

ArisFlow

The connecting Genius

Version 2.8.1



User's Guide

Contents

What is New in this Version	8
What is New in this Version	8
Version 2.8.1	8
Version 2.8.0	9
Version 2.7.1	9
Version 2.6.1	10
Version 2.5.2.1	11
Version 2.5.2	11
Version 2.5.1	12
Version 2.4.1	14
Version 2.4	14
Version 2.3	16
Version 2.2.1	18
Version 2.2	18
Version 2.1	21
Version 2.0.8	24
Version 2.0.7	26
Version 2.0.6	27
Version 2.0.5	27
Introduction	29
Introduction	29
The ArisFlow Concept	31
The ArisFlow Concept	31
ArisFlow Flowchart	31
ArisFlow Definitions	33
ArisFlow Definitions	33
Variables	33
Directory Aliases	34
Action Types	34
Data Types	35
ArisFlow Execution	36
ArisFlow Execution	36
State of Actions and Data	36
Execution of Actions	37
Cyclic Integrity Checking	38
Working with ArisFlow	40
ArisFlow Executable Types	40
Starting ArisFlow	40
Registration	42
Registration	42
Single Use License	42
Floating License	43
Menu	45
Toolbar	49
Mouse Cursor	51
Flowchart	51
How to set up an ArisFlow Project	54
How to set up an ArisFlow Project	54
Quick Start of an ArisFlow Project	54
Opening an existing ArisFlow Project	55

Project Management	56
Properties	56
Drawing the Flowchart	57
Alignment and Spacing Functions	58
Selection of Flowchart Elements	59
Delete and Undo	60
Copy and Paste	60
Zoom	63
Data	63
Actions	65
Action Script	68
Action Script	68
Action Script Edit Field	68
Connected Input Data and Connected Output Data Listboxes	69
Action Script Buttons	70
Execution of the Action Script	70
Data Script Element Types	71
Variables	72
Directory Aliases	74
Directory Alias Repair	77
Environments	78
Action Types	81
Action Types	81
System Details	82
DDE Details	85
System/DDE Edit Fields	89
System/DDE Options	90
Data Types	91
Data Types	91
File and Directory Data Types	92
User-defined Data Types	95
User-defined Data Types	95
User-defined Data	97
Specifications of the Browser Program	98
Specifications of the Checker Program	99
Specifications of the Viewer Program	101
User-defined Data Access Program	102
Import/Export of Definitions	105
Import / Export of Definitions	105
Import / Export Through Definitions Files	105
Importing From Definitions Windows	106
Page Set-up	108
Page Set-up	108
Body Text	108
Margins	109
Header/Footer	110
Flowchart Tab	111
Page Grid Lines	111
Print	113
Print	113
Print Preview	114
Print To File	115
Preferences	117
Preferences	117

Execution Preferences	117
File Preferences	119
Trace Preferences	120
In Case of Failure	121
Browsing a File	121
Pre-defined Option Selection	122
Variable Selection	122
Directory Selection	123
Quote Script Elements and Quote Clever	124
Tooltips, Context Menus and Special Key Options in Dialogs	125
Find Dialogs	126
Input=output Data	127
Check Status	128
Execution	128
Execution Progress	129
Execution Log File	130
Execution Parsing	135
Execution Parsing	135
Command Lines and Script Files	135
Quoting	138
ArisFlow as DDE Server	139
ArisFlow as DDE Server	139
How to interact with ArisFlow through DDE commands	139
ArisFlow DDE Commands	141
ArisFlow DDE Command List	141
Handling errors occurring during the DDE conversation	144
Managing the ArisFlow project	145
Managing the appearance of ArisFlow	146
Executing or checking the status of (parts of) the flowchart	147
Requesting the results of the last or the current execution run	148
Retrieving information about the flowchart and its components	149
Setting path definitions	154
Changing flowchart parameters	155
Obsolete commands	160
Utilities	161
Usage Hints	161
ArisFlow Commander	163
ArisFlow Commander	163
Starting the ArisFlow Commander	163
ArisFlow Commander Program	164
ArisFlow Commander File	166
ArisFlow Commander File	166
[Variables]	167
[ServerScript]	168
[Output]	169
[OutputList]	170
Commentary	171
Assignment of values to Commander Variables	172
Multi-part Files	172
Execution Commands	173
Assignments	173
Control Structures	175
Arrays	178
ArisFlow Commander Topic Handling	180

Commander Errors	181
Examples of ArisFlow commander files	182
Examples of ArisFlow commander files	182
Example of a Single-part ArisFlow Commander File	182
Example of a Multi-run ArisFlow Commander File	183
Example of an ArisFlow Commander File with Assignments, Control	186
Structures and Arrays	
ArisFlow Utilities	188
ADO table browser	189
ADO table checker	191
ADO table viewer	192
Scripting language for ArisFlow	204
ArisFlow-ArcView Server Extension	205
Send ArcInfo commands to a server	207
Execute an ArcMap 9.x VBA macro from the command line	208
ArcMap 9 data viewer	209
ArcMap data viewer	210
ArcView data viewer	211
ArcInfo coverage browser	212
ArcInfo coverage checker	213
Application that converts a space delimited file to a comma delimited file	213
Errorchecker	213
FileGeodatabase browser and checker	215
Geodata browser and checker	216
ArcInfo grid browser	217
ArcInfo grid checker	218
ArcInfo INFO table browser	219
ArcInfo INFO table checker	219
INFO table viewer	221
Conversion of Access to dBase	221
MDB table browser	222
MDB table checker	223
MSAccess table viewer	223
Force not-up-to-date checker	224
Force up-to-date checker	225
Parameterfile generator	225
Display a dialog window with text for a user action	226
Close a windows application	226
Start an application	226
ArisFlow Examples	227
Calling another ArisFlow project through the ArisFlow commander	227
Command Lines and Script Files	227
Execution Quoting	227
List ArisFlow directories - AFCommander	228
Topic Demonstration	228
ArcMap Geoprocessing - ModelBuilder	228
ArcMap Geoprocessing - Python	229
GeodataTypes	229
ArcInfo on local machine	229
ArcInfo where coverages and grids are treated as directories	230
ArcInfo on remote machine	231
AFScript - MsgBox	231
AFScript - ADO - MS Access	231
AFScript - ADO - Oracle	232

AFScript - ADO - SQLServer	232
AFScript - Sendkeys	233
AFScript - Idrisi	233
MapInfo command	233
MS-DOS	233
MS-DOS program	234
Windows	234
Windows program	234
Excel - DDE	234
Excel - AFScript	234
Word - DDE	234
Word - AFScript	235
ArcMap VBA Macro	235
ArcView command	235
ArcView script	235
ArcView run Avenue	236
ArcView Spatial Analyst	236
Kill ArcView3	236
License Agreement	237
ARIS Software License Agreement for ArisFlow	237
Index	240

What is New in this Version

What is New in this Version

This section describes:

- The changes in ArisFlow [version 2.8.1](#) relative to version 2.8.0;
- The changes in ArisFlow [version 2.8.0](#) relative to version 2.7.1;
- The changes in ArisFlow [version 2.7.1](#) relative to version 2.6.1;
- The changes in ArisFlow version 2.6.1 relative to version 2.5.2.1;
- The changes in ArisFlow version 2.5.2.1 relative to version 2.5.2;
- The changes in ArisFlow version 2.5.2 relative to version 2.5.1;
- The changes in ArisFlow version 2.5.1 relative to version 2.4.1;
- The changes in ArisFlow version 2.4.1 relative to version 2.4;
- The changes in ArisFlow version 2.4 relative to version 2.3;
- The changes in ArisFlow version 2.3 relative to version 2.2.1;
- The changes in ArisFlow version 2.2.1 relative to version 2.2;
- The changes in ArisFlow version 2.2 relative to version 2.1;
- The changes in ArisFlow version 2.1 relative to version 2.0.8;
- The changes in ArisFlow version 2.0.8 relative to version 2.0.7;
- The changes in ArisFlow version 2.0.7 relative to version 2.0.6;
- The changes in ArisFlow version 2.0.6 relative to version 2.0.5;
- The changes in ArisFlow version 2.0.5 relative to version 1.0.1.

Please check www.arisflow.com for the latest ArisFlow developments.

Version 2.8.1

This chapter lists the changes in ArisFlow version 2.8.1 relative to [version 2.8.0](#)

General:

- ArisFlow and ArisFlow Commander are now presented with multi resolution icons in Windows
- The Readme and Whatsnew text files are removed from the installation as this information is already available in help and user manual

Help:

- Added a note on [\[ServerScript\]](#) regarding special characters in variable interpretation
- Repaired links to ArisFlow Commander [List ArisFlow Directories example](#) and some [DDE Commands](#)

Utilities:

- [ArcMapView](#) is now compatible for use with ArcGIS 10.3.1
- Regional settings other than "US" are now better supported by [ArcMapView](#)
- Checking data with [FGDBBrowser](#) now ignores by default changes in index files
- [FGDBBrowser](#) has a new option "-a" to check all files (including index files)
- Browsing data with [FGDBBrowser](#) now defaults to a list in alphabetical order
- Fixed opening the help from [ArisFlow Commander](#)

Examples:

- Added an FGDB Raster dataset to the example data.
- The [geodatatypes](#) example now also contains FGDB Raster data
- For some [data types](#) the viewer definition is completed in the [geodatatypes](#) example
- [ArcMapModelBuilder](#) and [ArcMapPython](#) are compatible with ArcMap 10.3.1
- The ArisFlow Commander [List Directories example](#) has been modified with hints in case of some ServerScript errors

Operating System:

- See www.arisflow.com for the current supported operating systems.

ArisFlow compatibility:

- Projects of earlier ArisFlow 2.x versions can be opened.
- Projects saved in version 2.8.1 cannot be opened by versions before 2.8.0

Version 2.8.0

This chapter lists the changes in ArisFlow version 2.8.0 relative to [version 2.7.1](#)

Dialogs:

- Dialogs are opened with a default size and on a default position after starting ArisFlow
- Dialog size and position will be remembered within an ArisFlow session (until closing ArisFlow)

[Print](#):

- Special Directories, Environment Variables and Variables are new in the Definition section of Print report

Setup:

- Removed obsolete components in Installation Setup

Operating System:

- Support has ended for Windows Vista / Windows Server 2003 and older (but might still work).
- Windows 8 and Windows Server 2012 are new supported platforms.
- See www.arisflow.com for the current supported operating systems.

ArisFlow compatibility:

- Projects of earlier ArisFlow 2.x versions can be opened.
- Projects saved in version 2.8.0 cannot be opened by previous versions.

Version 2.7.1

This chapter lists the changes in ArisFlow version 2.7.1 relative to [version 2.6.1](#)

[Logfile](#):

- Special directories ("ArisFlow program directory" and "current project directory" are new in logfile: Name and Actual Value
- Environment variables are new in logfile: Name and Actual Value
- Variables are new in logfile: Name, Definition and Actual Value
- Directory aliases are extended with definition row
- Input and Output data extended with location and entry rows, both as Definition and as Actual Value
- Order in logfile has been changed from:
 - Action types, Directory aliases, Data types
 - to:
 - Directory aliases, Action types, Data types
- Action type and Data type in Action type and Data type section have one extra tabulation.
- Script file in Action execution section has one extra tabulation.
- Program definition is new in Action execution section with location and entry rows, both as Definition and as Actual Value
- Input and Output data: Data type is moved to next row
- Directory aliases: Definition and Actual Value start at next rows.

[View logfile:](#)

- Tabulation is changed for better reading quality

ArisFlow as server:

- New request: [[GetProjectPath](#)] to get the current project directory of the flowchart

Help:

- Help is now in CHM format so the help is suitable for Windows Vista/7/2008
- Examples are now separate topics in the help

Examples:

- A 32/64-bit compatible streameditor (sed.exe) is included with the examples [ArcMapModelBuilder](#) and [ArcMapPython](#)
- [ArcMapModelBuilder](#) and [ArcMapPython](#) are compatible with ArcMap 10.1

Utilities:

- [ArcMapView](#) is compatible with ArcMap 10.1

Bug fixes:

- [[GetVariableValueDefinition](#)] now returns the definition instead of the value.

Version 2.6.1

This chapter lists the changes in ArisFlow version 2.6.1 relative to [version 2.5.2.1](#).

64-bit platform:

- ArisFlow should now run under Windows 7 64-bit and Windows Server 2008 64-bit. ArisFlow itself is a 32-bit application.

[Examples:](#)

- The new example geodatatypes.afd contains most user defined datatypes that use the geodatabrowser or FGDBBrowser for browsing and checking geographical datasets.

Utilities:

- A new browser and checker for File Geodatabase datasets: [FGDBBrowser](#).

Version 2.5.2.1

This chapter lists the changes in ArisFlow version 2.5.2.1 relative to [version 2.5.2](#). This version includes the changes issued in the ArisFlow 2.5.2 Service Pack 1.

Examples:

- ArcMapModelBuilder example is now available for ArcMap 9.3 and ArcMap 10.
- In ArcMapModelBuilder the "Dissove field(s)" parameter was added in the toolbox and in the Python script in ArisFlow.
- ArcMapExec uses VBA and keeps working for ArcMap 9.2 and ArcMap 9.3. Because of deprecation of VBA in ArcMap 10 it is advised to use Python instead (see ArcMapPython example).

Utilities:

- ArcMapView is upgraded to be able to be used with ArcMap 10.
- ArcMap9View is kept for compatibility with ArcMap 8.x and ArcMap 9.x.
- ArcMapExec uses VBA and keeps working for ArcMap 9.2 and ArcMap 9.3.
- Because of deprecation of VBA in ArcMap 10 this utility is not available in ArcMap 10. For executing scripts in ArcGIS it is advised to use Python instead (see ArcMapPython example).

Help:

- Utility help-files are updated to reflect the above changes and some minor text changes in the help texts.
- The Examples Help file is updated to reflect the changes in the ArisFlow examples.

Bug fixes:

- Floating license did not check out after program exit.
- Floating license could not use server ip-adres in ini-file.

Version 2.5.2

This chapter lists the changes in ArisFlow version 2.5.2 relative to [version 2.5.1](#).

Menus:

- The context menu (right mouse click) has been redesigned. The menu depends on whether the mouse is on an action object, a data object or somewhere else in the flowchart. Many options have been added to the context menu.
- When clicking with the right mouse button on an action or data object, only that object becomes the selected object in the flowchart.
- Some menu options in the [Edit menu](#) were renamed.

Help:

- The section describing Delete/Undo and Copy/Paste was split and is rewritten.

Windows 7:

- ArisFlow and the [ArisFlow commander](#) program should run without problems under Windows 7.
- ArisFlow Help requires the WinHlp32.exe program. Windows 7 does not support WinHlp32.exe. However it can be downloaded and installed, after which the ArisFlow Help should work properly under Windows 7. When asking for Help in ArisFlow a window appears with the details and useful website links for this download.
Remark: due to security features in Windows 7 a macro warning (1037) may appear when starting the ArisFlow Help in Windows 7.

Bug fixes:

- Removing connections (in the Action Script dialog) did not work correctly in version 2.5.1. This problem is fixed.
- Paste Action made an incomplete script in version 2.5.1. This problem is fixed.
- It should now be possible to draw a connection using the context menu (right mouse click) without any problems.
- Renaming a directory alias in the Directory Alias Repair dialog might result in an error. This problem is solved.
- When page grid lines are shown and clicking near one of the edges of the flowchart ArisFlow dumped. This problem is fixed.
- When using tabs to navigate through any dialog the focus should not "disappear" anymore. Also all buttons, including the ones the pictures, can now be accessed using the tabs.
- ArisFlow should not dump anymore when attempting a print or print preview with no printer connected.
- Printed text (e.g. properties) is now outlined correctly.

Version 2.5.1

This chapter lists the changes in ArisFlow version 2.5.1 relative to version 2.4.1.

Menu:

- The [View menu](#) contains a menu option to display the contents of the Trace file.

Dialogs:

- In the Preferences dialog under the "trace" tab the user can define the preferred viewer for the trace file. The trace file is viewed by selecting the [View/Trace file](#) menu option.
- In the Action script dialog it is now possible to continue a line on the next line by starting that line with an underscore (_).
- In [Variable](#), [Directory Alias](#), [Action Type](#) and [Data Type](#) dialogs the dialog caption now consequently displays whether it is a new definition being defined, or whether the definition is being updated or read-only.

ArisFlow Commander Program (version 2.1):

- When the user clicks the contents button for displaying the contents of the Commander file or the Errorfile, the ArisFlow Commander Program now first checks whether any such file was already displayed earlier after that same button was pushed. In that case that open file now becomes the foreground window, instead of the commander opening another window displaying the contents of the commander or error file.

Examples:

- All ArcGIS examples were tested and should run without problems using ArcGIS 9.3.
- An ArcGIS ModelBuilder example.
- A kill ArcView3 example. Kills all processes with "ArcView GIS" as caption in the window.

Help:

- The section describing the Action Script dialog is rewritten.
- The section describing the import/export of action types, data types, variables and directory aliases is rewritten.
- The section describing the Trace file preferences (previously called Advanced preferences) was extended.

Registration:

- Registration now takes place using hardware fingerprints for both the ArisFlow development executable and the ArisFlow runtime executable.

Windows Vista:

- ArisFlow and the ArisFlow commander program should run without problems under Windows Vista.
- ArisFlow Help requires the WinHlp32.exe program. Windows Vista does not support WinHlp32.exe. However it can be downloaded and installed, after which the ArisFlow Help should work properly under Windows Vista. When asking for Help in ArisFlow a window appears with the details and useful website links for this download.
Remark: due to security features in Windows Vista a macro warning (1037) may appear when starting the ArisFlow Help in Windows Vista.

Bug fixes:

- The server commands [ExecuteUpToAction](#) and [ExecuteUpToData](#) would not execute correctly if any data or action happened to be selected in the flowchart. A message 'Action: "..."' is not executed because some input-data are not up-to-date' would then be displayed. This problem is now fixed.
- When browsing for the path of a data file, the select file dialog would often not show all relevant files in the directory browsed. This problem is fixed.
- When the datatype in de Data dialog is changed from a User-defined to File or Directory datatype the data's directory alias is now set correctly, instead of being wiped.
- When execution of selected actions fails because input data is not up-to-date the message displayed now correctly states the data were not "up-to-date" (instead of the very confusing "undefined").
- When zooming in or out or when changing the printed page margins the page grid lines do now zoom or change accordingly.
- When the print dialog is closed while the print preview dialog is still open ArisFlow should not dump anymore.
- The footer is now positioned as defined in the Header/Footer tab in the Page setup dialog during printing.
- When going to a different page in the Execution Log File dialog ArisFlow should not dump anymore.
- Importing of action types in the Action Type List dialog, of data types in the Data Type List dialog, of directory aliases in the Directory Alias dialog and of variables in the Variables dialog directly from other ArisFlow projects should now work correctly.
- Changing the selected action type from a program type to a non-program type in the Action dialog should not give any problems anymore when pushing the *Script* button.
- In some very rare circumstances the flowchart could be flickering continuously when started, making it impossible to work with it. This problem is now fixed.
- ArcView3 dataviewer (AVView) can now correctly display ESRI grid.
- Windows 2000, XP and Vista: Black rectangles surrounding edit boxes do not occur anymore for *description* field in [Variable](#), [Directory Alias](#), Action and Data dialogs, *action script*

field in Action Script dialog and *pre*- and *post*-fields in DDE and System dialogs and in the display of the Log File contents.

Version 2.4.1

This chapter lists the changes in ArisFlow version 2.4.1 relative to version 2.4.

ArisFlow Executable Types:

- ArisFlowRT (runtime version) is available as a separate executable.

Flowchart:

- When double clicking on an action or data in the flowchart the action or data cannot accidentally be moved by anymore

Dialogs:

- It is now also possible to Import definitions from ArisFlow (.afd) files as well as from definitions (.def) files.

ArisFlow Commander Program (version 2.0.5):

- With [Open](#) and [SaveAs](#) commands it is now possible to pass file entries only. The directory of the file opened/saved becomes the work directory, i.e. the directory of the ArisFlow Comander File (.afc file).

Bug fixes:

- When browsing for a file (e.g. after pressing the browse button in Data, File Preferences or User-defined Datatypes dialogs) all files of the type browsed for are listed again. This bug only occurred in version 2.4.
- The problems with [Open](#) and [SaveAs](#) commands when executing an ArisFlow Commander File with the ArisFlow Commander Program should now be solved.

Version 2.4

This chapter lists the changes in ArisFlow version 2.4 relative to version 2.3.

Flowchart:

- A context menu is displayed when clicking the right-mouse button.
- Tooltips describe characteristics of data and actions under the mouse-cursor.
- It is possible to move the name of an action independent of the action circle representing the action.
- When clicking in the flowchart near an action ArisFlow recognises this as clicking on that action.
- When clicking on the action name the action is selected as well.
- When an action is selected the action name is also enclosed by selection markers.

Dialogs:

- The context menu displayed when clicking the right-mouse button now also provides specific ArisFlow options.
- Tooltips are displayed when moving the cursor over any pre-defined script options or buttons.

- Automatic word wrapping for *Description* fields in Properties, Data, Action, [Directory Alias](#) and [Variable](#) dialogs.
- Dialogs have minimum sizes, preventing overlapping of fields and buttons.
- A Help section describing Tooltips, Special Key and Context-Menu Options in Dialogs.

Examples:

- An ArcGIS example for dissolving ArcMap Geoprocessing with Python.

Bug fixes:

- When editing a [directory alias](#) local or server path it is now possible to insert a root directory when text from the beginning of the edit line is selected.
- When selecting a script element at the end of a editbox (e.g. in the Action Script, DDE or System dialogs) and replacing it by inserting a new script element the end-of-line following the script element should not be removed anymore.
- The *Full Path* in the Data dialog should now be displayed correctly after assigning a datatype for the first time or when changing between datatypes with location as string and location as directory alias.
- The F1 key should generate the correct Help when pressed in any dialog.
- Capital letters in the project name are now displayed correctly in the flowchart caption.
- ArisFlow should not dump anymore when the "kill" button (the cross in the top right) is clicked in the [Variable](#) or [Directory Alias](#) selection windows.
- The *Cancel*, *OK* and *Help* buttons should now always align correctly in the File and Directory Data Types dialog.
- The *Apply* button in the Trace Preferences dialog should now become enabled after changing the file type.
- Sizeable dialogs have minimum horizontal and vertical sizes so that different fields should not overlap anymore.
- When two data have equal location and no entry or equal entry and no location assigned they are not considered to be equal anymore, and therefore do not qualify as input=output data.
- When Printing or Print Previewing any action types the base type displayed is now correctly displayed as "System" or "DDE" instead of "action type".
- The user-defined datatype setting "location as directory" should now always be correct after exporting the datatype to a definitions file and importing it from that definitions file into another flowchart.
- ArisFlow should not dump anymore after attempting to read a project file generated by an ArisFlow version later than this version.
- ArisFlow gives an error message instead of dumping when attempting to read any non-existing files or when attempting to write to a file on a non-existing or read-only directory. This applies to every type of file in ArisFlow, such as project files, import/export files, temporary files (when the TEMP directory is read-only), print files, backup files, log files, trace files, etc.
- Due to a storage bug in a previous ArisFlow version, a filepath could appear erroneously in the [Directory Alias](#) dialog. When leaving that dialog ArisFlow could dump. In this version the directory path read is fixed so that this problem does not occur anymore.
- ArisFlow Server commands [\[GetDirAliasLocalPath\]](#) and [\[GetDirAliasServerPath\]](#) now return the actual local/server path instead of the path definition.
- The ArisFlow Commander should not yield a "<Variable name> depends on recursive <variable name>" message anymore when the variable <variable name> is used more than once in the ArisFlow commander file.
- The ArisFlow Commander should now give correct error messages with unrecognised DDE requests containing brackets, instead of assigning the value between the brackets to the request variable.
- ArisFlow Commander assignments starting with an opening bracket "(" should now be evaluated by the commander instead of giving an ArisFlow error message.

- Inconsequences in the Command Lines and Script Files section have been corrected.

Version 2.3

This chapter lists the changes in ArisFlow version 2.3 relative to version 2.2.1.

ArisFlow Commander:

- It is possible to use Assignments, Control Structures and Arrays in ArisFlow Commander Files.
- The ArisFlow Commander Program statusline displays the last command send to ArisFlow.
- The request [\[GetArisFlowPath\]](#) returns the ArisFlow base-directory path.

Directories:

- [Directory alias](#) local paths can be defined using the «ArisFlow program directory» or the «current project directory» pre-defined options. The first option replaces the ARISFLOWDIR environment variable, which is not used by ArisFlow anymore.
- When reading older flowcharts ARISFLOWDIR environments are automatically replaced by a directory alias that uses the «ArisFlow program directory».

View data:

- When viewing data without a viewer defined in the datatype definition, ArisFlow attempts to start the viewer associated by Windows for the data path extension.

Action script dialog:

- The action script field has non-proportional font Courier 10 points.
- When exporting the Action Script ArisFlow suggests the action name as exportfile name.
- The Data Script Element Type dialog now displays the data path for the selected script element type.

