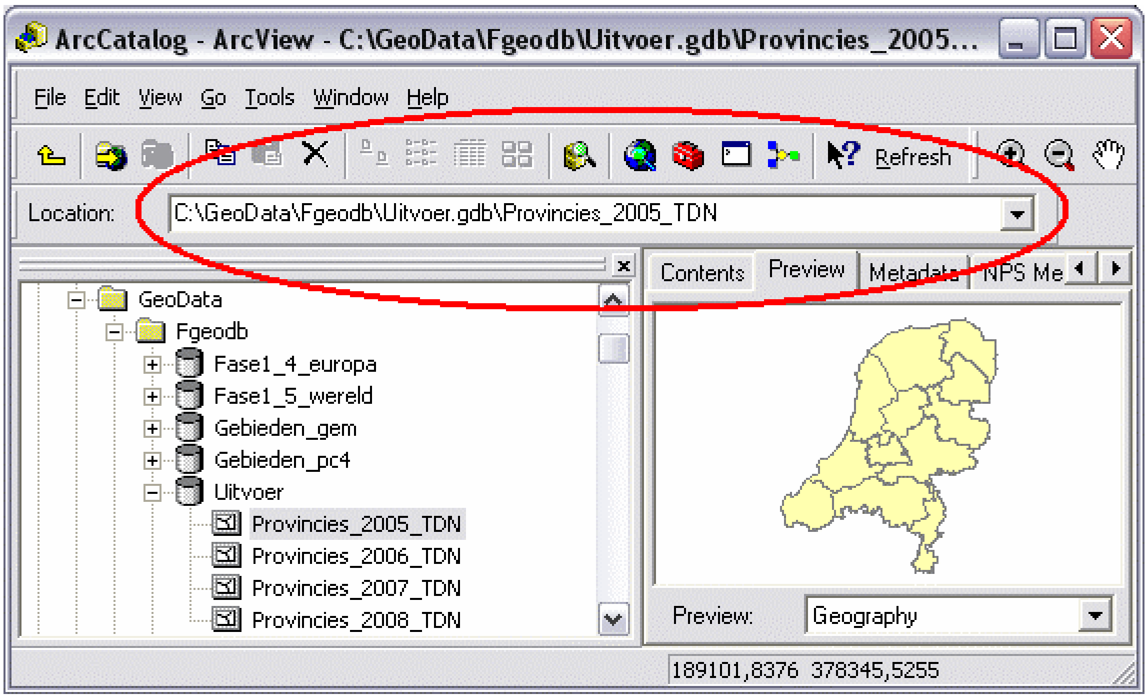


```
Executing: dptmetadata
Start Time: Thu Apr 23 13:48:29 2009
Running script dptmetadata...
Open datasource: C:/GeoData/Pgeodb/VectorPGB.mdb/basis
Getting tag: Esri/CreaDate
Tag value: 20090316 13:42
Completed script dptmetadata...
Executed (dptmetadata) successfully.
End Time: Thu Apr 23 13:48:39 2009 (Elapsed Time: 10,00 seconds)
```

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

9 Datasources and ArcCatalog paths

The 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 DataProcessing Tool uses this path to find the datasource.


ArcCatalog - ArcView - C:\GeoData\Fgeodb\Uitvoer.gdb\Provincies_2005...
File Edit View Go Tools Window Help
Refresh
Location: C:\GeoData\Fgeodb\Uitvoer.gdb\Provincies_2005_TDN
GeoData
Fgeodb
Fase1_4_europa
Fase1_5_wereld
Gebieden_gem
Gebieden_pc4
Uitvoer
Provincies_2005_TDN
Provincies_2006_TDN
Provincies_2007_TDN
Provincies_2008_TDN
Contents Preview Metadata NPS Me
Preview: Geography
189101,8376 378345,5255

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"