Dialogs:

- The F1 key should generate the correct Help when pressed in any dialog.
- SaveAs dialogs have default filenames filled in.
- When the data name in the Data dialog has not yet been changed it is automatically the same as the data entry without extension.
- The Print to File dialog has changed. The print file or files are now selected when the *Print* button is pressed.
- A value entered in a combobox in any dialog can be wiped by making the combobox the active window and pressing the *delete* or *backspace* key.

Execution:

- Blank lines are not passed as execution commands anymore. However when executing as script blank lines are included in the script file (see System and DDE action type definitions).

Utilities:

- A new generic geodatabrowser and checker for a range of geographical datasets: [Geodatabrowser](#). The following dataset types are supported:
 - ArcInfo Coverage or Featureclass;
 - ArcInfo Grid
 - ArcMap Layer File (lyr)
 - CAD datasets (dxf, dwg, dgn)
 - Images (jpeg, bmp, tiff, gif, MrSid, bip, bil, gis, img, ras, png, asc)

- Personal Geodatabase
- PC Arc/Info Coverage or Featureclass
- ShapeFile
- TIN
- MapInfo file (map, tab)
- The older browsers/checkers for ArcInfo coverages and grids have become obsolete: covbrws, covchk, gridbrws and gridchk. These browsers and checkers are still included in the setup but will not be maintained anymore.

Examples:

- An SQLServer example is included, using the utility ADObrowse for browsing for an SQLServer table. The utility ADOcheck (with "number of bytes") is used for determining changes in the content of the SQLServer table. The ADOview utility is used to view the contents of the SQLServer table.
- The new example geodatabrowser.afd contains most user defined datatypes that use the geodatabrowser for browsing and checking geographical datasets.
- Examples containing dataset types covered by the new geodatabrowser are changed in order to use this new browser/checker utility.
- Examples using "standard" viewers like Notepad, MS-Word etc. now use the viewer associated by Windows for the data path extension.

Bug fixes:

- Problems with undefined environments should now be solved.
- When clicking in the flowchart with the left mouse-button after moving an action or data this is not considered to be a double click anymore.
- When pressing the Ctrl-right key combination in an edit field with the cursor position before a blank the cursor moves to the first word beyond the blank, not the second word.
- When importing variables or directory aliases from another flowchart the description of these aliases is now also imported.
- The definition type dialog (such as variables, directory aliases, action types or data types dialog) now displays the correct unique name of an imported definition that has the same name as an already existing definition.
- When browsing a directory in the Directory Alias dialog and the current local path definition starts with a directory separator (backslash \ or forward slash /) ArisFlow should not crash anymore.
- When ArisFlow is in read-only mode (e.g. when *Showing* the contents of a datatype that maybe imported or when ArisFlow is *executing*) directory alias comboboxes should display the correct directory alias.
- When ArisFlow is in read-only mode all browse and specify buttons are now disabled.
- When attempting to [view data](#) without a datatype assigned to the data ArisFlow gives an error message instead of a dump.
- When location and/or entry fields of data are updated after browsing directories aliases and/or variables in the previously entered in these fields will stay in these fields, if possible.
- The *Full Path* field in the Data dialog of user-defined data with location not as a directory alias should now always display the correct path.
- When browsing user-defined data where the data location is stored as a string ArisFlow does not unnecessarily ask for a directory alias definition anymore.
- When ArisFlow calls the browser when browsing user-defined data ArisFlow should now enclose the location in double quotes.
- When closing the Data dialog before the user-defined data browser has terminated ArisFlow should not crash anymore.
- In some situations ArisFlow used to read "" instead of "OK" as the first line of a checker file, making the user-defined data undefined. This was because ArisFlow would start reading the checker file as soon as it appeared, which could be before anything was written to it. Now ArisFlow first checks whether the checker program has terminated before checking the

contents of the checker file, ensuring the checker file is completely written by the checker program before it is evaluated.

- When the datatype of data with status up-to-date is changed from a file or directory to a user-defined datatype ArisFlow should not crash anymore when saving the project.
- When pressing *OK* in the Action Script dialog of an action that connects Input=output data ArisFlow should not dump anymore.
- Writing information to the Trace File is not abandoned anymore (message: "Could not write to trace file") when many lines are to be written to the trace file in short intervals.
- ArisFlow should not dump anymore when printing with option *Print Execution Results* checked where the logfile does not exist.
- When pushing the *Runs* radiobutton a second time in the Print or Print To File dialog the page number fields do not disappear anymore.
- An error message is generated when the ArisFlow commander attempts to start up ArisFlow, but ArisFlow is not present on disk where it should be.
- Commands [\[SetDataEntry\]](#) and [\[SetProgramEntry\]](#) should not reset the data or action location directory alias anymore.
- After a succesful ArisFlow Commander Program run the error file from a previous run is deleted, if present. This also lead to an adjustment in the AFCommander [example](#).
- Under Windows 2000 the correct version of comcat.dll should now be installed, if that file was not already installed on the system directory.

Version 2.2.1

Bug fixes:

- Undefined environments should not lead to a "Program Error" message anymore when opening an existing ArisFlow project.
- ArisFlow should not dump anymore when pressing *OK* in System Details, DDE details, File or Directory datatype details or User-defined data details dialogs.
- When importing the contents of a script file in the Action Script dialog this should now be inserted correctly into the action script.
- When browsing user-defined data in a new directory a directory alias for that directory should now be enquired and filled in correctly in the Data dialog by ArisFlow.
- ArisFlow should not dump anymore when browsing for data in a Data dialog where the datatype has just been set for the first time.
- Help text should be now displayed when the *Help* button in dialog Variable List, Directory Alias List, Action Type List or Data Type List is pressed.
- Help examples in System Details dialog and Command Lines and Script Files chapters have been adapted to show correct error handling through a controlfile.
- The System Details and DDE dialogs now check again whether they contain exactly one occurrence of «shellscript» in the *Program* field and exactly one occurrence of «actionscript» in the *User* field.
- ArisFlow acting as a DDE server should not dump anymore when it detects an error while the showing of error messages is suppressed.

Version 2.2

This chapter lists the changes in ArisFlow version 2.2 relative to version 2.1.

ArisFlow as a DDE Server:

- A server accessing ArisFlow through DDE commands may use the filepath or the fileentry of the ArisFlow project file as a topic. This is described in the How to interact with ArisFlow through DDE commands section.
- The ArisFlow Commander is adapted so that DDE commands are send to the right ArisFlow project. This is described in detail in the ArisFlow Commander Topic Handling section.
- Requests [GetvariableCount] and [GetVariableName] are implemented.
- The request [GetActualPath] returns the actual path for a path definition containing environments, directory aliases and variables.
- Commands [WriteToFile] and [AppendToFile] can now write newline characters as well.
- DDE commands that set or append descriptions, such as [SetFlowchartDescription] and [AppendFlowchartDescription], can be passed with an empty string as description. This will reset the description or insert an open line into the description.

ArisFlow Commander:

- The ArisFlow Commander is adapted so that DDE commands are send to the right ArisFlow project. This is described in detail in the ArisFlow Commander Topic Handling section.
- A view button is introduced in the ArisFlow Commander Program screen. By pressing this button the contents of the ArisFlow Commander File or the Error file can be displayed and edited using Notepad.

Flowchart:

- When showing the name of data displayed in data blocks with multiple lines the text may be cut along hyphens, certain interpunction characters, and changes from digits to letters and from lower case to upper case letters as well.

Import:

- It is now possible to import variables and directory aliases from another ArisFlow project in the Variable List, resp. Directory Alias List dialogs.
- The Import Action Type and Import Data Type dialogs have been altered.

DDE dialog:

- Topics can be defined using directory aliases and variables.
- Control variables can be defined using variables.

Messages:

- When a user-defined data's checkerfile returns an error message on its first line that message is now displayed by ArisFlow.
- When viewing data a progress window appears when the checking of the existence of data takes a while. This enables the user to cancel the viewing.
- ArisFlow gives clearer error messages when attempting to view data while the location or entry of the viewer program is not defined in the datatype.
- The contents of the trace-file (see Trace Preferences) should be clearer to understand due to a better lay-out and more appropriate information.

Utilities:

- The program InfoView displays the contents of an INFO table (like tables used with ArcInfo coverages and grids).
- ArcMapView for viewing GIS-data with ArcMap 8.x.
- The ADOcheck utility has been extended with the option to detect changes in a database table, by checking information about that table in a lookup table.
- ADOview is a new utility to view the contents of any table that is accessed through an ADO connection.
- A VBA macro written in ArisFlow can now be executed in ArcMap using the new utility

ArcMapExec.

Examples:

- The new example AFTopic shows a DDE example with ArisFlow as the DDE server containing variable topics and using directory aliases and variables.
- An Oracle example is included, using the utility ADObrowse for browsing for an Oracle table. The utility ADOcheck (with "lookup table option") is used for determining changes in the content of the Oracle tables. The ADOview utility is used to view the contents of the Oracle table.
- The ArcInfo examples include the use of the new InfoView utility.
- The new ArcMapExec example demonstrates how to send a VBA macro to ArcMap using the utilities ArcMapExec and ArcMapView.

Bug fixes:

- Projects with undefined environment variables saved in a version prior to 2.1 should now open correctly, instead of displaying a "program error" message box.
- In Action Script definitions the .entry and .loc script elements should not occasionally disappear anymore.
- Upon cancelling the Action dialog the toolbar Save button is not enabled anymore.
- When opening a flowchart containing environments variables in any directory alias definition the the toolbar Save button is not enabled anymore.
- The connection between a copied data and not-copied action is copied with the *Paste Special* menu option.
- When data, that were connected to but not used by an action (in its action script), are changed such that they become so-called input=output data, thus making the connecting action defined, the action's status is now displayed correctly as not-up-to-date in the flowchart.
- When browsing for user-defined data by pressing the *Browse* button in a Data dialog with a location definition that uses a variable the correct location path should now be passed to the browser.
- When starting a user-defined browser for user-defined data by pressing the *Browse* button in a Data dialog where the browser details were not filled in in the User-defined details dialog ArisFlow should not dump anymore.
- The execution of more complex flowcharts should now always be performed completely and in the right order.
- A deadlock, which very occasionally could occur when ordering actions for execution, should be fixed.
- During execution ArisFlow could dump because it kept writing empty lines to a script file. This problem is solved.
- ArisFlow does not crash anymore after the evaluation period has passed.
- ArisFlow should not crash anymore when it cannot write to the trace file.
- ArisFlow should not crash anymore when the TEMP directory does not exist.
- ArisFlow should not crash anymore when printing the flowchart for a second time.
- When check status is cancelled ArisFlow now asks whether the user wants to continue when there are any actions or data that are not yet checked.
- When checking status ArisFlow would write some incorrect messages to the trace file, in particular that certain project-input data would become not up-to-date. These messages are adjusted.
- ArisFlow now returns "executing" after the server request [GetActionExecuteStatus] is passed while ArisFlow is still checking the input and output data before the actual execution. In older versions ArisFlow would return "none", making client programs like the ArisFlow Commander think ArisFlow was already finished with the execution.
- The ArisFlow Commander will now also wait for completion of execution after the [ExecuteAction] command, just like it does after the other execution commands.

- When the [CreateVariable] DDE server command is send to ArisFlow the value should now be assigned to the variable.
- The Save toolbar button should now become enabled after a [CreateDirAlias], a [CreateVariable], a [SetDirAliasLocalPath], a [SetDirAliasServerPath], a [SetVariableValue], a [RelocateDirAliasLocalPath], a [RelocateDirAliasServerPath] or a [RelocateVariableValue] command is passed to the ArisFlow DDE server.
- The ExcelOLE example now contains the ARISFLOWDIR environment variable instead of a hard directory path, so that it should now always execute correctly.

Version 2.1

This chapter lists the changes in ArisFlow version 2.1 relative to version 2.0.8.

Flowchart:

- Text wrapping is introduced for data in the flowchart. Multiple lines may occur in a data box, meaning that much more of the name of the data can be shown.
- Alignment and spacing functions assist in positioning the elements of the flowchart.
- The size of the flowchart is stored in the project file and automatically set upon opening the project.
- In the page set-up the flowchart font on the screen is shown and selected, instead of the print size font. The print size font is calculated before the flowchart is printed.
- When browsing for files or directories the start-up directory is set more consequentially.

Definitions:

- New in ArisFlow are [variables](#). Variables can be defined in directory paths, entry paths, action type definitions, action scripts, command definitions for checker, browser and viewers, print header and print footer texts.
- [Action types](#) and [data types](#) can be imported directly from other ArisFlow project files.
- Simple meta-information can be entered for [directory aliases](#) and [variables](#) through a description field.
- A simple *Edit* button has replaced the *Relocate* and *Redefine* pushbuttons when editing [directory aliases](#). The option "Affected data and actions become not-up-to-date" can be checked when editing directory aliases.
- It is now possible to create a new [directory alias](#) when selecting a directory alias for insertion into an edit field, without having to go to the Directory Alias List dialog.
- Environments used in directory alias and variable definitions should now be selected from a list, which contains all the defined environments. Environments are written in environment notation.
- Environments can now occur at any position in almost any path, either directly as environments or indirectly through [variables](#).
- Environments are not anymore written with a \$ character followed by the environment name. When writing environment paths in server commands use the environment notation or the [\[GetEnvironmentNotation\]](#) command.
- It is now possible to open and work with a project containing [undefined environments](#).
- A directory alias server path can also be derived from another [directory alias](#).
- With directory alias local path definitions no directory separator (\) is inserted automatically anymore between a root directory or environment and the remainder of the local path definition. In old projects the separator is inserted when the project is read.
- Quote Script Element and Quote Clever options should facilitate the proper insertion of quotes and directory separators in Action Scripts and System and DDE commands parsed for execution.

- When defining checker or viewer command lines for file, directory and user-defined datatypes it is possible to insert data location, data entry and data full path options.

Edit menu:

- A simple search mechanism for actions and data is introduced, analogous to the searching for undefined actions and data.

Execution:

- A new [execution menu](#) option "Execute up to selection" is introduced. This attempts execution until all currently selected data and actions are up-to-date.
- The execution order of actions is related to their positions in the flowchart. Different options can be selected through the Execution Preferences dialog.
- Blank lines in the action script of [action type](#) definitions are included in script files passed for execution.
- Handling of situations such as missing server directories and undefined temp directories is improved.
- When cancel is pressed during a check status the status of data and actions is not changed, instead of becoming not-up-to-date or undefined.
- The ArisFlow Help contains Execution parsing sections, which give extensive examples of execution commands send to the operating system for various settings for scripts files and script element quotation for the action type definition's System details.

Opening an existing ArisFlow project:

- It is now possible to open and work with ArisFlow projects containing [undefined environments](#).
- When an ArisFlow project contains any undefined directory aliases the user now has the option of repairing the undefined directory aliases or just accepting them. The repairing of directory aliases can be finished anytime, accepting all remaining undefined directories.
- When opening a flowchart its size and positioning is set according to the settings when the flowchart was last saved.

ArisFlow information:

- The log file and trace file now contains the version of the executing ArisFlow program.
- The log file also contains the actual commands executed.
- A section Usage Hints gives the user some useful hints on working with ArisFlow.

Dialogs:

- The dialogs [Directory alias](#), [Variable](#), Data, Action, System and DDE action type and File/Directory and User-defined data type have become sizeable. The Log File, Print Preview and Action script dialogs were already sizeable.

ArisFlow as a Server:

- Variable related commands: [\[CreateVariable\]](#), [\[GetVariableValue\]](#), [\[GetVariableValueDefinition\]](#), [\[SetVariableValue\]](#), [\[RelocateVariableValue\]](#).
- Meta-data handling commands and requests for directory aliases and variables: [\[AppendDirAliasDescription\]](#), [\[AppendVariableDescription\]](#), [\[GetDirAliasDescription\]](#), [\[GetVariableDescription\]](#), [\[SetDirAliasDescription\]](#), [\[SetVariableDescription\]](#).
- Server commands: [\[SetDirAliasServerPath\]](#) and [\[RelocateDirAliasServerPath\]](#) changes [directory alias](#) server paths.
- Server commands: [\[SetDataDirAlias\]](#) and [\[SetProgramDirAlias\]](#) changes the directory alias of data / program action.
- Server requests: [\[GetDataDirAlias\]](#) and [\[GetProgramDirAlias\]](#) returns the directory alias of data / program action.
- Server commands: [\[SetDataFullPath\]](#) and [\[SetProgramFullPath\]](#) changes the full path alias of data / program action.

- Server requests: [\[GetDataDirAlias\]](#) and [\[GetProgramDirAlias\]](#) returns the full path of data / program action.
- Server requests: [\[GetDirAliasLocalPath\]](#) returns actual path of directory alias, [\[GetDirAliasLocalPathDefinition\]](#) returns local path definition of directory alias. A similar distinction exists for server paths.
- Execution server commands: [\[ExecuteUpToAction\]](#) and [\[ExecuteUpToData\]](#).
- Requests: [\[GetDirAliasNotation\]](#) returns directory alias name in directory alias notation, [\[GetEnvironmentNotation\]](#) returns environment variable in environment notation and [\[GetVariableNotation\]](#) returns variable name in variable notation.
- Server command: [\[MessageBox\]](#) displays a text in a message box.
- Server commands: [\[WriteToFile\]](#) and [\[AppendToFile\]](#) write information send to ArisFlow to a file.
- With commands [\[Open\]](#) and [\[Quit\]](#) ArisFlow now closes all dialogs first, preventing a dump when these dialogs are closed later by the user.
- Obsolete command: [\[GetDirAliasString\]](#) is replaced by [\[GetDirAliasNotation\]](#). ArisFlow however will support [\[GetDirAliasString\]](#) up to version 2.x.

The ArisFlow Commander Program new version (1.3) has the following changes relative to version 1.2.5:

- The ArisFlow Commander Program waits until execution is finished for the new [\[ExecuteUpToAction\]](#) and [\[ExecuteUpToData\]](#) commands, as it already did for the [\[ExecuteAll\]](#) and [\[ExecuteAction\]](#) commands.
- Commander variables can be derived from other variables in the Commander file.
- [\[Output\]](#) sections can be grouped together in [\[OutputList\]](#) sections in the Commander file.

Utilities:

- [ADO browser](#) and [ADO checker](#) accessing any ADO database.
- [ArcView data viewer](#) for viewing GIS-data with ArcView 3.x.
- [MS-access table viewer](#).
- [OK](#) and [NOK](#), which always return "OK", respectively "NOK".

Examples:

- MS-DOS showing the use of [variables](#) in ArisFlow.
- Command lines and script files, showing which command lines are passed and which script files are created for different settings in the "Execute As" sections in the System Action Type dialog.
- Execution quoting, showing when and how quotes are placed around script elements.

Bug fixes:

- When opening an ArisFlow project containing input=output data these data are not set to the not-up-to-date status anymore.
- When opening a project from an MS-DOS prompt the [File menu](#) should display the correct filename, and it should be possible to open this project from the [File](#) menu.
- It should now possible to open a project with filename 7 characters and extension 3 characters by double clicking on the file in the explorer or from a MS-DOS prompt.
- When a [directory alias](#) path is changed any actions, of which the action script or the [action type](#) uses the directory alias, will become not-up-to-date [where applicable](#).
- [After repairing a](#) directory alias when opening a flowchart other directory alias paths may become existing as well. These directory aliases are now removed from the Directory Alias Repair dialog immediately.
- When user-defined data checkers return "NOK" this is now handled as it should be.
- When defining new actions and data the actiontype or datatype combo box should not display the first actiontype or datatype of the list anymore.
- The Help topic of the File Properties dialog and the ArisFlow commander help should now

exist.

- A data can still have the same name as an action, but confusion between these should not occur anymore.
- When copying an action and its attached data the action script elements in the Action Script should refer to the data linked to copied and copy actions.
- When browsing user-defined data ArisFlow will generate an error message if the location path is required for starting up the browser, and both the data location and the user-defined datatype's browser default directory have not been assigned a location path.
- After correct execution all temporary files should now be removed if the remove temporary files flag was set in the File Preferences dialog.
- When ArisFlow receives server commands with the wrong number of parameters it should not dump anymore.
- The startup directory when browsing for files in the commander program should now always be correct.
- The commander error file should no longer display any incorrect warnings that variables in the Commander file were not assigned a value.
- In the previous version the first [Output] section of a list was executed twice and the last section was not executed. This bug is fixed.

Version 2.0.8

This chapter lists the changes in ArisFlow version 2.0.8 relative to version 2.0.7.

Project management:

- File history list, which enables quick selection of ArisFlow project files the user previously worked with.

Flowchart:

- ArisFlow now supports so-called input=output data. Input=output data are used where data are both input and output of an action, for example database tables, which are updated by the action.
- When copying and pasting data in the flowchart, the location of the copied data is now assigned to the data copy as well.

Action types:

- The *Quoted Directory/Filename* option the System Type dialog is now called *Quote Script Elements*. Checking this option now encloses all script elements in quotes, not only directories and file paths as it used to be.
- The *Quote Script Elements* option in the System Type dialog now encloses every script element in quotes, including the script elements where the path does not contain any blanks.
- By default the *Quote Script Elements* option is now switched off. This is to make the user consider quoting script elements a useful option and not the standard situation.

Data:

- All parameters for browser, checker and viewer programs are enclosed in double quotes, even when these parameters do not contain any blanks.
- In previous versions data that cannot change (Files and directories, where no time/size nor checksum is checked, and user-defined data where the checker file contains one single line: "OK" or "NOK") were automatically set not-up-to-date when ArisFlow checked their statuses. These data now remain up-to-date, as no change was detected.

Check Status:

- When checking status, ArisFlow goes into execution mode to prevent user interference with the status checking process.

Temporary files:

- Temporary files are now written to the directory defined by the directory alias Temp, or to the environment TEMP, or to the ArisFlow directory. This directory is shown in the File Preferences section.
- During execution and checking status more information is written to the trace file. The trace file also has an improved layout.

ArisFlow commander:

- Being a utility the ArisFlow commander is placed in the ArisFlow util directory and not anymore in the ArisFlow bin directory.
- Commentary lines in the ArisFlow commander file can be indented.
- It is possible to start the commander program with options and parameters.
- Variables which are declared but not used will give warnings instead of errors.
- In previous ArisFlow versions the user could define one run, possibly with multiple output sections. The ArisFlow commander can now also handle multiple parts with varying numbers of output sections. This is useful for example where flowcharts for different countries are executed and where one flowchart combines these results into results for a larger region.

Utilities:

- There are new utilities for accessing MS-Access databases. These are a browser: MdbBrowse, a checker: MdbCheck and a utility for converting an MS-Access table to a dBase table: Mdb2dbf.
- Minor changes were implemented in utilities AFScript and Wkill.

Examples:

- Access.afd is removed.
- Ado.afd has become AccessADO.afd.
- Excel.afd has become ExcelDDE.afd.
- Word.afd has become WordDDE.afd.
- Arcinfo_local_dir.afd: the datatypes "coverage" and "grid" are not user-defined datatypes anymore, but have become of directory base type instead.
- Arcinfo_local_dir.afd: In the definition of actiontype "ArcInfo server local dir" the shell script user definition is changed to the easier but less controllable: arc &run «action script».

Bug fixes:

- Instead of automatically inserting a backslash (\) between location and entry in data when using server directories, ArisFlow now inserts either a forward slash (/) or backslash (\) depending upon the directory separator that is used in the definition of the server directory path.
- Other problems occurring when using server directories should be solved. Also when using a server directory where the server path is not defined an error message is displayed.
- In the Directory Alias dialog a local path definition which ends with a slash is now recognised as an appropriate actual path.
- The problem, where very occasionally the selected directory alias in any directory alias edit box was not updated, is solved.
- With user-defined data where the location is defined as a string ArisFlow does not wrongly ask for a directory alias anymore.
- When checking status of data, the progress window is always displayed and will show the data currently checked.
- During execution the message displayed in the progress window is more accurate.

- Pressing Help in the ArisFlow Commander Program will start the ArisFlow Commander help.
- In previous ArisFlow versions, when the ArisFlow Commander would pass a Quit command to ArisFlow, it would not notice the shutdown of ArisFlow. This problem is now fixed.
- In previous ArisFlow versions actions, which were edited but not altered, could be set not-up-to-date. This problem is fixed.
- In previous ArisFlow versions when checking the status of user-defined data these could occasionally be set not-up-to-date. This problem is fixed.
- Occasional dumps, occurring when cancelling the definition of new action types and new data types, are eliminated.
- Occasional dumps, after errors were detected and displayed during execution, should not occur anymore.

Version 2.0.7

This chapter lists the changes in ArisFlow version 2.0.7 relative to version 2.0.6.

Project management:

- ArisFlow project files have improved and more efficient storage of ArisFlow project data.

User-defined data:

- User-defined data can have a directory alias as location. An automatic conversion is performed in all data of the user-defined data type when changing the location kind between location as a directory alias and location as a string.
- A full path can be defined for every type of user-defined data. Therefore the full path (.full) script element option can be used in action scripts.
- When changing the selected data type in the Data dialog the data entry is not changed anymore. The location is only altered when altering between location defined as directory alias and location defined as a string.

Directories:

- Default directory local paths, containing no characters at all, are allowed. These are treated as undefined directories. This also makes the import of directory aliases easier.
- Improved handling of directory paths and data paths, also in dialogs like the Directory Alias List. Better checking of the existence of the actual paths.

Execution:

- Improved handling of data of the file or directory data type, where the changes in time/date stamp, size or checksum are not checked.
- More information is written to the trace file.

ArisFlow as a server:

- ArisFlow Commander version 1.2.1, making it possible to comment out a single line by using two forward slashes (//).
- The following new server commands have been introduced: [AppendActionDescription](#), [AppendDataDescription](#), [AppendFlowchartDescription](#), [GetFlowchartTitle](#), [SetFlowchartTitle](#).

Help:

- Both Examples and Utilities are integrated in the ArisFlow Help file.

Examples:

- There are new examples: ADO, AFCommander, ExcelOle, Idrisi, MsgBox, SendKeys,

WordOle.

- There is a directory with import files (.def extension) from all examples.
- The ArcInfo example has been updated where user-defined data with location as a directory alias is used.

Bug fixes:

- Import action types and / or datatypes without directories.
- When the case of data / action names, [data type](#) / [action type](#) names or [directory alias](#) names is changed, the new name is now displayed correctly in the flowchart.

Version 2.0.6

This chapter lists the changes in ArisFlow version 2.0.6 relative to version 2.0.5.

Installation set-up:

- DefaultIcon and FileType extension are set to better values during set-up.
- Better handling of shared DLL's in set-up.

[Examples:](#)

- For Windows NT users the environment variable ARISFLOWDIR is now set in the registry instead of in the autoexec.bat file.
- Improvements in "DOS for Windows NT" example.
- New "Windows", "ArcView" and "MS-DOS program" examples.

Utilities:

- Test on &terminal in airun has been changed.
- [AfwServer](#) can now run scripts with attributes.
- UserAct returns an answer file (which contains OK if the action was successful) and a control file.
- [Dat2txt](#) can manage directory names with spaces.

Bug fixes:

- Importing existing definition for directory alias, data type and/or action type.
- File locations (directories) for print-to-file, log file, backup file and trace file.
- Cancel button in print-to-file window.

Version 2.0.5

This chapter lists the changes in ArisFlow version 2.0.5 relative to version 1.0.1.

ArisFlow as a server:

- A large set of DDE commands enables a DDE client to access ArisFlow as a server. This command contain extensive facilities for handling errors in commands passed by the client.
- The ArisFlow Commander is a very useful utility for accessing ArisFlow as a Server through DDE commands.

Flowchart:

- The font of the flowchart and of different printed text components can be set independently of each other in the Page Set-up.
- The positioning of flowchart objects has improved, in particular when Zooming.
- Copy and paste of actions, data and connections.

Directory aliases:

- A directory alias local path can be derived from a parent directory alias local path.
- Directory aliases can be relocated.
- Data of base type directory are browsed through a specific directory browser.
- Directory and file paths containing blank characters, and used in commands or script files passed to the system or to a DDE server, can be enclosed by double quotes, if required.

Execution:

- Control files can be used at any point in checking the execution of actions. The number of control files used is not limited.
- The trace file writes all commands, passed by ArisFlow during execution, status checking, file browsing, etc. The trace file is an Trace Preferences option.

Printing:

- ArisFlow project data can be Printed To Files: an ASCII file (text components) and a Windows Meta File (flowchart).

Dialogs:

- Some dialog screens (Execution Log File, Action Script, Print Preview) are sizeable. ArisFlow remembers the last size of these screens.
- Editing of some dialog edit fields (Action Script's script, System and DDE action type program, pre, user, post) has improved.
- Pushbuttons in a dialog can be "pushed" by Alt-keys from the keyboard. Each dialog can be exited by pressing the Escape (Cancel) or Enter (OK) key.

Introduction

Introduction

ArisFlow is a dataflow management tool for managing automated company and production processes in organisations. ArisFlow reduces the accompanying process-chain, with all its mutual relations between executable programs and data files, to a well-ordered graphic representation by means of a flow diagram.

ArisFlow is capable of executing and controlling the complete process-chain or parts of it and supports working with an unlimited number and variety of files and programs. ArisFlow can be useful in many situations, ranging from ad-hoc data analysis projects to building state-of-the-art applications, like e.g. decision support systems.

One of the key features of ArisFlow is its ability to detect the change of data or programs and to execute only those programs necessary. Working with ArisFlow facilitates real-time documentation and offers the possibility of reproducing the results of an entire project.

The concept of ArisFlow is based on the principle of a generic tool providing insight into the data or process flow and facilitating the re-execution of processes triggered by a change in data or programs (see Chapter The ArisFlow Concept). This insight is achieved by using a visual representation of the process chain as the main interface to the user. The tool is generic as a result of the implementation of definitions. Processes and data are generalised into generic definitions in order to avoid duplication and to facilitate easy-to-use pre-defined types of processes and types of data. An advanced set of status-checking options facilitates checking whether the process chain has to be executed in its entirety or only partly.

ArisFlow can work with many different kinds of processes, such as:

- Executables (MS-DOS, Windows, UNIX);
- MS-DOS batch-files;
- UNIX shell scripts;
- SQL scripts;
- Python scripts;
- Application specific scripts;
- OLE-servers, ActiveX- and COM-objects
- Processes, which can be accessed through the DDE protocol.

It is possible to use ArisFlow for any type of data, such as:

- ASCII-files;
- Binary files;
- Database files;
- Database tables;
- Spreadsheets;
- Text documents;
- (Geo) graphical files.

The interface of the flowchart and the numerous menu and toolbar functions make it easy to work with ArisFlow. The steps needed to create a flowchart and to get the processes executed are described in detail in the chapter: "Working With ArisFlow".

The requirements of the ArisFlow program are about 3 - 4 MB internal memory for execution. However, programs or applications used for the processes defined in the flowchart, and the operating system itself probably add up to much higher requirements.

ArisFlow was built using operating system independent libraries. Therefore ArisFlow can be made available for a range of commonly used operating systems (e.g. Windows, OS/2 and UNIX).

The ArisFlow Concept

The ArisFlow Concept

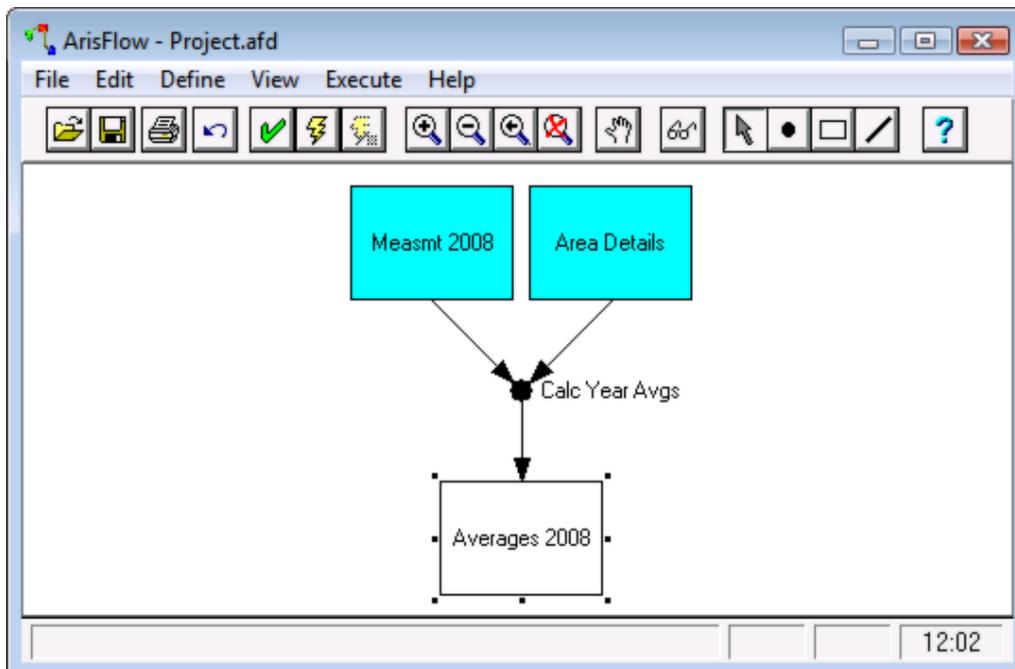
ArisFlow manages data flow processes. It is based on three major concepts:

- ArisFlow flowchart. The flowchart is a graphic representation of the data managed, the processes acting upon these data (called "actions" in ArisFlow) and how these data and actions are connected;
- Generalisations into definitions. Some actions (processes) and data can be grouped together because they share many characteristics. These common features are described in action and data types. Directories are also shared between many objects, which leads to the definition of directory aliases;
- Execution of the actions (processes) and managing the execution results. ArisFlow also detects whether execution of an action is required.

ArisFlow Flowchart

A flowchart in ArisFlow may contain the following graphic elements:

- Data, displayed as rectangles;
- Actions, displayed as circles;
- Connections, displayed as line elements.



ArisFlow project display.

Data are the elements, which are manipulated by ArisFlow processes (actions). A data object is recognised in ArisFlow by its logical name. Data represent a physical location, such as a file, directory, database table, or anything else the user can define.

Actions form the core of the flowchart. An action is recognised in ArisFlow by its logical name. An

action describes one or more commands which manipulate(s) data. These commands are called the "action script". The action script commands can actually be executed from ArisFlow. This execution may be performed by:

- The operating system, e.g. Windows or through DOS commands. Special cases are those where the operating system starts an executable program;
- Another program through the DDE (Dynamic Data Exchange) protocol.

Connections describe relations between actions and data. The connection arrow points to the output data. A connection is either from (input) data to an action or from an action to (output) data. In the ArisFlow flowchart this leads to the characteristic data-action-data-action-etc. appearance. Connections should conform to the following integrity rules:

- A data object can only be the output of a single action;
- Connections may not form a circle.

ArisFlow Definitions

ArisFlow Definitions

ArisFlow distinguishes four major definition components:

- Variables;
- Directory Aliases;
- Action Types;
- Data Types.

General definitions are very helpful in simplifying the creation of flowchart objects. As both data and actions share many common features, these features can be assigned by simply nominating the data type or action type of the data, or the action respectively.

Directory aliases define directory paths. Directory aliases are easier maintained than directory paths. Therefore wherever ArisFlow uses a directory path the directory alias is shown.

When starting an ArisFlow project it is good practice to create the definitions first, before any flowchart element is drawn. The user creating the flowchart can fill in these definitions. Alternatively, definitions can be imported into an ArisFlow project by means of a definition file, created from another project. The advantages of using definition files are:

- Easy to use. A new project can be started easily, as most or all definitions do not have to be redefined;
- Correctness. Definitions that have been used in other projects have possibly proven their usefulness already;
- Uniformity. Definitions can be maintained in a central place. Then all projects of one particular user, unit or company can use the same ArisFlow definitions.

Pre-defined action and data definitions can be found in the ArisFlow Examples directory.

Variables

Variables may occur throughout ArisFlow. A variable represents a string. Wherever the variable occurs that string is substituted for it. By simply changing the variable string definition every occurrence of it is updated automatically.

Variables may occur in:

- Other variables;
- Directory alias path definitions;
- File entry definitions;
- User defined data locations;
- Action scripts;
- Command strings, such as those used for starting browser, checker and viewer programs;
- Action type definitions;
- Printed pages.

Directory Aliases

Definition of directory aliases has the following advantages:

- It makes it easy to change locations of data and executable directories;
- It generates organised lists for directories.

The directory alias is a logical name for a directory. There are two directory path definitions:

- **Local path.** This is the character string by which a directory is known on the operating system currently running ArisFlow;
- **Server path.** This optional parameter in the directory alias definition is the character string by which the directory is known on a remote system executing an ArisFlow action.

Directory aliases are used everywhere in ArisFlow where directories are used. Often this is where a file is required, for example in data objects, program actions, back-up file definition, user-defined data programs, etc. In such cases the file is defined by the location (directory alias) and the file entry, which is the name of the file (including the extension).

Action Types

Action types generalise action characteristics. ArisFlow currently distinguishes two base types of action types:

- System;
- DDE.

System actions pass commands to the operating system. They can be Windows commands or commands passed to a DOS-box. Some system action types are programs. Actions of these types are to be linked to a physically present program executable.

DDE (Dynamic Data Exchange) actions pass commands to other executing programs. In DDE terms this program is called the "server". The server executes the commands. ArisFlow, the program passing the commands, is called the "client". This interaction between client and server is called a "DDE conversation".

Action type definitions define:

- The executor. This can be the system or any DDE server;
- Access to the executor;
- Monitoring the execution progress;
- Testing and interpreting the execution results.

The action type executor is either the system or the DDE server. The system executor is accessed, by embedding the action command into a command accessing the system. The DDE server is accessed, by sending the command to the server. This requires identification of the server accessed by the "name" and the "topic" of that server, both of which are to be defined in the action type.

Each command is executed separately. Commands may be grouped together into a script file. The name of the script file is then embedded into a command, which is then executed as one single command. The completion of the execution of a command is tested, as described in the Execution of Actions section. When execution has completed ArisFlow determines and assigns the new state to the action.

Data Types

Data types generalise data characteristics. ArisFlow distinguishes three base types of data types:

- Files;
- Directories;
- User-defined data.

Data types of the file base type are usually simple file entries located somewhere on the disk. The user may want to distinguish file types such as ASCII files, binary files, and specific in- or output files of any of the programs he/she is using (i.e. dBase files).

Data of the directory type have been specially developed for programs like ARC/INFO and Genamap, which use a directory for each map layer in their geographical databases.

The user-defined data base type opens the possibility to define many kinds of data types. User-defined data types can, for example, be used in combination with an RDBMS, like Access, Ingress and Oracle. Data of user-defined data base types are accessed through their location and their entry. Both location and entry are strings, which can be substituted where required by action scripts using data of the user-defined base type.

Data type definitions provide for three aspects of data:

- Browsing of the data;
- Checking the state of the data;
- Viewing the data.

The browsing of the data supports the user in indicating the location and the entry, which define the data. Simple file/directory browsers work fine for files and directories. User-defined data type definitions define a browser program, which performs the browsing, and how the browser program is to be started.

For ArisFlow it is important to detect when data have changed. A change means that actions using these data are no longer up-to-date and must therefore be executed. Files can be checked in terms of simple criteria, such as date/time of creation, size in bytes or a checksum algorithm. Directories can be tested by applying the same criteria to every file in the directory and in its sub-directories.

A change in user-defined data is detected by running the so-called "checker program" defined in the user-defined data type. A user-defined checker describes the state of user-defined data in a number of string lines. A change in any character in these lines implies that the state of the data involved has changed.

For viewing the data, specific viewer programs can be assigned to the data type. By pressing the "view-data" button the viewer program is started, with the data being viewed as input parameter. A viewer program can be specified for file, directory and user-defined data types.

ArisFlow Execution

ArisFlow Execution

The Execution of Actions is the most important feature of ArisFlow. The State of the Action and the Data linked to it determines whether execution of an action is required. The status of actions and data is checked and maintained by ArisFlow.

ArisFlow enables repeated execution of actions during a single execution run. The mechanism is explained in the Cyclic Integrity Checking section.

State of Actions and Data

Data and actions can have three possible states:

- Undefined;
- Not-up-to-date;
- Up-to-date.

Check status checks and updates the state of all actions and data in the flowchart.

Data are undefined (coloured grey in the flowchart) if:

- No data type has been assigned to them;
- No physical location has been assigned to them;
- The data involved are project input-data (not the result of any action), which do not exist. This means that:
 - Data of file or directory base type are not physically present on the specified location;
 - The user checker of user-defined data has not returned "OK" as result;
- The data involved are not project input-data (the data are generated by an action) but cannot be created. This means that:
 - The directory referred to by the directory alias does not exist for data of the file or directory base type or for user-defined data where the location is stored as a directory alias.
- The user checker of user-defined data has not returned "OK" or "NOK" as result.

Data are up-to-date if they are defined and:

- The data are project input-data; that is they are not the result of any action. Therefore project input-data are either undefined or up-to-date. Project input-data which are up-to-date are coloured light-blue in the flowchart;
- The data are the result of an action whose most recent execution was successful. These data are represented by a white rectangle with a black edge.

Data lose their up-to-date status if:

- The data type is changed;
- The data type, which is user-defined, has a different checker program assigned to it;
- The data location is altered;
- The physical data have changed in terms of size, time or checksum, depending on which properties should be tested according to the data type;
- The physical path of a directory alias used in the data location, has been changed (not relocated);

- The action of which the data are output data loses its up-to-date status.

Not-up-to-date data appear in the flowchart as white rectangles with a red edge.

Actions are undefined (coloured grey in the flowchart) if:

- No action type has been assigned to them;
- The action script has not been defined properly;
- The action script does not contain all the data connected to the action;
- The action type is a program, but the actual program has not been defined or is not physically present on the specified location;
- The action type has not been properly defined. This is usually because the action type definitions may contain directory aliases with a directory path that is not physically present.

Actions are set up-to-date (drawn in the flowchart as solid black circles) when they have been executed correctly.

An action is not up-to-date anymore if:

- Its action type changes;
- The definition of the action type is changed;
- Its script has been altered (and is still defined);
- Directory aliases or variables used in the action script or action type definition are changed (not relocated);
- Any of its input data or its output data lose their up-to-date status;
- The action is a program and this program has changed in size, time or checksum, depending on which properties should be tested according to the action type;
- The physical path of a directory alias used in the action type or the action program location has changed.

Actions, which are not-up-to-date, appear in the flowchart as solid red circles.

Execution of Actions

The most important feature of ArisFlow is the execution of its actions. The ArisFlow program has toolbar buttons through which either all or only selected actions are executed.

An action can only be executed if the action, its input data and its output data have the correct status:

- The action should be defined;
- All of the action's input data should be up-to-date. Project input-data (which are not created by any action; these data are drawn in light-blue in the flowchart) are always up-to-date when the data exist;
- All output data of the action should be defined, or it is not possible to create the new output data. Output data, which have non-existent directories, are marked as undefined.

When an action is to be executed, its script is parsed. Pre and post components, defined in the System or DDE action type dialog, are also evaluated. This leads to a number of statements, which can be grouped together into script files. The statements or the scripts are passed to the executor, which is either the DDE server program or the operating system. Before DDE statements can be executed, it may be required to start the DDE server program first. ArisFlow can facilitate this.

The monitoring of the execution progress is very important in ArisFlow. In particular:

- Whether execution of the statement has been completed. In that case ArisFlow will continue either by passing the next statement to the executor, or by starting the next action, or by terminating the execution process;
- Whether the execution of the statement has been performed correctly. If that is not the case, ArisFlow will skip the execution of the remaining statements in that action and continue with the execution of another action, if possible.

Both DDE actions and system actions pass commands to the executor. Commands may be grouped together in script files. In that case the name of script file, embedded in a command, is passed to the executor. ArisFlow monitors whether the command is completed correctly, before continuing with the next command. For each command there are two steps:

- A check whether completion of the execution of the command is acknowledged by either the operating system or the DDE server. For system actions acknowledgement is only checked when the action type's parameter Wait On Terminate is set. For example, if the command was executed through a DOS-box, ArisFlow checks whether the DOS-box is closed;
- In the Action Script dialog or in the DDE and System action type dialogs components control files may be defined. Before execution starts ArisFlow generates a unique name for each control file. The control file name is on the TEMP directory. There is no file with that name present before the execution of the action. The control file name is substituted in the command passed to the executor. Of course the execution of the action should ensure that the executor is writing each control file. After execution is completed ArisFlow checks the existence and the contents of the control file. The text written in the control file should be "OK", otherwise ArisFlow assumes the command execution has failed.

The execution of an action may involve one or more commands to be executed separately. After all commands have been executed and their execution has been completed correctly ArisFlow can perform the following extra tests:

- A DDE action type may have a control variable defined. ArisFlow requests the value of that control variable to the DDE server. The value returned by the server should be "OK", otherwise ArisFlow assumes that the execution of the action has failed;
- An ArisFlow Execution Preferences option is to request that the status of output data be tested after execution. If this option is checked output data must be different from before when the action was executed. If any output data have not changed, the execution has failed.

The action and its output data will only be set up-to-date, as after the execution the action has been completed correctly. For both input and output data and for program actions a stamp is assigned, which states the time, size and/or checksum for files and directories after the action execution is completed. To user-defined data the checker result is assigned. This stamp is used when checking whether the data have changed.

While execution is in progress, the results are written to the Execution Log File. This log file describes which actions have been executed, the scripts of these actions, and the result of the execution. When the execution of an action has failed, the reason of the failure is described in the log file. An execution failure message, with the failure reason, is also displayed on the screen.

Cyclic Integrity Checking

It is possible to repeat the execution of actions by means of the so-called "cyclic integrity checking" option. In this way ArisFlow facilitates the iterative execution of actions.

When ArisFlow executes in the normal (non-cyclic) mode, it will execute actions in a top-down manner. Actions, which have either no data or only project input-data, will be executed first. Next child actions, using output data from that action, may be executed, etc. Thus all actions

will be executed only once in an execution run.

With cyclic integrity checking set, ArisFlow checks the status of all data and actions after executing an action. Actions, which are now no longer up-to-date, will be executed once again.

To facilitate cyclic execution, some input data should change during the execution of the project. Most likely this is because an action at the bottom of the flowchart represents the same data as a data object higher up in the hierarchy. ArisFlow does not forbid multiple representations of data in the flowchart, making this construction possible. When the lower positioned data change, the higher positioned data lose their up-to-date status and ArisFlow re-executes all actions derived from those data.

Cyclic integrity checking is a powerful tool. It is important that at some point in time the data driving the cyclic execution should not change anymore, otherwise the execution will go on indefinitely (the user can, of course, break off the execution process manually).

Working with ArisFlow

ArisFlow Executable Types

There are two different executable types of ArisFlow:

- Project development executable (ArisFlow.exe);
- RunTime executable (ArisFlowRT.exe).

The development executable has all the features described in this chapter. The RunTime executable is specifically designed for the execution of ArisFlow projects (e.g. for distributing demos or projects on CD-ROM), and has less options for changing a project's data. Therefore most of the project data are displayed read-only.

The run-time executable allows the user to:

- Move flowchart objects (data and actions);
- Change meta-data, such as Properties and descriptions of Actions and Data;
- Printing, print to file and change any print settings as well as the page set-up for printing;
- Change log file and backup file entries in the File Preferences dialog;
- Create new [variables](#) and [directory aliases](#) and removal of unused variables and directory aliases;
- Change [variable](#) values and [directory](#) paths;
- Change location and entry of data;
- View data and change data viewer paths in [Data Type](#) dialogs.
- Check status of flowchart;
- Execute the flowchart;
- [Saving](#) the ArisFlow project.

The possibility to change directory paths is an important option in the run-time executable. This allows the user to execute similar projects with a varying directory set-up, for example where different directories contain different versions of input data for a certain project.

ArisFlow as a DDE server is designed to facilitate the execution of projects from outside ArisFlow. The RunTime executable can also handle all ArisFlow DDE-server commands.

Starting ArisFlow

ArisFlow can be started:

- From the Windows desktop by clicking its icon;
- By double-clicking the executable in the Windows Explorer. The executable is opened;
- From a DOS-box by typing the name of the executable.

ArisFlow can be started with one single parameter. The parameter should be the name of an ArisFlow project (".afd") file. ArisFlow will then automatically open that project.

When an existing ArisFlow project file is opened ArisFlow may perform certain steps, as described in the [Opening an existing ArisFlow Project](#) section. Before starting a new project it might be useful to read the [How to set up a new ArisFlow Project](#) section first.

Registration

Registration

ArisFlow has two license types:

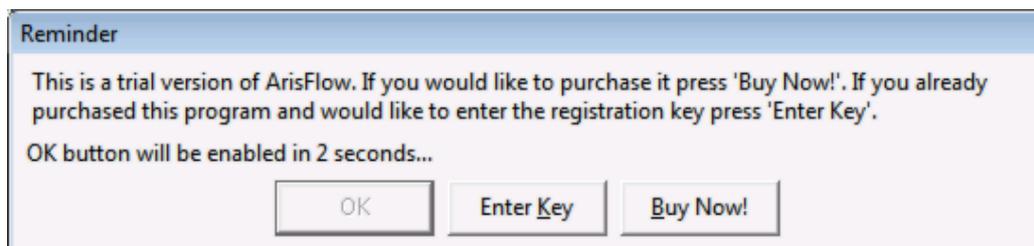
- [Single Use License](#). One license is valid for one computer only;
- [Floating License](#). A floating license is handled by a central server and allows a maximum number of users to simultaneously use the executable.

Registration of both license types now takes place using hardware fingerprints.

Single Use License

A single use license is installed as a 10-days evaluation version, which expires after the 10 days have passed. After this period, ArisFlow is locked until a valid licence key is entered. The evaluation version provides all the functionality of the ArisFlow standard version except for the [Print](#), [Print Preview](#) and [Print to File](#) functions.