10 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. Because <metadata> is the default root node, it can be omitted.

```
      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 DataProcessing Tool is **not** bound to a specific metadata profile (CEN3, CEN4, ISO19115 etc.). The DataProcessing Tool is **profile independent**.

The 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 DataProcessing Tool also supports single files (for example Word documents, PDF documents, Excel spreadsheets, etc.) with metadata. In these cases the metadata should be in a separate file called <filename>.doc.xml, <filename>.pdf.xml, <filename>.xls.xml etc..

11 Using metadata shortcuts

To make working with metadata a lot easier the 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 file:

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

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

```
<?xml version="1.0" encoding="utf-8" ?>
<Config>
  <ShortcutTemplates>
    <File>MetadataKeywordsCEN3.xml</File>
    <File>ShortcutTemplates\MetadataKeywordsCEN4.xml</File>
    <File>C:\MetadataTemplates\MetadataKeywordsISO.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.

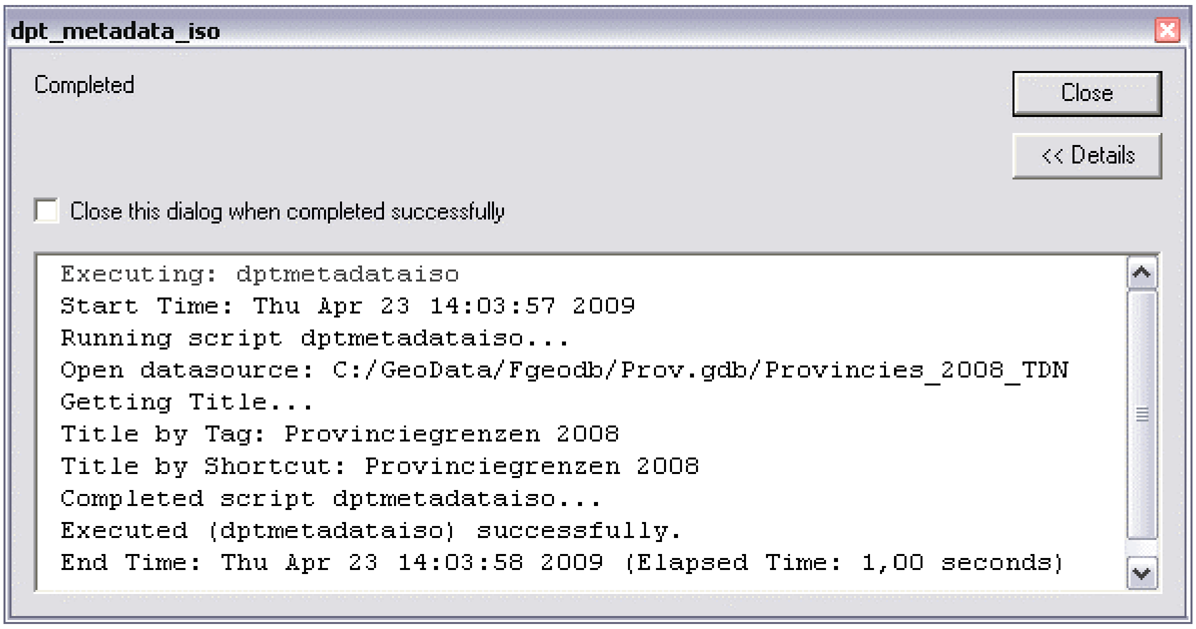
Note, when creating multiple shortcut templates be sure that all shortcut names are unique.

The following script is an example how to access the metadata of a datasource in a File Geodatabase.

```
import sys, string, os, arcgisscripting, dataprocessing

GP = arcgisscripting.create()

DP = dataprocessing.create()

try:

  # Set datasource.
  Datasource = "C:/GeoData/Fgeodb/Prov.gdb/Provincies_2008_TDN"

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

    # Open metadata.
    GP.AddMessage("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"
        GP.AddMessage("Getting Title...")

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

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

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

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

      else:
        GP.AddMessage("No metadata available.")

    except:
      Error = DP.GetMessages(2)
      GP.AddError(Error)
```