While in evaluation mode, each time you start and leave ArisFlow and during some operations a reminder message will be shown:



ArisFlow trial version reminder screen.

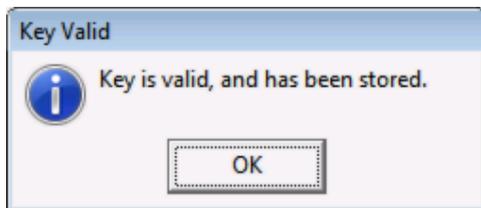
Pressing the Buy Now! button will take you to our online store, where you can order this product. Note that you will need the hardware fingerprint of the computer where you want to install the tool, shown in the dialog below (appears when you press Enter Key). After you complete your purchase, a personal registration key will be sent to you by email. Please store this key in a safe place.

Pressing the Enter Key button will present you with the following dialog, where you must enter your name and the registration key.



ArisFlow enter key dialog asking for registration name and key.

Once you have entered a valid registration key, press OK. A message box should confirm your registration:



Messagebox confirming valid key.

The key will be stored on your PC. The reminder message will not show anymore.

Floating License

A floating license is handled by a central server and allows a maximum number of users to simultaneously use the executable. This chapter describes how to register and start the central server license manager.

The floating license executable has no evaluation period. After the floating license executable is installed the ArisFlow License must be registered first. This is done by entering the following command in a DOS-box:

```
> ArisFlow.exe SERVER REGISTER
```

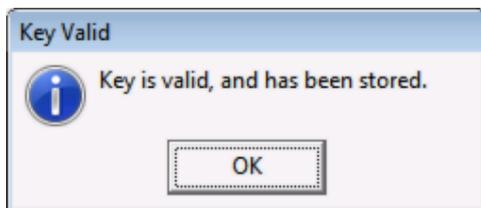
This example displays the registration of the ArisFlow executable, hence the "ArisFlow.exe" command.

The "Enter key" dialog-box appears:



ArisFlow enter key dialog asking for registration name and key.

Remember the hardware fingerprint and send it back to ARIS (mail to helpdesk@aris.nl). Then you will be given a name and key to register. You can enter these in the "Enter Key" dialog. A message box should confirm your registration:

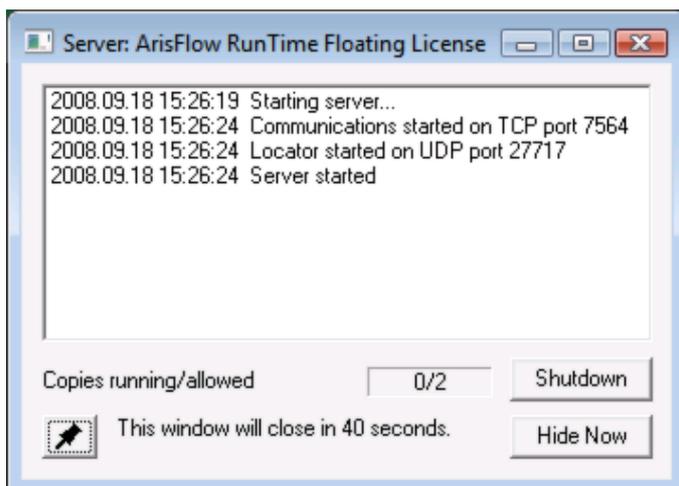


Messagebox confirming valid key.

After you have registered you can start the license manager:

> ArisFlow.exe SERVER

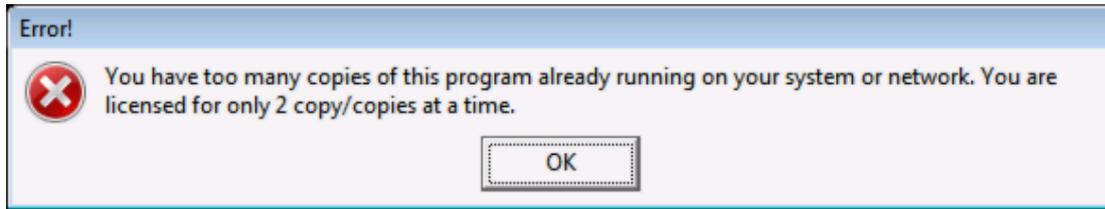
This shows the server dialog-box for sixty seconds. There can be only one server instance at a time per program.



ArisFlow server window.

When the server (or in this case the ArisFlow executable) is running the ArisFlow executable can be started on any client machine. A maximum number of users (as agreed when you purchased your license) can execute the executable at any time. The server window displays how many users are running the executable and how many are allowed to run the executable.

When someone tries to start the executable after the maximum number of users was reached that user receives an error message:



Error message displayed when maximum number of users was already reached.

The application can only be started after another user releases his copy. Note that it may take a while (approximately 30 sec.) before the server notices that someone has exited the application.

Other usefull commands on the server-side are:

- To start the server and display the server window for a certain time period instead of the normal 60 seconds:

> ArisFlow.exe SERVER10

This will display the server window for a maximum of 10 seconds. There should be no blanks between the SERVER and the time-out period.

- To start the server window without a timeout use the command:

> ArisFlow.exe SERVERX

Alternatively you can press the timer-button in the server window.

- To shut down the server:

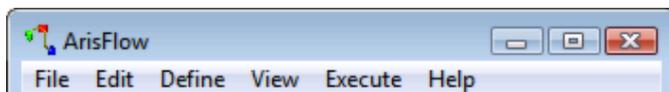
> ArisFlow.exe SERVERDOWN

After this the server must be re-started before any clients can start the executable (in this case ArisFlow).

- To unregister the hardware locked program:

> ArisFlow.exe SERVER UNREGISTER

Menu



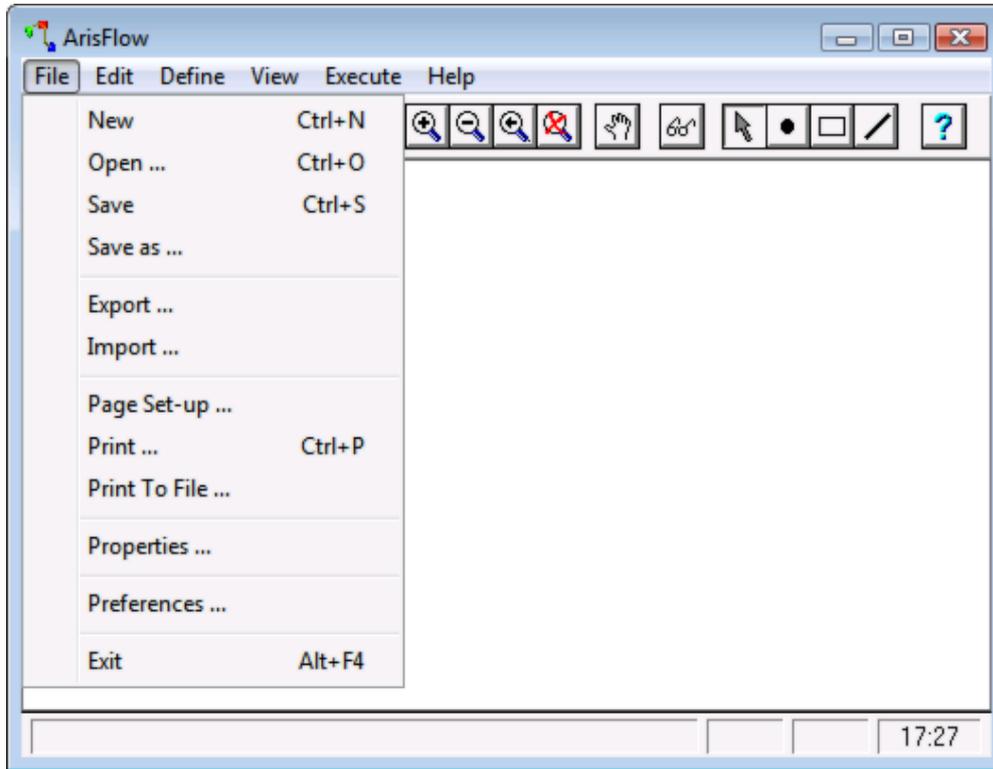
ArisFlow main menu.

The ArisFlow menu consists of the following sub-menus:

- **File.** This menu provides options for handling any project files, for setting project preferences, for setting up the printer options and finally for quitting the ArisFlow program;
- **Edit.** This menu provides options for selecting, changing and deleting any flowchart objects;
- **Define.** This menu provides access to the definition of directories, action types and data types;
- **View.** This menu provides options for viewing data, the execution log file, zooming in or out of the flowchart, and showing or hiding the printer grid lines;

- **Execute.** This menu provides the options for execution and checking the status of the flowchart;
- **Help.** The access to the ArisFlow *help* and the *about* box.

Most menu options can be performed by means of so-called "hot keys". Pressing any of these keys (which may be a combination of keys) starts that menu option directly. Hot keys are given for most ArisFlow menu options. Some menu options are also started quickly through toolbar buttons located on the Toolbar.



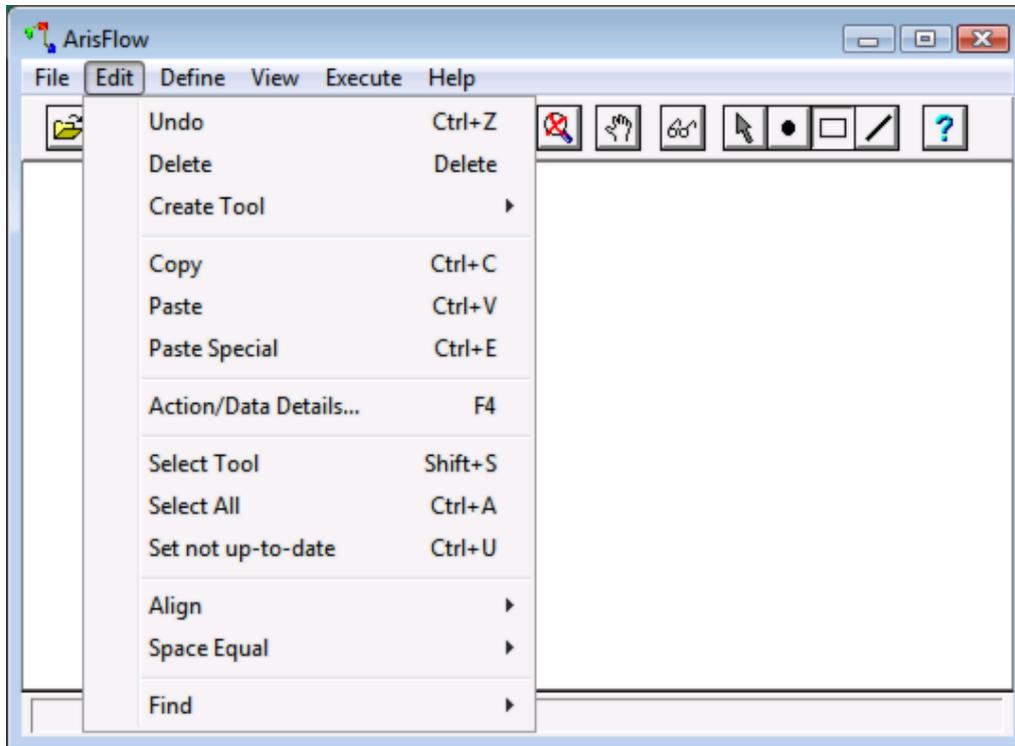
ArisFlow *File* menu options.

The *File* menu contains the following options, of which the first four are described in more detail in the "Project Management" section:

- *New.* Start new project. The currently opened project is closed;
- *Open.* Start an existing project. Asks user the name of the project to be started. The currently opened project is closed;
- *Save.* Save current project into its project file;
- *Save As.* Save current project into a file with a different name;
- *Export.* Export ArisFlow definitions to ASCII file as described in the Import / Export of Definitions section;
- *Import.* Import ArisFlow definitions from ASCII file as described in the Import / Export of Definitions section;
- *Page Set-up.* Assign printer page characteristics and flowchart font as described in Page Set-up section;
- *Print.* Define printer set-up, print preview and print ArisFlow flowchart, definitions and flowchart object characteristics as described in the Print section;
- *Print To File.* ArisFlow data is written to ASCII text (definition) and Windows Meta Files (flowchart) as described in the Print To File section;
- *Properties.* Display and edit general project meta-data as described in the Properties section;
- *Preferences.* Make project settings as described in the File Preferences and Execution

Preferences sections;

- File history list. Any of the last four ArisFlow projects the user has worked on can be selected as a menu option;
- *Exit*. Close flowchart and exit ArisFlow.

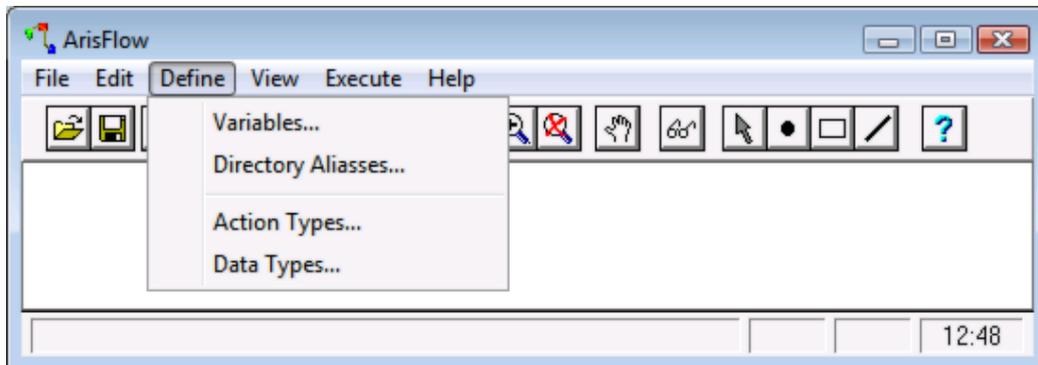


ArisFlow *Edit* menu options.

The *Edit* menu contains the following options:

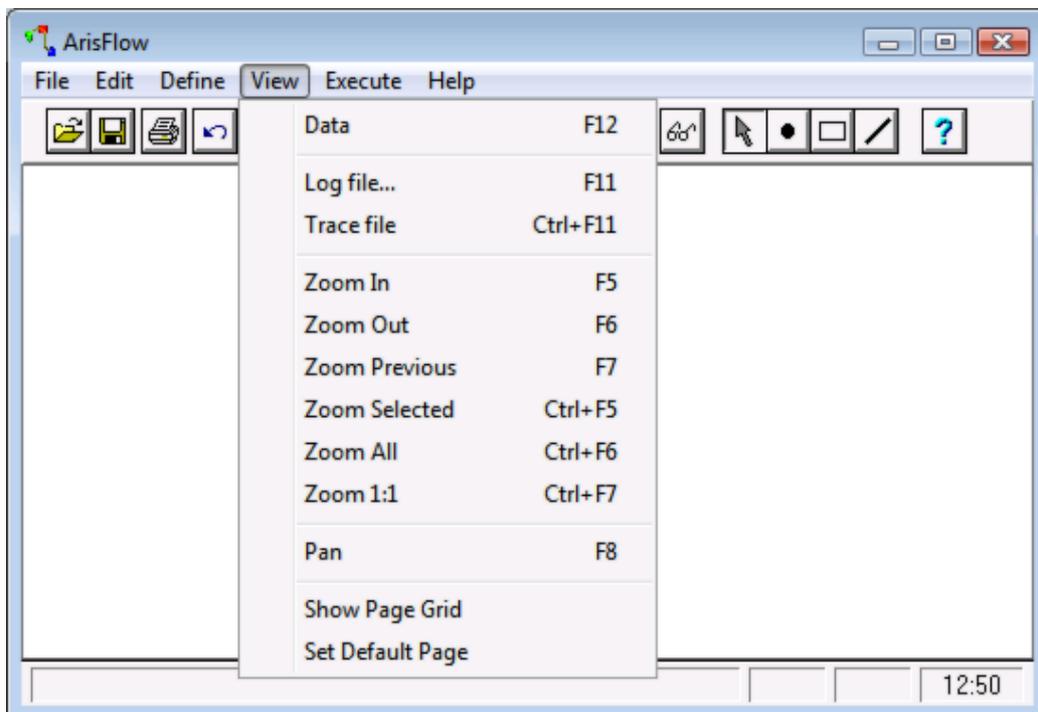
- *Undo Delete*. Reinstates deleted objects into the flowchart as described in the Delete and Undo section;
- *Delete*. Delete all actions and data currently selected. All connections connected to an action or data deleted are also removed;
- *Create Tool > Action/Data/Connection*. Set create tool to action, data or connection. The ArisFlow Cursor is set to the appropriate state;
- *Action/Data Details*. Open the Action or Data dialog of the action or data currently selected. Using the F4 hot-key is an alternative to double clicking the left mouse button on the data or action to be edited;
- *Copy*. Puts all selected flowchart elements (actions, data and attached connections) into the Copy Buffer.
- *Paste*. Drops all actions and data from the Copy Buffer into the flowchart. Connections between actions and data, which were both in the paste buffer, are also dropped into the flowchart.
- *Paste Special*. Drops all actions and data from the Copy Buffer into the flowchart. Connections involving one or both actions in the copy buffer can also be dropped into the flowchart.
- *Select Tool*. Sets the select tool and cursor, enabling selection of objects in the flowchart, and opening of Action or Data dialogs by double clicking on an action or data in the flowchart;
- *Select All*. Select every object in the flowchart;
- *Set not-up-to-date*. Selected objects, which currently have the up-to-date status (black colour), become not-up-to-date (red colour). Derived data and actions (and even parent actions) may become not-up-to-date as well;

- *Align*. Aligning of two or more actions and data to the right, left, top or bottommost object;
- *Space Equal*. Sets equal spaces in horizontal or vertical direction between three or more data and actions;
- *Find Action /Data/Undefined*. Lists all actions in the flowchart, or all data in the flowchart, or all actions and data, which currently have the undefined status. The actions and/or data are listed in the Find dialog. ArisFlow focuses on the action or data selected from this dialog.

ArisFlow *Define* menu options.

The *Define* menu allows for defining:

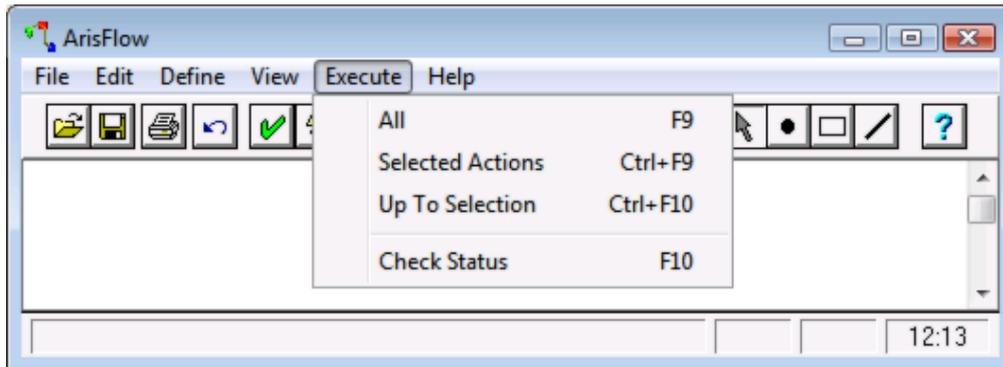
- Variables;
- Directory Aliases;
- Action Types;
- Data Types.

ArisFlow *View* menu options.

The *View* menu enables zooming and display of a number of entities. It has the following options:

- *Data*. View the currently selected data. This starts the viewer which was defined for the assigned data type in the File and Directory Data Types or User-defined Data Types dialog. If no viewer was defined and the data is a file the operating system will attempt to open the file

- represented by the data;
- *Log file.* Display the contents of the execution log file in a separate window;
 - *Trace file.* Display the contents of the trace file in a separate window according to the Trace Preferences settings;
 - Zoom options as described in the Zooming section;
 - Panning, which is described in the Drawing the Flowchart section.
 - *Show Page Grid.* Show the page grid lines, as described in the Page Set-up section. If page grid lines are currently showing (menu option has a check mark), the selection of this menu option will hide the page grid lines;
 - *Set Default Page.* Page grid lines are set in such a way that the screen is printed approximately 1:1 on the print paper.



ArisFlow *Execute* menu options.

The *Execute* menu contains the following options:

- *All.* Check status and execute all actions in the flowchart, which are not-up-to-date;
- *Selected Actions.* Check status and execute those flowchart actions, which are currently selected and not-up-to-date. This option is most useful in testing the execution of a single selected action;
- *Up To Selection.* Check status and execute those flowchart actions, so that eventually all selected actions and data should become up-to-date, providing the execution never fails;
- *Check Status.* Check which actions and data have changed, setting them not up-to-date. Data and actions, which become undefined (because directories or project input-data do not exist anymore), can also be detected.

ArisFlow *Help* menu options.

The *Help* menu has the options:

- *Contents.* Start help with the ArisFlow Table of Contents;
- *About.* Display information about ArisFlow, ARIS and Registration details.

Toolbar



ArisFlow Toolbar.

The ArisFlow toolbar buttons provide quick access to the most important Menu options. Clicking the button with the left mouse-button activates a toolbar button. When moving the

cursor over a toolbar button, the so-called "Tool tip" text displays which menu option is linked to the concerned toolbar button. At the same time more information about the menu option is displayed in the status bar, which is located below the flowchart.

The toolbar options are listed below. A brief description is given for each. For more detailed descriptions the user is referred to the Menu section.

Button	Menu option	Cursor	Short description
	File/Open		Open ArisFlow dataflow project file;
	File/Save		Save the current project into the project file;
	File/Print		Print, and printer set-up and print preview;
	Edit/Undo		Reinstate flowchart objects previously deleted;
	Execute/Check Status		Check status of data and actions;
	Execute/All		Execute all actions;
	Execute/Selected		Execute selected actions;
	View/Zoom In		Zoom in on the flowchart;
	View/Zoom Out		Zoom out of the flowchart;
	View/Zoom Previous		Return to previous zooming state and position;
	View/Zoom All		Display the entire flowchart on the screen;
	View/Pan		Panning, moves centre of display with cursor;
	View/Data		Start viewer program for selected data object;
	Edit/Select Tool		Cursor becomes select tool to select an object;
	Edit/Create Tool/Action		Clicking left mouse button in flowchart creates new action;
	Edit/Create Tool/Data		Clicking left mouse button in flowchart

	Edit/Create Tool/Connection		creates new data; Clicking left mouse button creates connection;
	Help/Contents		Enter ArisFlow Help.

When the Zoom In, Zoom Out, Select Tool, Create Action, Create Data or Create Connection button is pressed, that button remains pressed down. The accompanying cursor also appears in the screen. This is described in more detail in the Mouse Cursor section.

Mouse Cursor

The ArisFlow Flowchart cursor has 6 possible states, represented by the following cursors:

Cursor	Function
	Zoom In;
	Zoom out;
	Pan;
	Select Tool;
	Create tool is Action;
	Create tool is Data;
	Create tool is Connection.

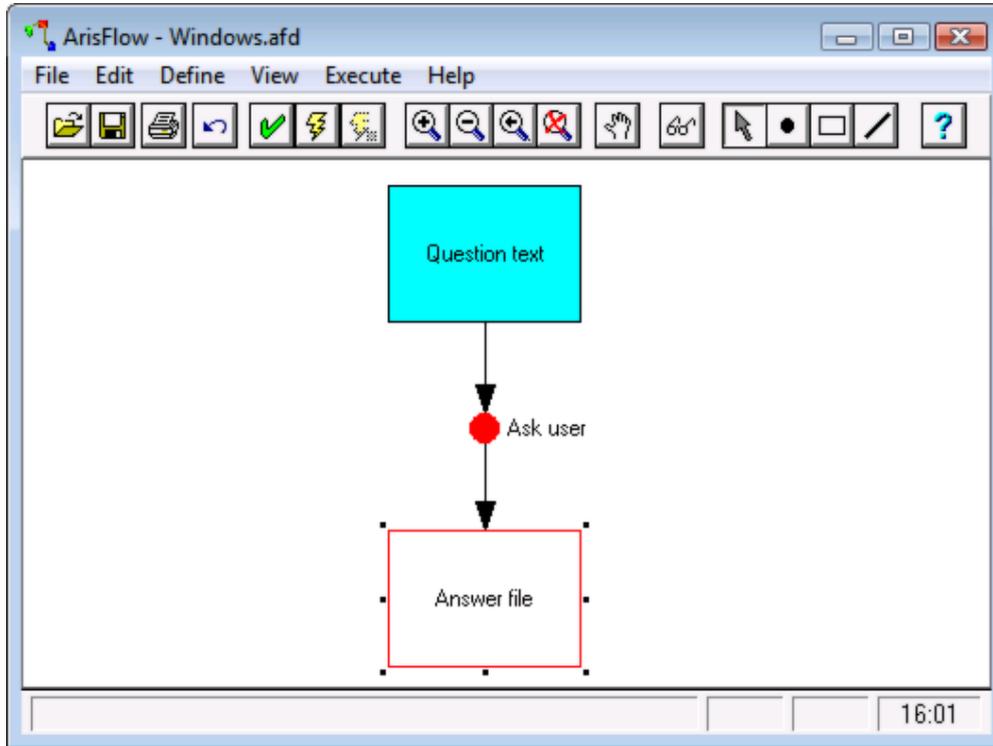
A cursor state is selected by pressing the toolbar button or by selecting the menu option linked to that cursor. This makes it possible to:

- *Zoom In or Zoom Out.* Alters extent of Flowchart displayed as described in the Zooming section;
- *Select.* Select and move flowchart objects and move page grid lines as described in the Drawing the Flowchart, Selection of Flowchart Elements and Print Page sections;
- *Create Action, Create Data or Create Connection.* Creates new action, data or connection when the left mouse-button is pressed, as described in the Drawing the Flowchart section.

Flowchart

The flowchart is a graphic representation, which may contain the following objects:

- Actions;
- Data;
- Connections between actions and data;
- Page grid lines.



Example of ArisFlow flowchart.

Actions are drawn as circles. The name of the action is displayed right of the action. The colour the action displays the state of the action. It can be:

- Grey. The action is undefined;
- Red. The action is not-up-to-date;
- Black. The action is up-to-date.

Data are drawn as rectangles. The name of the data is displayed inside the rectangle. Longer names could be displayed over multiple lines, which can be cut along blanks, hyphens, punctuation characters such as .,:;!/?_+*=/|~\@#& and changes between digit and character and from lower case to upper case letters.

A distinction is made between project input data and other data. Project input data can be drawn as:

- Grey with a blue edge. This resembles undefined project input data. Project input data can be undefined because the datatype, location or entry is not assigned to the data or because the data is not present in the disk;
- Light blue with a black edge. This resembles project input data, which are defined and therefore automatically up to date.

Other data can be drawn as:

- Grey with a black edge. The data is undefined. This can be because the datatype, location or entry is not assigned to the data or because the location is not present on the disk, making it impossible to create the data;
- White with a red edge. The data is not up to date;
- White with a black edge. The data is up-to-date.

Connections are drawn as arrows. The arrow can be directed:

- Away from the data towards the action. The data is **input data** for the action;
- Towards the data and away from the action. The data is created by the action, i.e. it is **output data** from the action;

The connection can be drawn with a:

- Solid line. The data connection does occur in the action script;
- Dashed line. The data is not named in the action script. As a consequence the action appears grey (undefined) in the flowchart.

Whether the connected data is used is also displayed in the Action Script dialog.

Page grid lines can be displayed by checking the *View/Show Page Grid* menu option. When printing the flowchart the grid lines marks the contents of a page to be printed.

When clicking with the right mouse button in the flowchart, a context menu is displayed. The menu options are determined by whether the user clicked on an action or a data object, or somewhere else in the flowchart. When clicking on an action or data that action or data becomes the selected flowchart object.

The *Drawing the Flowchart* section describes how to create flowchart objects, delete them, move them, zoom in or out, move page grid lines, etc.

How to set up an ArisFlow Project

How to set up a new ArisFlow Project

The building of an ArisFlow project involves two major steps:

- Definitions of Preferences, directory aliases, action types and data types;
- Drawing the flowchart and defining data and actions.

It is good practice to start a new ArisFlow project with the definitions. These definitions are the more abstract component of an ArisFlow project. Fortunately this major step is not required in most ArisFlow projects, as there are many options for a Quick Start of an ArisFlow Project.

Before the creation of the flowchart the following steps are recommended:

- Fill in the Properties dialog called through the *File/Properties* menu option;
- Assign File Preferences (back up and log file characteristics) for this project. File preferences have default values, but may require adaptation for the new project.

The next step is the creation of the flowchart:

- Position new data, actions and connections in the flowchart;
- Assign values to actions, data objects and action scripts through the Data, Action and Action Script dialogs;
- Create Directory Aliases, Action Types and Data Types which were not yet defined in the project;
- Define Execution Preferences;
- Test flowchart components by executing one or more selected actions occasionally.

Position data and actions in such a way that most connections point in a similar direction (usually down), otherwise connections and the related execution progression become very difficult to follow.

It is good practice to test flowchart components by executing one or more selected actions occasionally. Actions should have been tested before it can be assumed that the actions and their action types have been assigned correctly. In particular the design of action types and the use of user-defined data may require special attention.

Logical names should be given to data, actions, directory aliases, data types and action types. ArisFlow accepts almost any character string, which should be an incentive to make logical names as clear and as meaningful as possible.

Quick Start of an ArisFlow Project

Directory Aliases, [Action Types](#) and Data Types should be defined before they can be used in the drawing of the flowchart. Therefore these so-called definitions should be assigned first when starting a new project.

Fortunately many projects may share the same definitions. Therefore the more abstract task of creating definitions can be minimised or possibly even eliminated by:

- Using the "new.afd" project file in the New subdirectory of the ArisFlow examples directory;

- Starting with another ArisFlow project file and thus re-using its definitions;
- Import Definitions using an ASCII text file;
- Import action types and data types from other ArisFlow projects.

The "new.afd" project file in the Examples/New subdirectory of ArisFlow contains many possible definitions. This file can be opened in a new project, renamed and thus be the basis of the new project. Unwanted definitions can be removed. It is possible to start a project with this project file by making it the start-up parameter when starting the ArisFlow program.

Any existing ArisFlow project file could also be the basis of a new ArisFlow project. Again flowchart objects and definitions could be deleted easily. Make sure the new project is saved under a different name as the project being copied, e.g. by first copying (and renaming) the existing project file.

ArisFlow has mechanisms to export definitions in a project and import these into other projects. This could be useful when starting a new project, or when requiring a specific definition from an already existing project. The import / export takes place through an ASCII file. This file can easily be edited, in particular to remove any definitions, which need not be imported.

Definitions may contain directory alias references. For example the MS-Excel action type definition requires the directory where the MS-Excel program is located.

Sometimes directory aliases may refer to non-existing directories. When inheriting or importing these directory aliases the user may be required to re-define the locations of these directories through the Directory Alias Repair dialog, which is displayed when starting up the ArisFlow project.

Opening an existing ArisFlow Project

An existing ArisFlow project file can be opened:

- By selecting the [File/Open menu](#) option and browsing the ArisFlow project file to be opened;
- By selecting one of the maximum of four most recently opened ArisFlow project files listed in the [File menu](#).

When opening an existing project file ArisFlow:

- Checks the existence of all **environments** used in the project. If any of the environments used in the project is not defined the [Undefined Environment](#) dialog appears. It is recommended to define undefined environments before proceeding with the ArisFlow project;
- Checks whether all [directory alias](#) local paths are actually present on the system executing ArisFlow. If any local path does not exist the user is asked whether these directories should be repaired, starting the Directory Alias Repair dialog. Alternatively the user may accept all these directories as being undefined;
- Checks the status of all actions and data, if the *Check status automatically* option is set in the Execution Preferences dialog. As a consequence an Execution Progress window may appear until the status checking is finished.

Project Management

Project management options are located under the [File menu](#). They are the *New*, *Open*, *Save*, *Save As* and *Exit* options. An ArisFlow project is stored in a single binary project file. The files have default ".afd" extension. Currently ArisFlow has a single device interface: only one project file can be open per ArisFlow program executing. So when starting another project the currently opened project is automatically closed first.

When closing a project, to which changes have been made that have not yet been saved, the user will be asked whether these changes should be saved before closing the project. It is always obvious that the *Save* toolbar button is disabled when there is nothing yet to be saved.

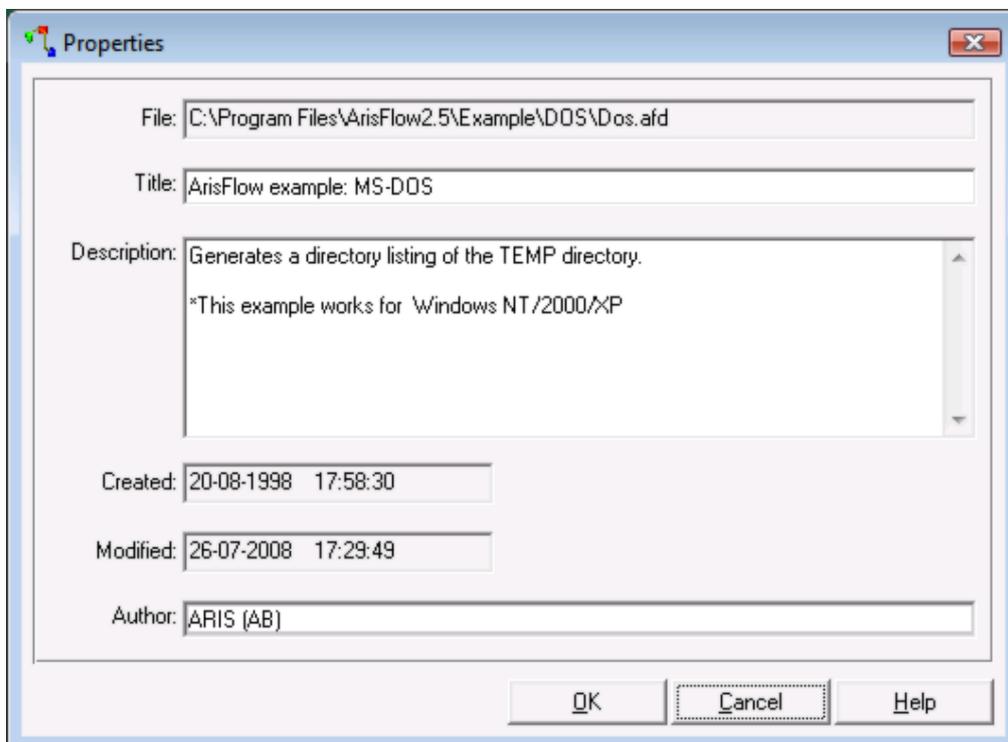
It is possible to recover a project by opening a project backup file (default extension ".~fd", properties can be set in the File Preferences dialog). When recovering a backup project file it is good practice to change its filename immediately, otherwise the project file and the project backup file may have the same name and .~fd extension.

The *File/Import* and *File/Export* menu options facilitate the exchange of definitions (directory aliases, action types and data types) between ArisFlow projects through ASCII text files. This is described in more detail in the Import / Export of Definitions section.

When a project is opened, ArisFlow checks environments, directory paths and possibly the status of actions and data, as described in the Opening an existing ArisFlow Project section.

Properties

When the [File/Properties](#) menu option is chosen, the following dialog appears:



Properties dialog.

This dialog allows the user to describe some meta-data about the project. The items whose values can be set are self-explanatory. ArisFlow sets three fields automatically:

- Full path of the project file;
- Created. The date/time when the project file was first saved under the full path name mentioned earlier;
- Modified. The date/time when the project file was last saved.

Project property values can be printed in two places:

- As a specification of all properties values by checking the *Properties* option in the Print dialog;
- On every printed page, in its header or footer. This is achieved by selecting certain property items in the [Header/Footer](#) tab in the Page Set-up dialog.

Drawing the Flowchart

In the flowchart the following functions can be performed:

- Create new actions, data, or connections;
- Select actions and/or data;
- Delete actions and/or data;
- Start an Action or Data dialog;
- Move actions, data, connections or grid lines;
- Move action names independently from the action;
- Zoom in or out;
- Scroll and pan through the flowchart.

Actions or data can be created if the corresponding Cursor is set after pressing the  or  button in the ArisFlow Toolbar. Actions or data are then created by pressing the left mouse-button in the flowchart. The action or data will then appear in the flowchart in that position as a rectangle (data) or a circle (action).

A connection is drawn between input flowchart object (data or action) and output object (action, respectively data). A connection is created if the *Connection* Cursor is set by pressing the  button on the Toolbar. Then first click the left mouse-button at the input object followed by clicking the left mouse-button at the output object. Alternatively a connection can also be created by starting the context menu (press the right-mouse button above the starting action or data) and selecting the *Start Connection* menu-option from the context menu. Connections should follow certain rules as described in the ArisFlow Flowchart section.

Actions and data can be selected as described in the Selection of Flowchart Elements section.

All actions and data selected are deleted by choosing the [Edit/Delete](#) menu option, by selecting the context menu *Delete* option or by simply pressing the *Delete* key. All the connections connected to any deleted data or actions will also be deleted from the flowchart. The *Undo* button will reinstate the actions, data and connections deleted during the last delete.

Connections cannot be deleted (nor selected) directly. A connection can only be deleted through the Action Script dialog of the action to which it is connected.

Action / Data dialog windows enable the user to edit actions/data. This is usually accomplished by double clicking on the action/data object in the flowchart. An alternative way to start an action or data dialog is to have the Cursor in the Select mode (, press Shift-S). Point the cursor at the action or data to be edited and then choose the *Edit* menu's *Action/Data details* option or simply press F4. Through the context menu it is also possible to start the action or data dialog, or even to go straight to the Action Script dialog.

Actions and data can be moved. Select the actions and data, position the mouse on one of the selected actions or data, press the left mouse-button and move the mouse with the left mouse button pressed down. All selected actions and data will move according to the mouse movement. When the user releases the left mouse button, the movement is finished. All connections with any actions and/or data, which have been moved, are now moved automatically.

For positioning actions and data the Alignment and Spacing Functions are very useful.

The action name is originally positioned to the right of the action circle. By clicking the right-mouse button on the action name or on the action circle and selecting the context menu option *Move Action Name* it is possible to position the action name independently from the action circle. The position of the action name relative to the circle is maintained when moving the action or zooming in the flowchart. The context menu option *Reset Action Name* returns the action name to its original position to the right of the action circle.

The Print Page section describes how to move page grid lines.

The Zooming section explains how to zoom in and out.

Moving the scrollbars performs scrolling through the flowchart, if they are attached to the flowchart display. If no horizontal or vertical scrollbar is attached, the horizontal, respectively the vertical size of the flowchart is less than the screen size, making scrolling unnecessary.

Panning is an alternative for using the scrollbars. The flowchart display follows the movement of the mouse cursor. First press the panning button  on the toolbar (or press the F8 hot key), position the mouse cursor  in the flowchart and move it with left button pressed down. The panning movement stops when the flowchart limits are reached.

Aligning and Spacing Functions

Both alignment and spacing functions are part of the [Edit menu](#). Alignment functions position the selected actions and data in the flowchart such that the centres of these flowchart objects are aligned horizontally or vertically. When drawing the flowchart this can be very useful where actions and data are connected: the connection lines will appear as perfectly straight lines in the flowchart. ArisFlow has the following alignment functions:

- Left Align: Vertical alignment of all selected objects. The horizontal positions of these objects becomes the horizontal position of the leftmost selected object;
- Right Align: Vertical alignment of all selected objects. The horizontal positions of these objects becomes the horizontal position of the rightmost selected object;
- Top Align: Horizontal alignment of all selected objects. The vertical positions of these objects becomes the vertical position of the topmost selected object;
- Bottom Align: Horizontal alignment of all selected objects. The vertical positions of these objects becomes the vertical position of the bottommost selected object.

Spacing functions space the centres of the selected actions and data evenly in the flowchart. There are two spacing functions:

- Horizontal Spacing: The leftmost selected object remains the leftmost selected object. The rightmost selected object remains the rightmost selected object. Objects in between are spaced evenly according to their initial horizontal positions;
- Vertical Spacing: The topmost selected object remains the topmost selected object. The bottommost selected object remains the bottommost selected object. Objects in between

are spaced evenly according to their initial vertical positions.

Alignment and spacing functions work on selected flowchart objects. Useful selection tips and tricks (in particular selecting elements with Control and/or Shift keys pressed down) are described in the [Selection of Flowchart Elements](#) section.

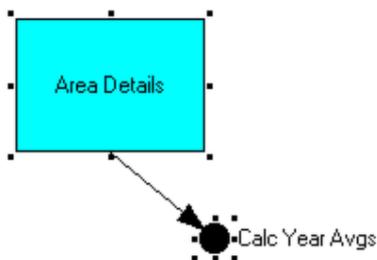
The following remarks are made:

- Alignment functions require at least two flowchart objects to be selected, otherwise alignment is not possible and the alignment menu options are disabled;
- Spacing functions require at least three flowchart objects to be selected, otherwise spacing is not possible and the spacing menu options are disabled;
- Alignment is done relative to the object with the centre of all selected objects, so that all connections form a straight line;
- Spacing is also done relative to the centre of objects.

Selection of Flowchart Elements

In the flowchart only actions and data can be selected. Selected elements are deleted by pressing the *Delete* button or selecting the [Edit/Delete](#) menu option. Connections or page grid lines cannot be selected. Page grid lines cannot be deleted; they can only be moved and become redundant. Connections are deleted when the data or action connected by the connection is deleted or indirectly by removing the connected data in the Action Script Dialog.

Selected objects are marked by eight small squares showing in regular positions just outside the actions/data selected. The action name is also surrounded by these selection markers.



Selected flowchart objects.

To select data and/or actions, the ArisFlow Cursor should be in the *Select* mode ( cursor). There are three methods of selecting data/actions:

- Clicking the left mouse-button on the flowchart object to be selected. An action is also selected when clicking on the action name.
- Drawing a selection rectangle in the flowchart.
- Selecting the [Edit/Select All](#) menu option.

When clicking the left mouse-button on a flowchart object or its name that object is selected. If the control button is pressed, all objects directly connected to this object are selected as well.

Create a selection rectangle by positioning the cursor in the flowchart (not on an action or data object) and pressing down the left mouse button. When the mouse is moved with the left mouse-button pressed down, a selection rectangle is drawn in the flowchart. When the left mouse button is released all data/actions, of which the rectangle/circle is completely inside

the selection rectangle, are selected.

When the *shift* button is pressed during selection, all objects, which had already been selected, will remain selected. However when clicking data or actions, which are already selected, with the shift button pressed down, the selection of those particular flowchart objects is undone.

Selected actions and data can be:

- Edited by selecting the [Edit/Action/Data Details](#) menu option. The Action or Data dialog of the flowchart element last clicked at by the mouse cursor is started.
- Deleted from the flowchart by the [Edit/Delete](#) menu option and restored by the [Edit/Undo](#) menu option option, as described in the Delete and Undo section.
- Copied by the [Edit/Copy](#) menu option. A copy of these copied flowchart elements can be pasted into the flowchart by the [Edit/Paste](#) or [Edit/Paste Special](#) menu options, as described in the Copy and Paste section.

Delete and Undo

When the [Edit/Delete](#) menu option is selected all currently selected flowchart elements are deleted. Any connections involving any deleted elements are also deleted.

When the [Edit/Undo](#) menu option is selected all flowchart elements that were last deleted are restored into the flowchart. Deleted connections are restored if possible, that is if all integrity rules are still satisfied upon restoring these connections.

When restoring a connection:

- Data occurrences in action scripts are not restored if only the connected action or data was restored, i.e. the other connected data or flowchart was still in the flowchart before the undo took place.
- Data occurrences in action scripts are restored if both the connected action and the connected data were restored in the flowchart.

The name of an action or data may be altered if an action or data with the same name is already present in the flowchart.

Copy and Paste

When the [Edit/Copy](#) menu option is selected all currently selected flowchart elements (actions, data and connections) are copied.

When the [Edit/Paste](#) or the [Edit/Paste Special](#) menu option is selected a copy of all flowchart elements that were copied last time is pasted into the flowchart. The entry of the copied data is not copied, otherwise the copied data and data copy will have exactly the same path. Thus all data copies will become undefined.

The [Edit/Paste](#) menu option only pastes connections between pasted actions and data into the flowchart.

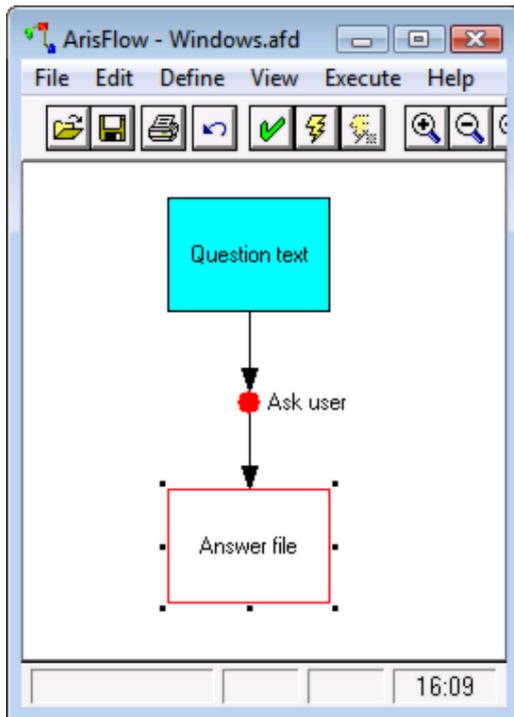
The [Edit/Paste Special](#) menu option also pastes connections between a pasted action or data and an action or data that was still in the flowchart before the paste took place. These

connections are pasted provided that the insertion of a connection does not violate any of the integrity rules for the flowchart.

Occurrences of pasted data will only occur in then connected Action's Script of both the Data and the Action were pasted into the flowchart.

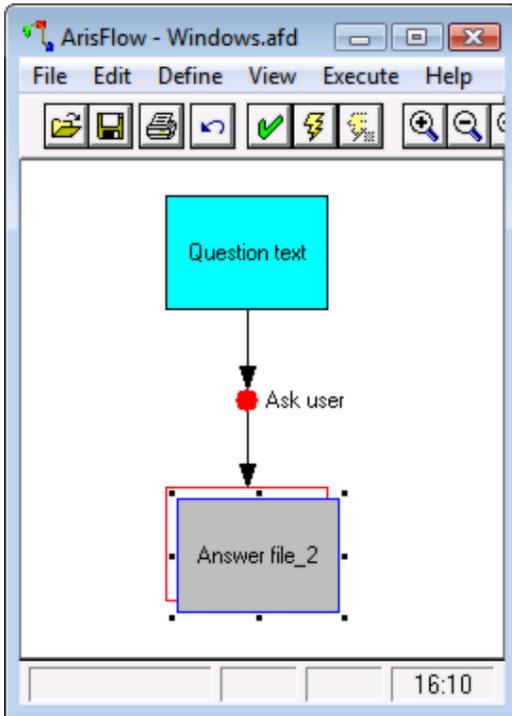
The name of an action or data may be altered if an action or data with the same name is already present in the flowchart.

The following example shows how the Copy and Paste/Paste Special menu options work. In the following situation the data "Answer file" is copied:



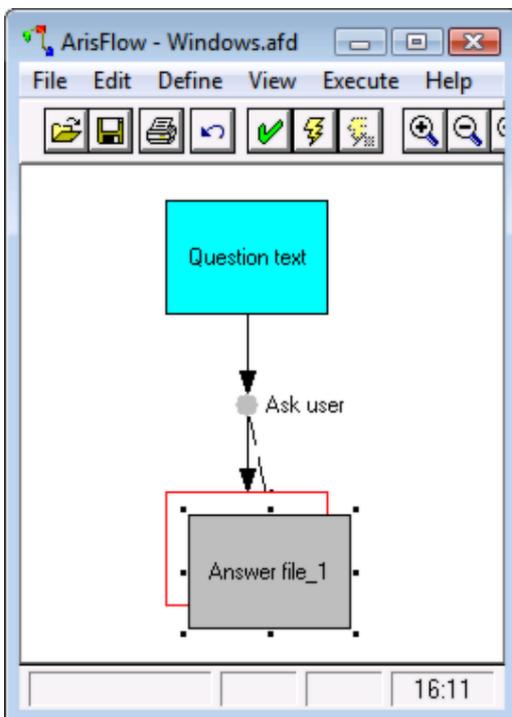
Flowchart where a selected data is copied.

The result after a paste using the the *Edit/Paste* menu option is:



Flowchart where selected data is pasted using *Edit/Paste*.

The result after a paste using the *Edit/Paste Special* menu options is:



Flowchart where selected data is pasted using *Edit/Paste Special*. A connection appears between the copied data (Answer file) and the action (that was not copied)

Zoom

The ArisFlow View menu contains the following zoom options:

- Zoom in;
- Zoom Out;
- Zoom Previous;
- Zoom 1: 1;
- Zoom All;
- Zoom Selected.

Toolbar buttons are defined for the *Zoom In* , *Zoom Out* , *Zoom Previous*  and *Zoom All*  menu options.

When selecting *Zoom in*, the ArisFlow Cursor is set at the *zoom in* state: . Each time the left mouse-button is clicked, a zoom in is performed in the flowchart. There are two possibilities:

- Click the left mouse-button in the flowchart. The flowchart display is enlarged by a factor 2. The position clicked at becomes the new centre of the screen display;
- Draw a rectangle with the left mouse-button kept down. After the left mouse-button is released, ArisFlow displays the portion of the flowchart that was enclosed by the rectangle.

When *Zoom Out* is selected, the ArisFlow cursor is set at the *zoom out* state: . Each time the left mouse-button is clicked in the flowchart, the portion of the flowchart displayed is doubled. The position clicked at becomes the new centre of the screen display.

Zoom Previous returns to the zoom state and flowchart origin position that was valid before the last zoom operation was performed.

Zoom 1:1 makes the flowchart appear with the same size as when printed on paper.