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



```
Executing: dptmetadataiso
Start Time: Thu Apr 23 14:03:57 2009
Running script dptmetadataiso...
Open datasource: C:/GeoData/Fgeodb/Prov.gdb/Provincies_2008_TDN
Getting Title...
Title by Tag: Provinciegrenzen 2008
Title by Shortcut: Provinciegrenzen 2008
Completed script dptmetadataiso...
Executed (dptmetadataiso) successfully.
End Time: Thu Apr 23 14:03:58 2009 (Elapsed Time: 1,00 seconds)
```

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]
```

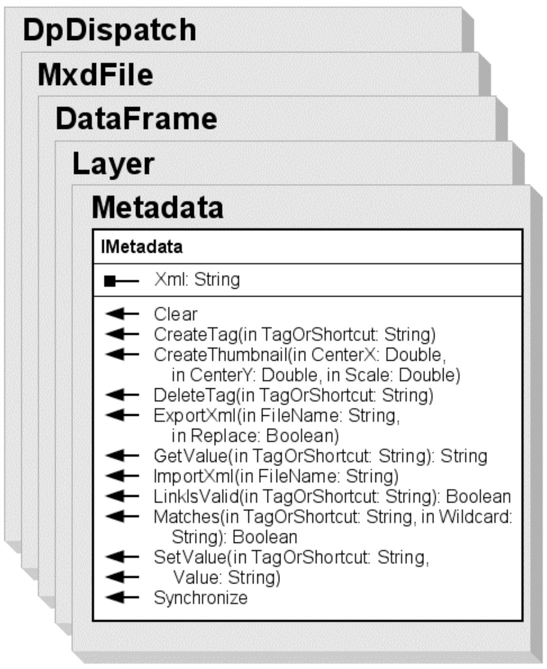
In order to know at what subtag level the index should be applied, dummy 0 indexes should be added at the end.

12 Performance

The DataProcessing Tool is primarily designed for use as a batch tool. Therefore priority is given to reliability and stability, rather than performance. In practice this means that at each call to one of the methods or properties of the DataProcessing Tool the accessed metadata, layerfile or mxd is opened and closed. This approach gives some execution overhead, but reduces potential problems with locking and loss of data.

13 Object model

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


DpDispatch
MxdFile
DataFrame
Layer
Metadata
IMetadata
Xml: String
Clear
CreateTag(in TagOrShortcut: String)
CreateThumbnail(in CenterX: Double, in CenterY: Double, in Scale: Double)
DeleteTag(in TagOrShortcut: String)
ExportXml(in FileName: String, in Replace: Boolean)
GetValue(in TagOrShortcut: String): String
ImportXml(in FileName: String)
LinkIsValid(in TagOrShortcut: String): Boolean
Matches(in TagOrShortcut: String, in Wildcard: String): Boolean
SetValue(in TagOrShortcut: String, Value: String)
Synchronize

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

- DpDistpatch

  This is the main object of the 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
  - 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 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

14 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.

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

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

Return value
Integer

MessageCount Property

The number of returned messages.

Read only.

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.

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

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

Return value
String

Version Property

Returns the current DataProcessing version.

Read only.

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

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

Return value
String

15 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.
DeleteTag	Deletes a xml-tag.
ExportXml	Exports the metadata to an external file.
GetAttributes	Returns the attributes of a xml-tag or shortcut.
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.
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.
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]

A empty tag will be created if the tag not already exists. If the tag exists the value will not be changed.

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**.

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**.

ExportXml Method

Exports the metadata to an external xml file.

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

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.

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.

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**.

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 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**.

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>/…

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>/…

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

Xml Property

Returns the metadata as Xml.

Read only.

Syntax
variable = *object*.**Xml**

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

Return value
String

16 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.

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.

17 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.

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

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.

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

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.

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.

18 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.
◀━	**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.

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

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.

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

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.

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.

Index Property

The index of the layer in a dataframe of grouplayer.

■━■ Read/Write.

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

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.

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

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.

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

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.

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

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.

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.

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

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.

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

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.