Zoom All adjusts the flowchart size to make it fit entirely onto the screen display.

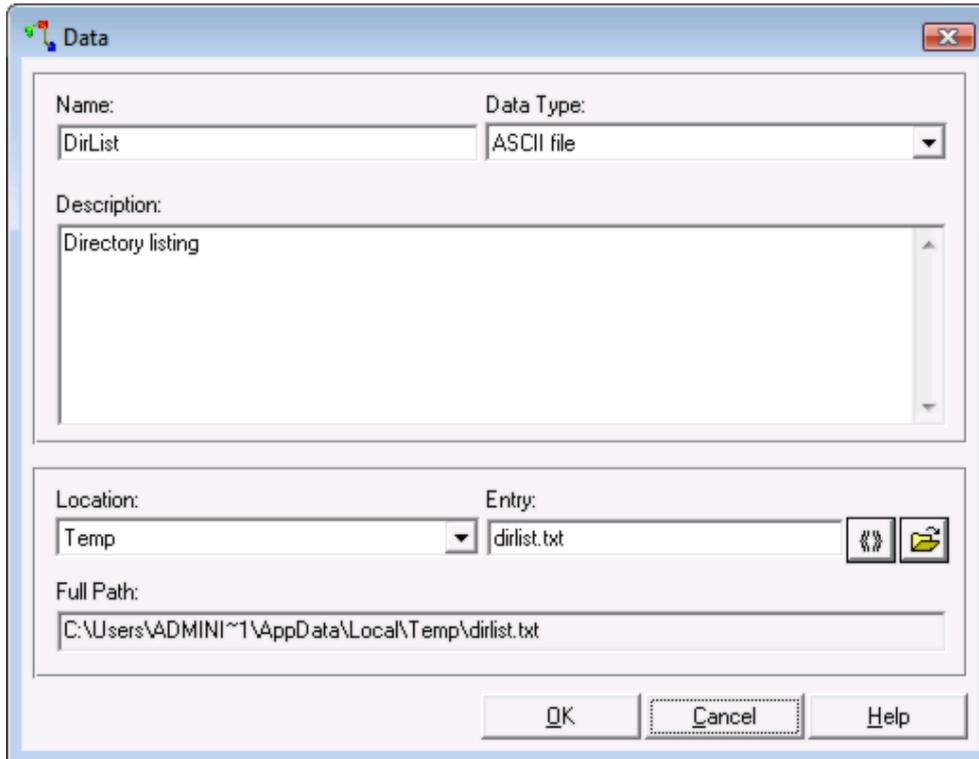
Zoom Selected will zoom on all actions and data, which are currently selected. This option is disabled if no flowchart element is currently selected.

There are two final remarks:

- ArisFlow will not display anything beyond the flowchart limits. Therefore when zooming out near the edge of the flowchart, it is possible that the centre of the flowchart display is not where the mouse was clicked in the flowchart before zooming out;
- The portion of the flowchart printed on a single page is not altered when zooming in or out, as the page grid lines also zoom in or out.

Data

A Data dialog is started by double clicking the data object in the Flowchart or by Selecting the data object and selecting the menu option [Edit/Action/Data Details](#) (which is under the F4 key).



Example of data dialog with data of the File base type

The data dialog asks the user to fill in:

- Logical name of the data used in the flowchart;
- One of the data types defined in the Data Type definitions;
- Description of the data. This box is for the user's convenience; any information about the data can be filled in here;
- Location and entry of the data.

Its location and its entry define data. The location is the directory where the data is located. The data type determines how the location is defined. The location is defined as either:

- A directory alias. This is the case for data types, which are files or directories or user-defined data-types where the location is stored as a directory alias. If a directory alias is assigned with not-existing directory path the data will be undefined.
- A string. This is the case for user-defined data-types where the location is not stored as a directory alias.

When changing the data type this may change whether the location is stored as directory alias or as a string. ArisFlow then automatically updates the matching location. When changing to a representation with directory aliases it is possible that no matching directory is present. In that case ArisFlow will create a new directory alias and ask the user for the name of that directory alias through a Directory Alias dialog.

The Full Path field displays the full path of the data, which can be used in any Action Script, which uses the data.

The *browse* button  helps the user search the location and entry of the data. For files this is a simple File Browser. If a default extension was defined in the File Data Type definition, the browser will search for files of that extension. It is, however, always possible to browse for any type of file. For data of directory base type a directory browser browses and returns directory paths. The last part of the directory becomes the directory's entry, the remainder

the directory location.

If location is stored as a directory alias this is always a location from the Directory Alias List. The browser may return a directory location, but this is converted into a directory alias. If the location returned does not belong to an existing alias, the user is asked to define a directory alias with that path.

When the browse button  is pressed for user-defined data, the browser defined in the User-Defined Data Type dialog is started. Upon completion of the browser program the location and entry written in the resulting temporary file are allocated in the respective fields. The User-defined Data section gives more specific information about user-defined data.

Pressing the pre-defined option selection buttons  makes it possible to insert variables in the entry and the user-defined data location.

A special category of data is the input=output data, described in a separate section. Input=output data is where two identical data are linked to the same action. Input=output data are mainly used where existing data (e.g. a database) is updated by a certain action.

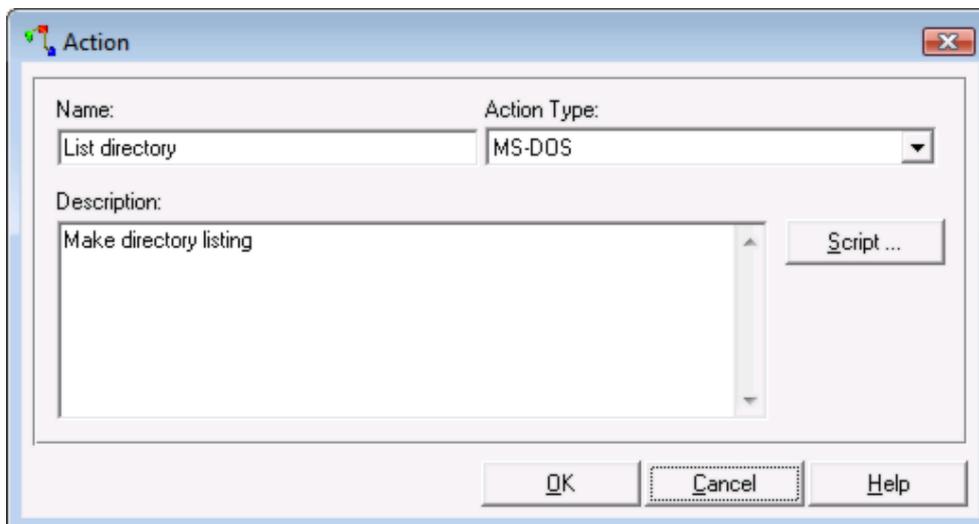
Data are set to a defined status (up-to-date or not-up-to-date) if the following requirements are met:

- A type has been assigned to the data;
- Location and entry have been assigned to the data;
- Project input-data should be present on the disk. For other data it should be possible to create them, i.e. the location should be present. This is described in detail in the State of Actions and Data sections.

Actions

An Action dialog is started:

- By double clicking the action object in the Flowchart;
- By Selecting the action object and selecting the menu option [Edit/Action/Data Details](#) (which is under the F4 key).



Example of action dialog.

The action dialog box asks the user to fill in:

- Logical name of the action, which is used in the flowchart;
- One of the action types defined in the Action Type definitions;
- Description of the action. This box is for the user's convenience; any information about the action can be filled in here.

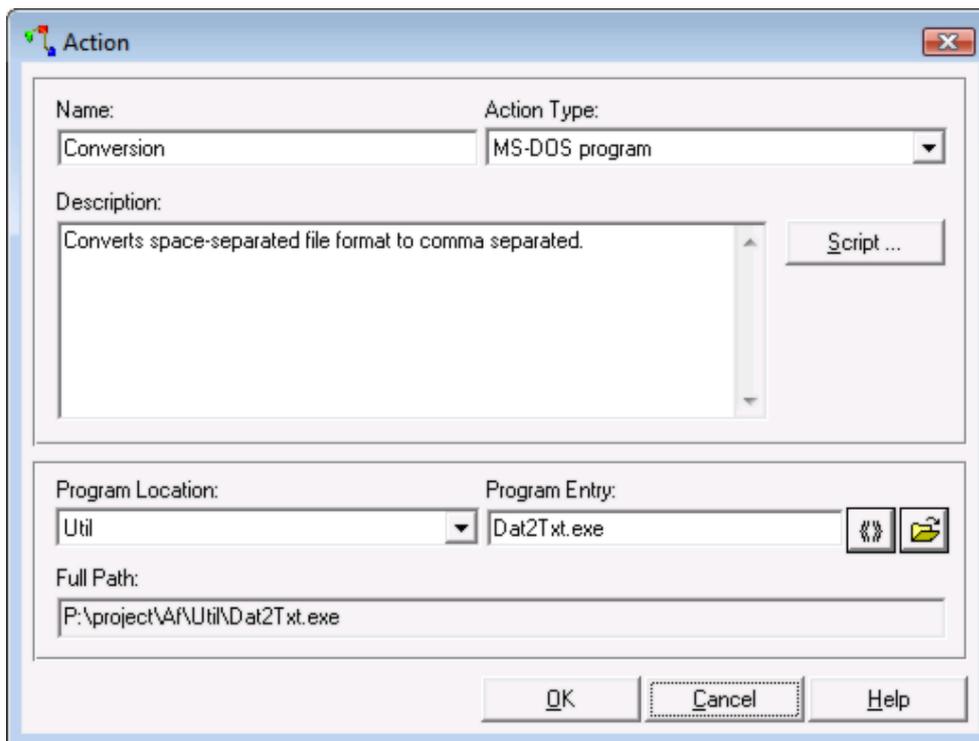
When the *Script* button is pressed the Action Script Dialog window is displayed. The Action Script dialog is the core of ArisFlow as it defines the commands that are to be executed by the action.

Actions are set to a defined status (up-to-date or not-up-to-date), if the following requirements are met:

- A type has been assigned to the action concerned;
- If the action is a program, an existing executable should be assigned to the location / entry fields;
- The action script has been defined properly. To define the action script the *Script* button must be pressed, which results in the appearance of the Action Script dialog.

The status of actions is described in more detail in the State of Actions and Data sections.

Some actions are defined as programs. An action is a program when the action type is of System base type with the *Program* option selected as the *Shell Execute As* option. In the program action window two extra fields appear:



Example of action dialog with action of the Program base type.

The program action's two extra fields define the program to be started when the action is executing. They are:

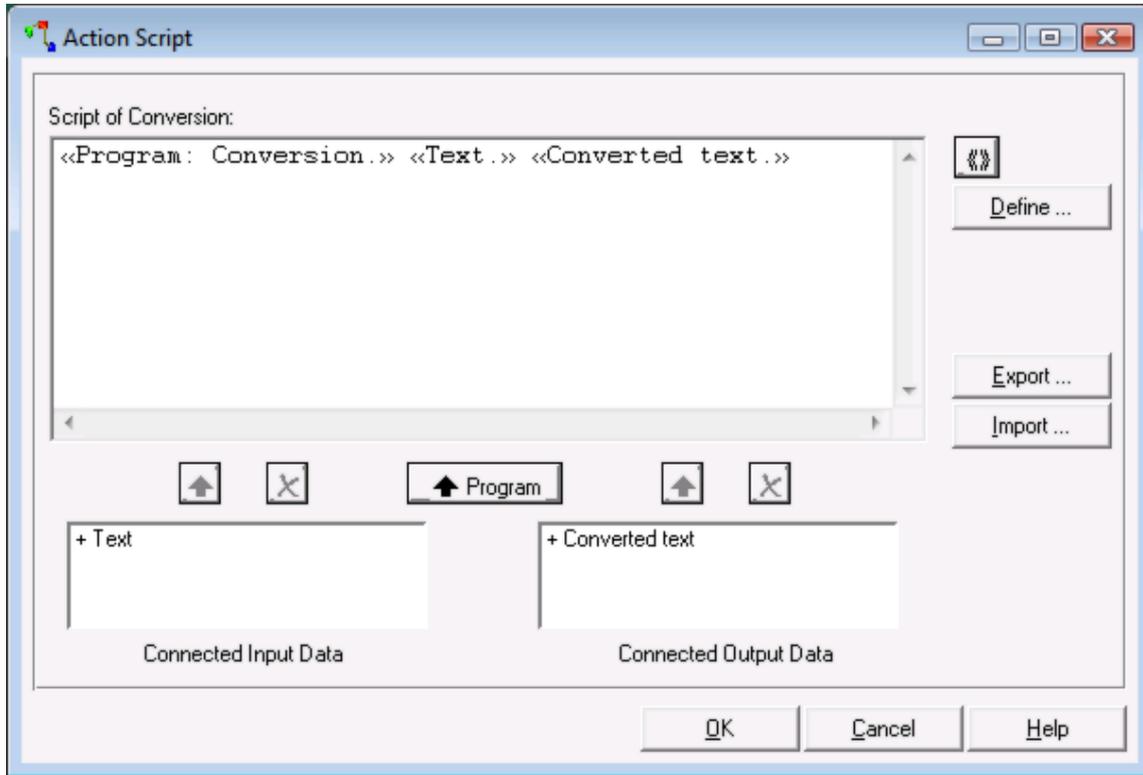
- Location. The directory alias of the directory where the action program is located;
- Entry. The name of the executable including its extension.

By pressing the *browse* button  it is possible to Browse the location and entry of the program file. Pressing the  button enables the insertion of variables in the program entry description.

Action Script

Action Script

Pressing the *Script* button in an Action dialog starts an Action Script dialog.



Example of action script window of program action.

The Action Script dialog defines the action script commands. These commands are executed when the action is executed. Therefore the action script is the core of the ArisFlow project. The action script dialog is sizeable.

The dialog contains the following features:

- The Action Script Edit Field;
- The Connected Input Data and Connected Output Data Listboxes;
- Buttons;
- Execution of the action script.

Action Script Edit Field

The action script is described in the Action script edit field. This field may contain multiple lines. An "_" (underscore) at the first position of a line indicates that a line is a continuation of the previous line.

The script may contain the following script elements, which are always enclosed between « and » brackets:

- Variables, which is displayed in the variable notation;
- Directory aliases, which are displayed in the directory alias notation;
- Control files, which are always displayed as «*control file*»;
- Program script elements; which are displayed as «*Program: ActionName*», where ActionName is the name of the action;
- Data script elements, which are displayed as «*DataName*», where DataName is the name of the input or output data.

Program script elements can only be part of the script if the action is a program, which depends upon the definition of the Action type.

How the script elements are inserted into the script is described under the Connected Input Data and Connected Output Data Listboxes and Buttons sections.

The Special Key Options chapter describes a number of shortcut keys useful when editing fields with script elements, such as the script field.

The action script is defined when the following requirements are met:

- The script edit field is not blank, but contains at least some text or a script element;
- All input and output data appear at least once as data script elements in the edit field. This implies that all the data in the input data and output data lists should have been labelled with a "+" sign;
- For program actions the script contains at least one program script element.

Actions can only become defined if the script has been defined. When any data in either data list remains unused, the user has two options to render the action script defined:

- Remove the data connection;
- Include the data in a commentary statement, which has no practical effect on the action execution.

Connected Input Data and Connected Output Data Listboxes

The Connected Input Data listbox lists are all data that are input to the Action Script. In the flowchart connected input data are shown by a connection from that input data to the action.

The Connected Output Data listbox lists are all data that are output of the Action Script. In the flowchart connected output data are shown by a connection from the action to that output data.

To make the action defined it is required that all these connected data occur at least once as a data script element in the Action Script Field. Inserting these data into the action script can be done by either:

- Pushing the  button. The selected data in the listbox underneath the button will be inserted into the Action Script Field at the cursor position;
- Double clicking on the data in that listbox.

Data which appear in the script field are labelled with a "+" sign in front of them. In the flowchart the corresponding connection is represented by a solid line. Data which do not yet appear in the script have a "-" sign in front of them. In the flowchart the corresponding connection is represented by a broken line.

A special case are input=output data. These are data connected to the same action and which have the same data type and the same actual path. One is input data, the other is output data

to the action. Special rules apply to the status of these data, as described in the [input=output data](#) section. But also, only one of these data has to be included in the Action Script Field. When that is achieved both data are preceded by a "+" sign in the listbox.

It is not possible to insert data directly into the listbox. If a data should be in either listbox (and in the Action Script Field) this can be done by drawing a connection between that data and the action in the flowchart.

Data can be removed from the listbox by pressing the  button. This also removes all occurrences of the data from the Action Script Field and removes the corresponding connection between the data and action in the flowchart.

Action Script Buttons

The Action Script dialog contains the following buttons (apart from *OK*, *Cancel* and *Help*):

- . Inserts a control file, directory alias or variable script element into the Action Script Edit Field. The script element is selected through the Pre-defined Option Selection dialog;
- *Define*. Starts the Data Script Element Type dialog, allowing the user to assign a type to the data script element which is currently under or directly left of the cursor position in the script field;
- *Import*. An ASCII script file can be imported into the script field at the cursor position. Script elements and directory aliases, which are recognised by the script, are left intact, other script elements and aliases are ignored. The import file name is nominated by the user and has default ".scr" extension;
- *Export*. Writes the script (including script elements and directory aliases) to an ASCII file. The export file name is nominated by the user and has default ".scr" extension;
- . Inserts the currently selected data script element from the Connected Input or Connected Output Data Listbox below the button into the Action Script Edit Field;
- . Deletes the currently selected data script element in the Connected Input or Connected Output Data Listbox below the button:
 - From that Input or Output Data Listbox;
 - Deletes every occurrence of that data script element from the Action Script Edit Field;
 - The data script element represents a connection in the flowchart. As a consequence this connection is also deleted from the flowchart. At this moment this actually is the only mechanism for removing connections from the flowchart.
- . This button only occurs when the action is a program action, i.e. the [Action Type](#) is represents a program. Pushing this button inserts a program script element into the Action Script Element Field at the current cursor position.

Execution of the Action Script

When the script is executed the script elements are replaced by their actual values:

- A [variable](#) is replaced by its value;
- A [directory alias](#) is replaced by the actual path of the directory alias;
- A control file script element is replaced by an actual path generated for the control file. For

each control file script element a unique, not yet existing, file path with extension ".\$\$\$" is generated on the temporary directory;

- A program script element is replaced by the actual path of the action, to which this action script belongs;
- A data script element is replaced by the actual path of the data.

Tooltips show the actual values for every script element, e.g. the full path of a directory alias.

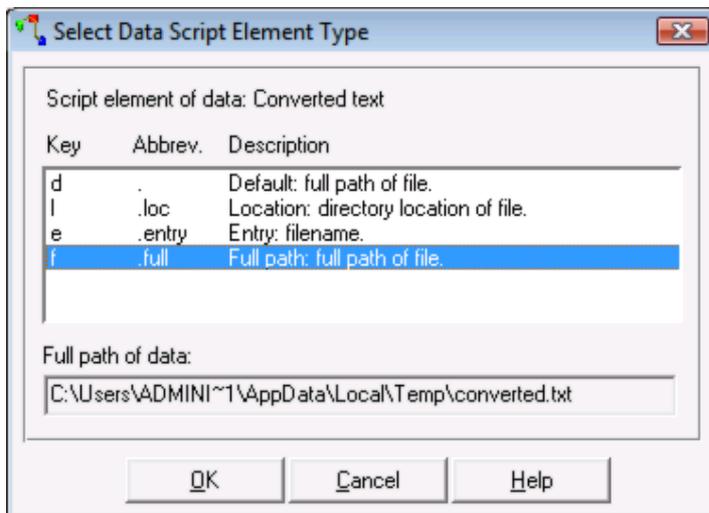
Each line in the action script is passed as a separate line during execution. How these lines are executed is determined by the *Action Script* section in the System or DDE dialog of the [action type](#), in particular:

- Whether each line is passed as a separate command or passed in a script file;
- Whether or when script elements are enclosed between quotes;
- Whether server path directories are substituted instead of the local path definitions for [directory aliases](#) in directory alias, program and data script elements.

Finally blank lines are passed as separate commands or script lines.

Data Script Element Types

The Data Script Element Type dialog defines the type of the data script element under the cursor in the action script dialog. Pressing the *Define* button or the Ctrl-D key combination in the Action Script dialog starts the Data Script Element Type dialog:



Selection of data script element type.

A single line from the list box could be selected. The type defined in that line is inserted into the data script element in the Action Script dialog. A data script element type is selected and returned after:

- Selecting the line and pressing OK; or
- Double clicking the line; or
- Pressing the corresponding key for quick selection.

Each line contains a data script element type and has the following fields:

- Key. A simple character for quick selection of that type;
- Abbreviation. The abbreviation of the type displayed for the data script element in the

Action Script dialog;

- Description. Describes whether the entry, location or full path is substituted for the data script element in the execution of the action script when that type is selected.

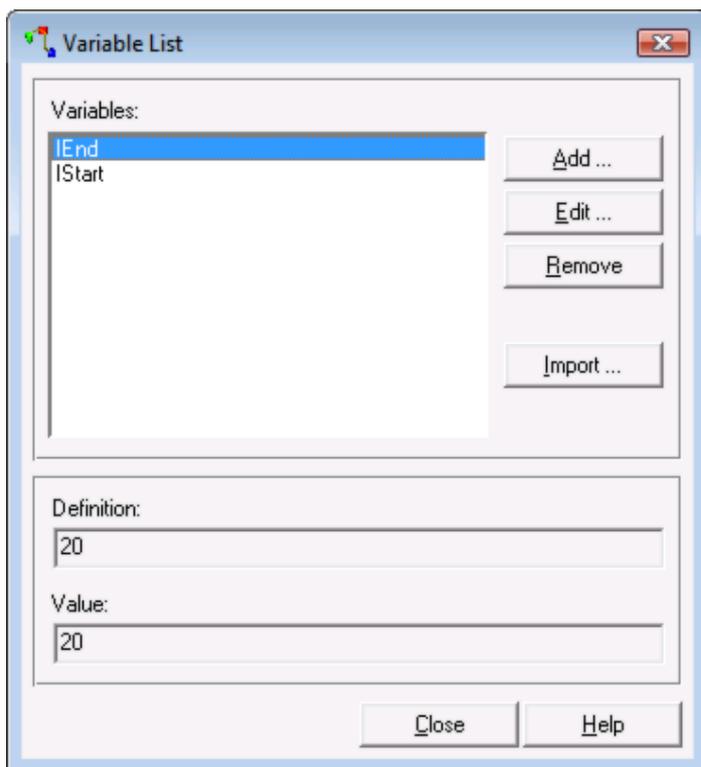
There are three or four possible script element types:

- ".". A simple dot. This is the default type, which is the same as the ".full" type;
- ".entry". The data in the script are to be replaced by the entry of the data only;
- ".loc". The data in the script are to be replaced by the location of the data only;
- ".full". The data in the script are to be replaced by the full path of the data. This is the location + entry, separated by a separator. Where [directory aliases](#) are used in the location definition the separator is a forward (/) or backward (\) slash, depending upon the directory path definition. User-defined data with location as a string use the separator defined in the user-defined datatype dialog.

Variables

Variables List

When the [Define/Variables](#) menu option is chosen, the following dialog window appears:



Variables list dialog displaying all variables defined in the project.

The list displays the names of all variables defined for the project. When selecting a variable the boxes in the lower part of the dialog display.

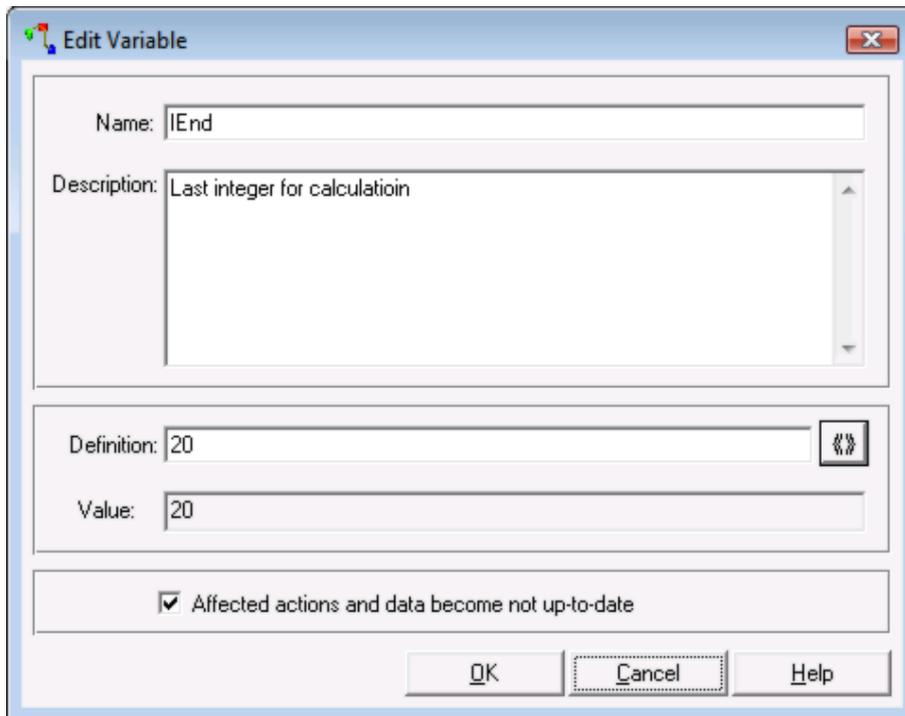
- The definition of the variable, including other variables that may make up the variable definition;
- The actual variable value, where the values of other variables, that define a variable, is substituted in the variable's value.

This dialog has three special buttons:

- **Add.** A new variable can be created and inserted into the variables list;
- **Edit.** The name or value of an existing [variable](#) can be changed.
- **Remove.** The selected variable is removed from the project. If the selected variable cannot be removed, ArisFlow lists all the occurrences of the variable in the project, which prevent its removal. This feature can be very useful in tracing the occurrences of a certain variable.
- **Import.** Variables can be selected from other ArisFlow project files and copied into this project. Non-existing variable definitions occurring in the imported variable definitions may be copied into this project as well.

Variables

When *Add* or *Edit* is pressed in the Variables List dialog the following dialog screen appears:



Variable dialog for defining the "Year" variable.

The Variable dialog has four major fields:

- The *Name* field is the logical name by which this variable is known in the project. Names should be unique. Blanks and most other characters are allowed, except for the « and » characters;
- In the *Description* field the user can enter information about the variable, its usage, etc.;
- In the *Definition* field the value of the variable is defined. This definition may contain other variables and environments, which can be inserted by pressing the  button;
- The *Value* field displays the actual value of the variable, where all the variables and environments included in the *Definition* field are substituted by their actual values. Any change in the *Definition* field is reflected in the *Value* field. The *Value* field cannot be edited.

The *Value* field in the variable definition may include other variables. However a "circle" may not occur, that is these other variables may not somehow depend upon the defined variable. If that happens ArisFlow displays an error message.

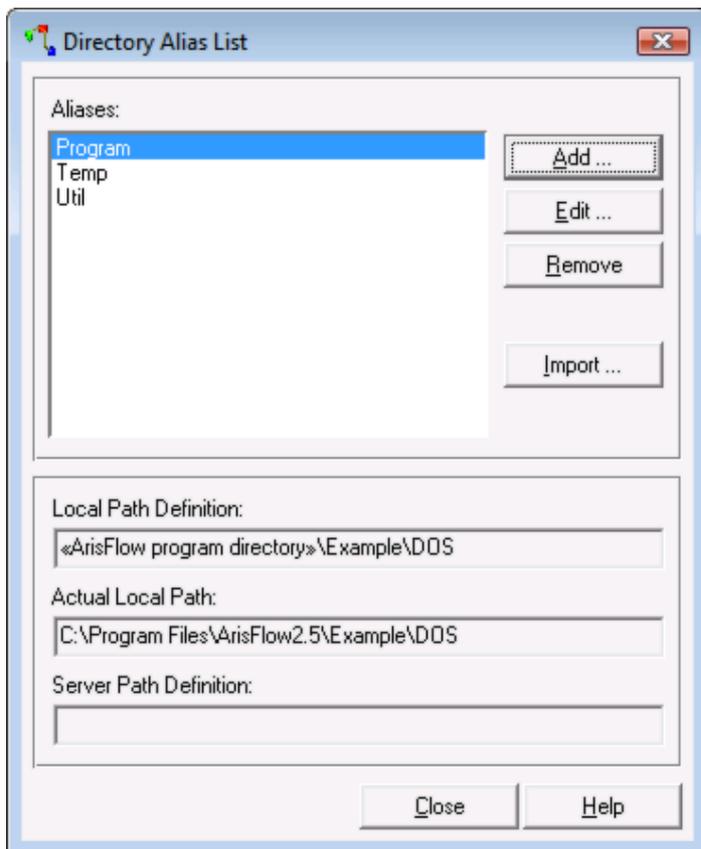
When the checkbox option: "Affected actions and data become not-up-to-date" is checked, and the variable's actual value is changed, all data and actions, which use this variable in their location or entry definitions, become not-up-to-date. Also actions, of which the action script or action type uses the variable, become not-up-to-date. Finally, because [directory alias](#) local or server paths using the variable may also change, this may set data and actions, which use that [directory alias](#), not-up-to-date as well.

When the "Affected actions and data become not-up-to-date" option is not checked data and actions are just relocated and keep their statuses, unless the affected data are not present anymore or have become present. Actions, of which the variable occurs in their action scripts or action types, also keep their statuses.

Directory Aliases

Directory Alias List

When the *Define/Directory Aliases* menu option is chosen, the following dialog window appears:



Directory alias list dialog displaying all directory aliases defined in the project.

The list displays the names of all directory aliases defined for the project. When selecting a directory alias three characteristics of it are displayed:

- Its local path definition;
- The physical location of the local path, which is called the actual path. If this location does not exist nothing a message displayed in this field;
- The server path definition.

This dialog screen has three special buttons:

- **Add.** A new directory alias can be created and inserted into the directory alias list;
- **Edit.** Details of the selected directory alias can be altered. Double clicking a directory alias in the alias list also starts the Edit dialog;
- **Remove.** The selected directory alias is removed from the project. If the selected directory cannot be removed, ArisFlow lists all the occurrences of the directory alias in the project, which prevent its removal. This feature can be very useful in tracing the occurrences of a certain directory alias.
- **Import.** Directory aliases can be selected from other ArisFlow project files and copied into this project. Non-existing variable and root directory alias definitions occurring in the imported directory alias definitions may be copied into this project as well.

Directory Alias

When *Add* or *Edit* is pressed in the Directory Alias List dialog the following dialog screen will appear:

Directory alias dialog for defining the "Temp" directory alias.

There are four edit fields:

- The *Alias* is the logical name by which this directory is known in the project. A name should be filled in. Any name will do, as long as it is unique. Blanks and most other characters are allowed, except for the « and » characters;
- In the *Description* field the user can enter information about the directory alias, its usage, etc. Descriptions have no impact in ArisFlow;
- The *Local path* defines the directory alias path on the computer from which ArisFlow is executed;
- The *Server path* defines the path of the directory alias when actions are executed from an

alternative server.

Local path and server path definitions may include:

- **Root directory.** This is another directory alias, which occurs at the first position of the path definition. The root directory should not be derived from the directory that uses it, otherwise a circle occurs. Root directories are enclosed between "«Dir: " and "»" markers;
- **Environments.** Environments may be inserted anywhere in the path definition. Environment variables are particularly useful for utilities and program directories, windows and temporary paths, etc. These directories may have different locations, but the same environment name on different systems. Defining these directories as environment enables using an ArisFlow project on different systems, without having to change directory definitions. Environments are written in environment notation.
- **Variables,** which may occur anywhere in the path definition. A variable is just a string, which is substituted wherever it occurs. Variables are enclosed between "«Var: " and "»" markers.

The local path may also include two pre-defined directories at its first (root) position:

- «ArisFlow program directory». This is the parent directory of the directory where the ArisFlow executable is positioned and thus the base directory for ArisFlow. It is a very useful directory for many ArisFlow examples;
- «current project directory». This is the directory where the current flowchart project file is positioned. In a new project this directory is the same as the «ArisFlow program directory», until the project has been saved in a new directory. This directory is useful in defining project-related directories.

Root directories, variables and environments and pre-defined directories can be inserted

through a Pre-defined Option Selection dialog by pressing the  button or by pressing the Ctrl-O button as described in the Special Key Options chapter. Root directories can only be inserted at the first position of the local path definition or server path definitions. If the cursor is not at the first position only variables and environments can be inserted. Pre-defined directories can only be inserted at the first position of the local path definition.

The  button enables browsing of the local path through a directory browser. The browsing starts at the currently filled in local directory. After the new directory was browsed its definition is assigned to the *local path definition* field, leaving as many root directory, environment and variable elements intact as is possible.

The Actual path is derived from the local path: the paths of script elements (environment, variable and root directories) are then substituted in the local path definition. Directory separators (\) may be a problem. If a directory separator occurs both at the beginning or end of a script element and at the path definition where the script element is inserted ArisFlow removes one directory separator. However when two directory separators occur in the same script element or in the normal path definition these were obviously intended to be there, so in that case ArisFlow will not remove anything.

ArisFlow does not insert any directory separators (\), so the user has to ensure that these separators are actually present in the path definition.

If the actual path is not physically present, ArisFlow will give a warning when the directory alias dialog is closed. The directory alias is still accepted, but any flowchart element using the directory will be marked as undefined. When starting a project with non-existing directories the user may define these directories through the Directory Alias Repair dialog.

Actions can be executed on remote systems called servers. When executing such an action the Action Script should contain directory locations that are known on this remote server. For each directory alias such a location can be defined through its Server path.

Server paths are only required when the *Use Server Directories* option is checked in the action type's *System* dialog for the directory aliases used by data, directory script elements, control files or script files connected to the remotely executed action. Local paths are always required when checking the status of data, and are therefore compulsory for every directory alias.

Very occasionally three paths may be involved in the definition of a directory alias. Suppose a project is executed on a Windows machine. Data are generated by an action running on a UNIX machine and are input to an action running on a VMS machine. This situation is solved by creating two data objects, which represent the same data (draw these close together in the flowchart), and two directory aliases, which represent the same directory. The output data object is defined using the directory alias with the UNIX server path. The other data object is defined using the directory alias with the VMS server path. Obviously both directory aliases have the same (Windows) local path, used for checking the status of the data from the local machine. To ensure the correct execution order a dummy action (system type, script in comment) is connected from the UNIX and to the VMS data.

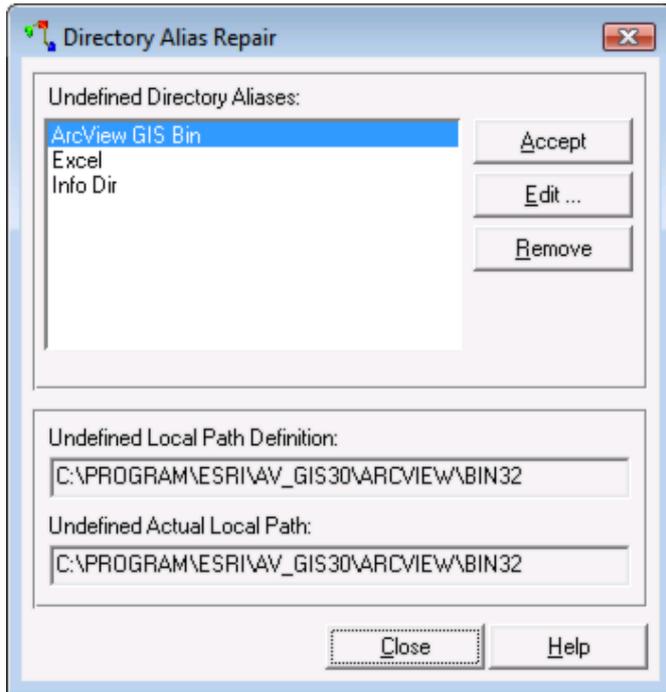
The checkbox option: "Affected actions and data become not-up-to-date" is not part of the directory alias definition, but indicates what should happen when the directory alias path is changed. When the directory local path was changed and this checkbox is checked, the data and actions that use this directory alias in their location or entry definitions, become not-up-to-date. This also applies where the directory alias is used in the action script or the action type definition (System or DDE). When the server path is changed only actions and data connected to actions, that use server directories, could become not-up-to-date.

When the "Affected actions and data become not-up-to-date" option is not checked, data and actions are just relocated. There are three major possibilities:

- Data or program actions that were up-to-date and are now located at another existing location. These data and actions remain up-to-date even when they now have different time, size or checksum characteristics;
- Data or program actions that were not present before but are now located at an existing location. These data and actions become not-up-to-date.
- Data or program actions that are now located at a non-existing location. Data, which are output of an action and of which the new directory location does exist, become not up-to-date. Project input data and actions, as well as data of which the new directory location does not exist, become undefined.

Directory Alias Repair

When a project is started through the [File/Open](#) menu option, ArisFlow will check all directory definitions first. If any directory's local path is not present on the disk, ArisFlow asks the user whether these directories should be defined. The directory alias repair dialog appears:



Directory aliases repair dialog displaying aliases of currently not existing directories.

For every local path to be repaired the user has the following three options:

- **Accept.** Accept the local path definition as it is. The directory becomes undefined and every flowchart object using the directory also becomes undefined;
- **Edit.** A [Directory Alias](#) dialog is opened to enable the user to set a different directory path for the directory alias;
- **Remove.** Removes the directory alias definition from the project.

The *Remove* button is only enabled if the undefined directory is not used anywhere in the project. In that case there is no harm in removing the directory alias definition. Just accepting that definition has no consequences either, simply because it is of no relevance to the project. If the *Remove* button is disabled, the directory alias is used in the project. In that case just accepting that alias has consequences for other parts of the project.

The directory alias repair dialog can be closed anytime, which accepts all remaining undefined directory aliases.

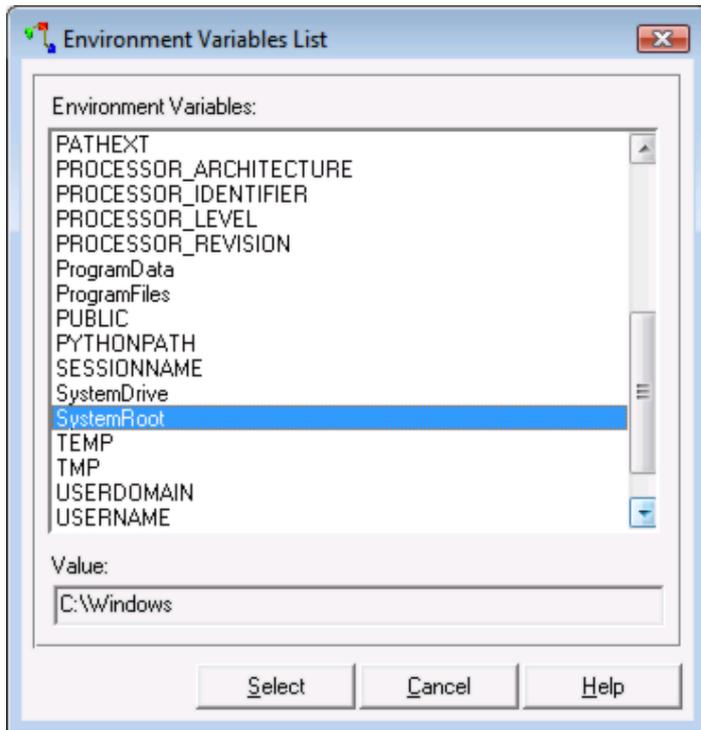
Environments

Environment List

Environments can be inserted into [variables](#) and [directory alias](#) local path definitions. Environments are particularly useful when ArisFlow projects are executed on different computers, with certain programs (mainly utilities) located on different directories. Often these directories are defined by the same environment variable, making this environment variable very useful in the directory alias definition.

Pressing the  button for the [Directory Alias](#) local path definition edit field or the [Variable](#) value field starts a Pre-defined option dialog, from which the «environment» option can be

selected. This leads to the environment selection dialog:

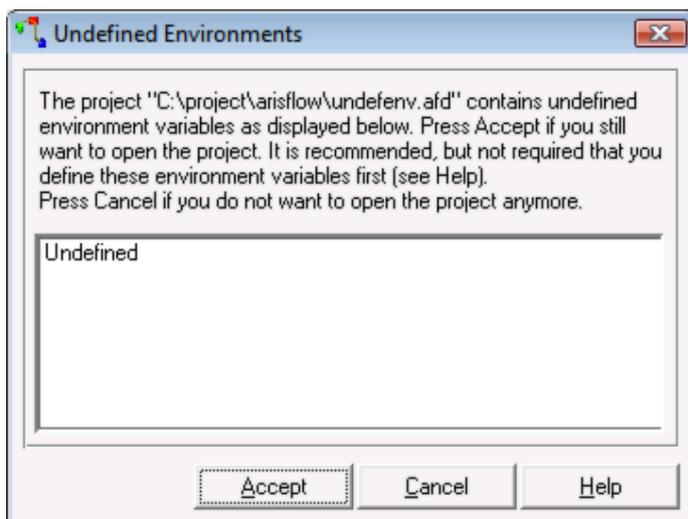


Environment selection dialog.

The environment selection dialog lists all the environments currently defined on the user's system. Selecting any environment variable will insert it into the local path definition of the directory alias. The environment value is substituted for every occurrence of the directory alias local path.

Undefined Environments

Environments used by ArisFlow should exist. When opening an ArisFlow project, which includes non-existing environments, the Undefined Environments dialog is displayed:



Undefined environment variables dialog displaying all the variables that have not been defined.

It is recommended that the user sets the undefined environment variables before continuing with

the project file, as described in the *Opening an existing ArisFlow Project* section. Under Windows an environment variable can be set by selecting *Settings /Control panel* menu option from the *Windows Start* menu, and then double clicking on the *System* icon and selecting the *Environment* tab.

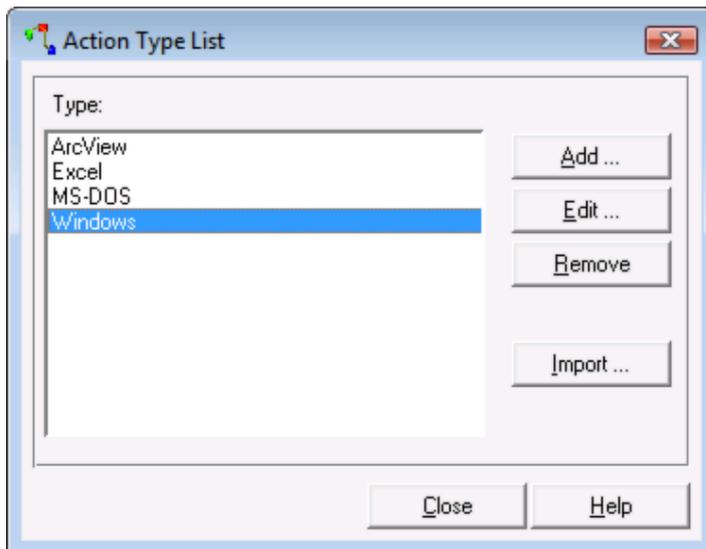
It is possible to push the *Accept* button and work with a project containing undefined environments. However this may lead to error messages where the undefined environment is involved. As environment variables can only occur in the definitions of [Variables](#) and [Directory Aliases](#) it is not difficult to manually remove every occurrence of the undefined environment from the project.

Action Types

Action Types

Action Type List

When the *Define/Action Types* menu option is chosen, the following dialog window appears:



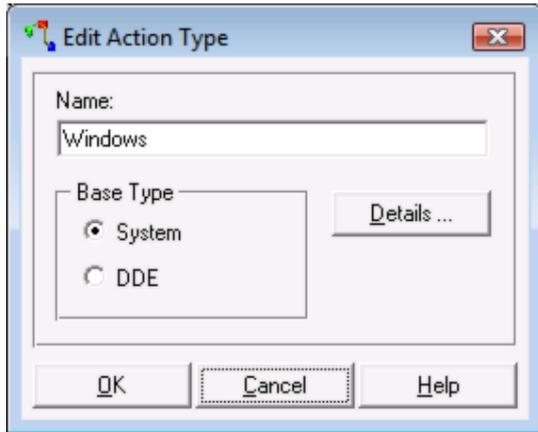
Action type list dialog displaying the names of all action types defined for the project.

This dialog screen has three special buttons:

- **Add.** A new action type can be created and inserted into the action types list;
- **Edit.** Parameters of the selected action type can be changed. Double clicking an action type in the action types list has the same effect;
- **Remove.** The selected action type is removed from the project. If the selected action type cannot be removed ArisFlow gives a message listing the actions that use the action type. This feature can be very useful in tracing actions, which are of a certain action type;
- **Import.** Action types can be selected from other ArisFlow project files and copied into this project. Non-existing variable and directory alias definitions occurring in the imported actiontypes may be copied into this project as well.

Action Type

When adding or editing an action type in the Action Type List dialog, the following dialog screen will appear:



Action type dialog for defining an action type.

The Name is the logical name by which the action type is known in the Action dialog. Any name will do, as long as it is unique. Blanks are allowed in the action type name.

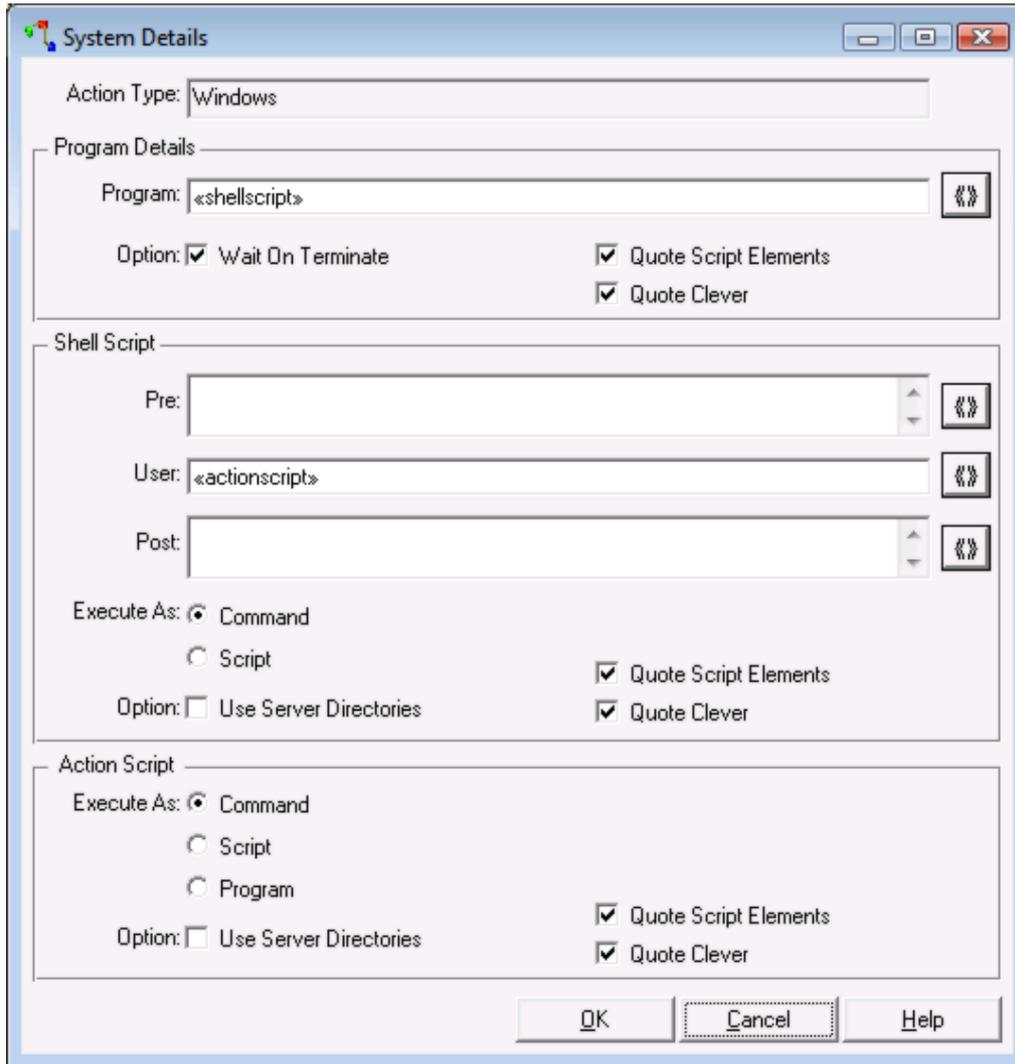
The *Base Type* has two possibilities:

- System. Actions of this action type are executed with system commands;
- DDE (Dynamic Data Exchange). Actions of this action type are executed in a different program using the client-server DDE protocol.

Further details of the action type can be filled in when the *Details* button is pressed. Pressing the *Details* button starts the System Details dialog if the "System" base type has been selected and the DDE Details dialog if the "DDE" base type has been nominated.

System Details

The System Details dialog is started by selecting system base type in the [Action Type](#) dialog and pressing the *Details* button.



Windows: a straightforward system action type definition.

The System Details window is divided into three levels:

- Program details;
- Shell script;
- Action script.

System action types pass commands to the operating system. The program details section describe how the system is accessed. It has three components:

- The Program Edit Field. Command lines are provided as defined by the shell script and actions scripts. These commands are embedded in the Program template (e.g. "cmd.com" in the DOS example below), where they are substituted for the «shell script», and are passed to the system;
- The "Quote Script Elements and Quote Clever" options. For the Program section both options are usually switched on, indicating that double quotes are inserted automatically around script elements;
- The "Wait On Terminate" option. When this option is set ArisFlow will only continue after the completion of the execution of a command has been acknowledged by the executing system.

The «shell script» option in the Program field is required.

The Shell Script consist of:

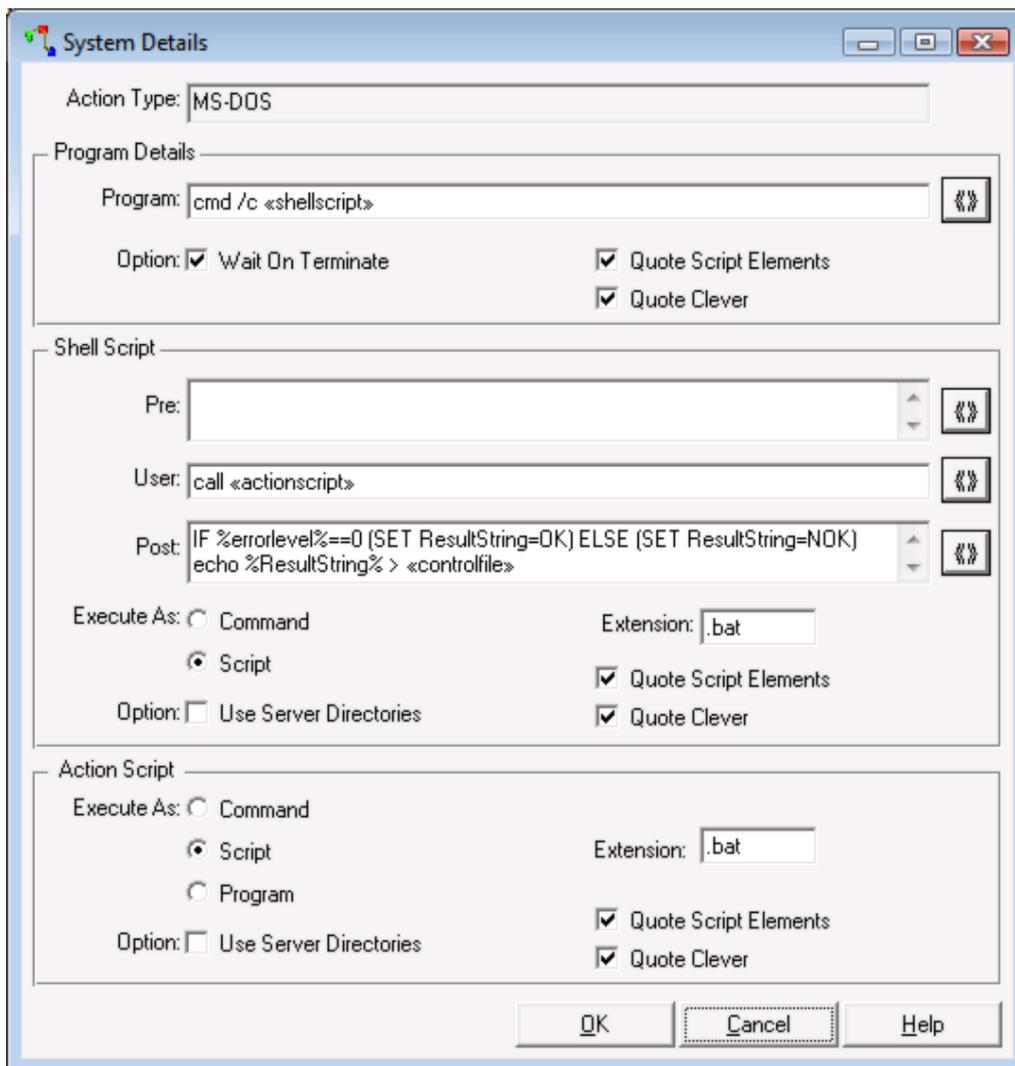
- Edit fields: Pre, User and Post fields;
- Options: Execute As, Use Server Directories and possibly Extension, and Quote Script Elements and Quote Clever.

The «action script» option in the User field is required.

The Action Script consist of:

- Options: Execute As, Use Server Directories and possibly Extension, and Quote Script Elements and Quote Clever.

An MS-DOS action type definition is a second example of a system details dialog:



Definition of MS-DOS action type.

This is a common system dialog. Cmd.exe executes the command. Upon completion the DOS box is closed ("/c" switch), which is the termination for which the execution waits. If the "/k" option is passed to "cmd.exe" instead of "/c", the DOS-box has to be closed manually. This can be useful for checking the commands that have actually been executed by an action of the DOS type.

Under Windows 95 the "dosprmt.tif" is the program executing the command used instead of "cmd.exe". This program is present in the util (Utilities) directory. Windows 95 users will find

the correct DOS action type implementation in the Examples\DOS directory.

When an action of this DOS type is executed, two temporary files of ".bat" extension are created. The action script file contains the contents of the action script. The shell script file contains two lines:

- call «action script». The filename of the action script file is substituted for the «action script». The (DOS specific) "call" is required here, otherwise the post command will not be executed;
- echo OK > «control file». When this line is reached, the action script has been executed and "OK" is written to the control file. A temporary filename of a not yet existing file is substituted for the «control file». ArisFlow generates an error if it does not detect this control file or if the control file does not read not "OK".

The operating system receives one command:

```
cmd /c «shell script».
```

The name of the shell script file is substituted for the shell script, making cmd.exe execute the two commands in this file. After this execution ArisFlow tests:

- Whether the execution (DOS-box) has finished;
- The existence and contents of the control file;
- When the *Status Checking Output Data* option in the Status Checking Preferences dialog has been checked: whether output data have changed after the execution.

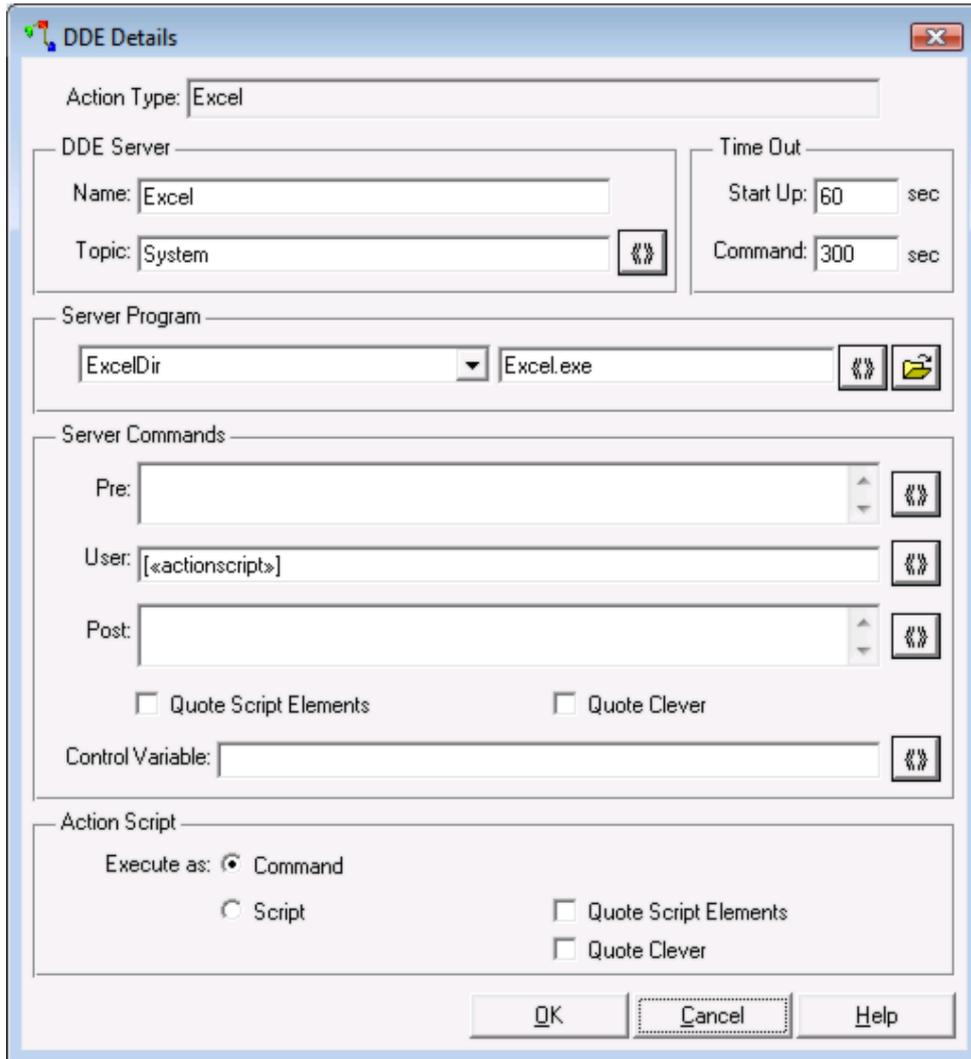
When all these requirements are met the action and its output data receive the up-to-date status.

The Special Key Options chapter describes a number of shortcut keys useful when editing fields with script elements, such as the *Program*, *Pre*, *User* and *Post* fields.

The section System Execution Parsing gives examples of System Details dialogs and explains what commands are eventually send to the operating system for these System Details dialogs.

DDE Details

The DDE Details dialog is started by selecting DDE base type in the [Action Type](#) dialog and pressing the *Details* button.



MS-Excel: a straightforward DDE action type.

In DDE (Dynamic Data Exchange) action types ArisFlow acts as a client passing commands to a DDE server, which is an executing program. This server executes these commands. The DDE details window defines how the DDE process is established. The window consists of four components:

- DDE Server;
- Server program;
- Server commands;
- Action script.

The DDE Server section describes how a DDE connection to the DDE server is established. The server is recognised through the Name/Topic combination. For details of this Name/Topic combination the user is referred to the manual of the server program. The server definition has the following fields:

- *Name*. This is the name by which the server is to be accessed by the client (ArisFlow) in any DDE activity;
- *Topic*. This is the topic passed to the DDE server by the client. This functions as a second key in accessing the DDE server. The topic can be defined using directory aliases and variables;
- *Start-up time-out*. This is a time limit in which ArisFlow will attempt to start up the DDE server as described in the Server program section. A start-up time out is not compulsory;

- *Command time-out.* This is a time period within which the DDE server is supposed to have acknowledged a single DDE command. If no acknowledgement of the command is passed back to the client (ArisFlow), the execution of that command is considered to have failed. Time outs are not compulsory. If no time out is assigned, ArisFlow will wait indefinitely for acknowledgement of the command (unless interrupted by the user).

The server program describes the executable program, which is the DDE server. It is not compulsory to provide the server path. When a DDE action is started, ArisFlow will check whether the Server is available through the Name/Topic combination. If not, ArisFlow will attempt to start the program defined in this section. If no program is defined or the program has not started within the start up time, the execution of the action has failed.

The Server Commands section consist of:

- **Edit fields:** Pre, User and Post edit fields;
- **Options:** Quote Script Elements and Quote Clever;
- **Control variable.** After the execution of the last command of the DDE script (or the last post command) ArisFlow may request the value of this control variable to the server. The value returned by the server should be "OK", otherwise the action is considered to have failed. The control variable is an extra check on successful completion of the DDE action. The control variable field is optional. It can be defined using variables.

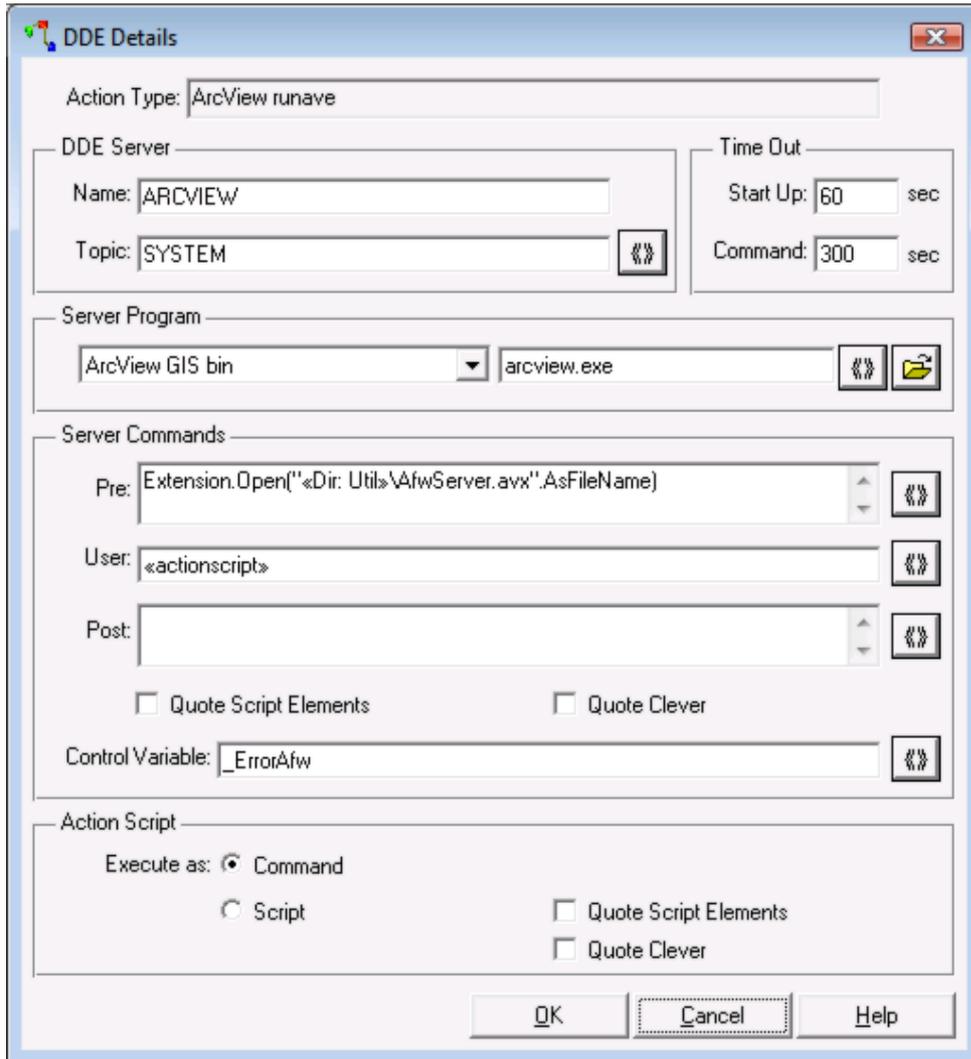
The «action script» option in the User field is required.

The Action Script consist of:

- **Execute As** and possibly **Extension**, and **Quote Script Element** and **Quote Clever Options**.

One important hint: when the DDE server does not acknowledge the existence of a program, this may well be because either the server name or the topic has been filled in incorrectly.

The second example of a more complex DDE action type definition is the ArcView runave example provided with ArisFlow.



ArcView DDE action type definition.

The Directory "Arcview GIS Bin" is the directory on which the program "arcview.exe" is located. Commands are grouped together into the action script file. This is a temporary file with extension ".ave". The action script file is passed in one single command, where the name of the script file is substituted for the «action script». When the script is executed ArisFlow requests the value of the `_ErrorAfw` control variable to the DDE server. The Action Script of the ArcView action should assign a value to the `_ErrorAfw` variable. Upon completion of the execution of the action script, ArisFlow will only pass the execution result as "OK", if the value of `_ErrorAfw` returned by the ArcView server is "OK".

In this special case an Avenue script is executed in ArcView. The ArisFlow utility "AfwServer" loads the script into the ArcView project, compiles the script and executes it.

The Special Key Options chapter describes a number of shortcut keys useful when editing fields with script elements, such as the *Topic*, *Pre*, *User* and *Post* fields.

System/DDE Edit Fields

The **System Shell Script** and **DDE Server Commands** sections contain three or four edit fields:

- **Program.** This single line occurs only in the System dialog. It defines the command for starting the system dialog on the operating system. The «shell script» is a required component in this field and should occur exactly once;
- **Pre.** These are command lines (possibly more than one) executed before the action script is executed;
- **User.** This single line describes how the «action script» (script of the action being executed) is to be executed. The «action script» option should occur exactly once in the User field;
- **Post.** These are command lines executed after the user script was executed. This may involve the clean up after the execution and the checking of the action results, usually through a control file.

The **Special Key Options** chapter describes a number of shortcut keys useful when editing fields with script elements, such as the *Program*, *Pre*, *User* and *Post* fields.

The Program, Pre, User and Post edit fields may contain the following pre-defined options, which

can be inserted by pressing the pre-defined options button :

- «shellscript» (Program field only) or «action script» (Pre and User fields only);
- «control file»;
- «variable»;
- «directory».

In the Program edit field the «shell script» defines how the shell script is to be executed. When the Shell Script Execute As Command option is set each line in the shell script is embedded into the line described in the Program edit field and then passed for execution. When the Shell Script Execute As Script option is set the shell scriptfilename is substituted for the «shell script» and passed as a single execution line.

In the User edit field the «action script» defines how the action script is to be executed. When the Action Script Execute As Command or Program option is set each line in the action script is embedded into the line described in the User edit field. When the Action Script Execute As Script option is set the action scriptfilename is substituted for the «action script».

In the Pre edit field the «action script» is only relevant when the Action Script Execute As Script option is selected. When executing the action the script filename is substituted. This allows some pre-execution actions or tests to be performed upon the action script file.

A «control file» checks the execution correctness as described in the **Execution of Actions** section. A control file is a unique name for a temporary file on the TEMP directory, which is generated by ArisFlow. After execution of a command ArisFlow checks the existence and contents of the control files. Control file checks can be inserted everywhere and as often as required into any edit field.

A «variable» or «directory» alias may be inserted into any edit field, leading to a Variable Selection or Directory Alias Selection dialog. The variable is displayed in the script in variable notation, the directory alias in directory alias notation. The directory path is substituted during execution.

The section **Execution Parsing** gives more examples of System dialogs and how these are parsed for execution.

System/DDE Options

The System Shell Script and the System and DDE Action Script sections may contain the following options:

- Execute as;
- Use Server Directories (System base type only);
- Quote Script Elements and Quote Clever;
- Extension.

Execute As has three possible options:

- Command;
- Script;
- Program (System Action Script section only).

When Execute As Command or Program is selected each statement is passed separately to the higher level, which is either the «action script» in the User field or the «shell script» in the Program field. When Execute As Program is selected every action of this type is treated as a program. Blank lines in the script definition are not executed as commands.

When Execute As Script is selected each statement in that section is grouped together into a script file. This is a temporary file with designated Extension. Blank lines in the script are included in the script file. ArisFlow generates a unique filename (on the TEMP directory) and creates it. The name of the action script file or shell script file is passed to the next higher level and substituted for the «action script» or the «shell script».

When Use Server Directories is checked execution of script commands will take place on a distant server. For every directory alias (e.g. in directory references, in action and data script elements, and TEMP directories in control and script files) the server directory path defined in the [Directory](#) dialog is substituted instead of the actual directory path. If the server path is not defined this will lead to an error message. Server directories cannot be used in the Program edit field; this field should actually describe how a command is to be passed to the server. When the Shell Script uses server directories it is obvious that the Action Script must do so as well.

Selecting the *Quote Script Elements* option implies that all script elements will be enclosed by double quotes. The *Quote Clever* option deals with directory aliases which may be part of a full path. More is explained in the Quote Script Elements and Quote Clever section. Examples are given in the Execution Parsing section.

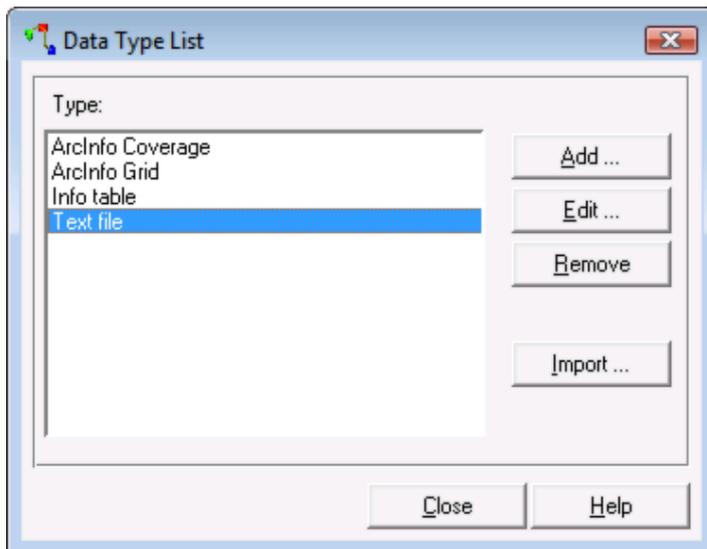
The Extension is the extension required for the temporary script file. The extension dot is optional in this field; ArisFlow will make sure the extension always starts with a dot.

Data Types

Data Types

Data Type List

When the *Define/Data Types* menu option is chosen, the following dialog box will appear:



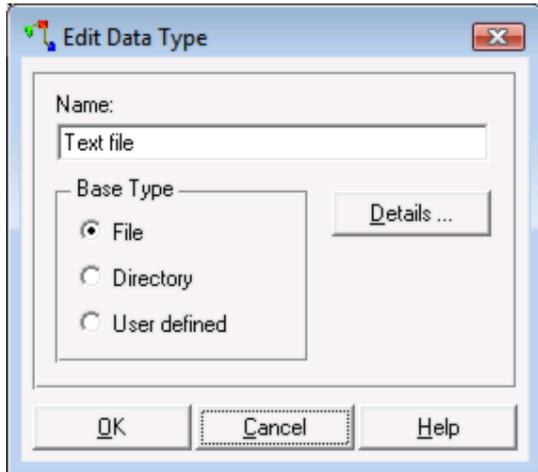
Data type list dialog displaying the names of all data types defined for the project.

This dialog screen has four special buttons:

- **Add.** A new data type can be created and inserted into the data types list;
- **Edit.** Parameters of the selected data type can be changed. Double clicking a data type in the data types list has the same effect;
- **Remove.** The selected data type is removed from the project. If the selected data type cannot be removed ArisFlow gives a message listing the data, which use the data type. This feature can be very useful in tracing data, which are of a certain data type;
- **Import.** Datatypes can be selected from other ArisFlow project files and copied into this project. Non-existing variable and directory alias definitions occurring in the imported datatypes may be copied into this project as well.

Data Type

When adding or editing a data type in the Data Type List dialog the following dialog screen will appear:



Data type dialog for defining a data type.

The Name is the logical name by which the data type is known in the Data dialog. Any name will do, as long as it is unique. Blanks are allowed in the name.

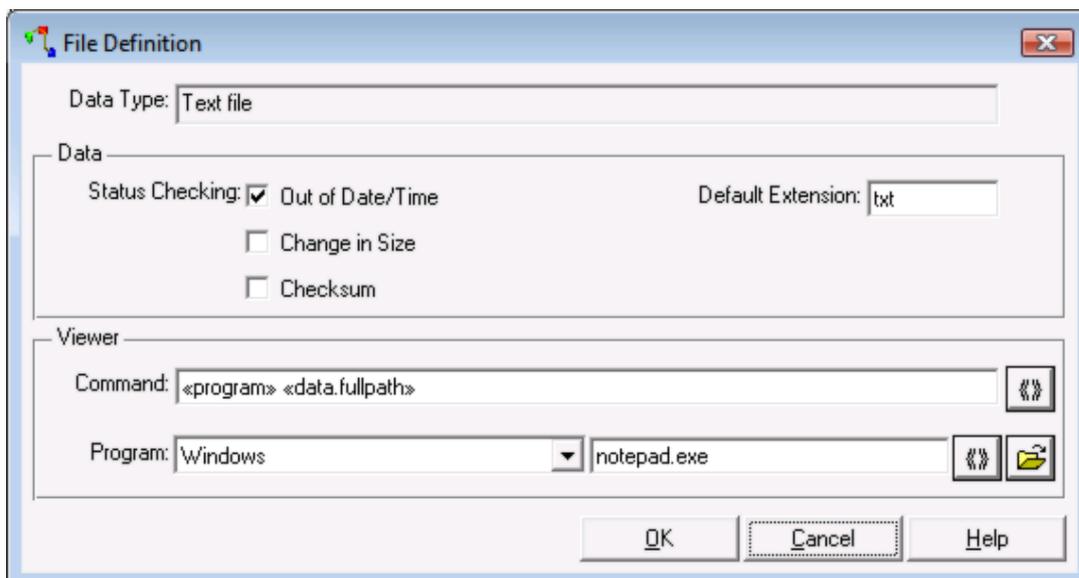
The *Base Type* has three possibilities:

- File. These are single files;
- Directory. Directory data represent whole directories;
- User-defined. Data not to be described as a File or Directory, e.g. a table in a database or a file in an archive.

When pressing the *Details* button the user can define more details of the data type, which is explained in the File and Directory Data Types and the User-defined Data Types sections.

File and Directory Data Types

The File, respectively Directory Definition dialog is started by selecting file, respectively directory base type in the Data Type dialog and pressing the *Details* button.



Definition of ASCII File data type.

Data of any File Data Type represent single files. At first glance all files are just files or directories, and different approaches in treating one file or another seem hard to justify. File and directory data types are distinguished because of differences in:

- The file browser default extension (file base type only);
- Testing criteria applied to detect a change of the file status;
- Viewer program used in displaying the contents of a file or directory;
- Meta-data.

Data of Directory Data Type represent a whole directory. The data, which are of such a data type, contain all files in that directory, in every sub-directory of that directory, and in all deeper sub-directories.

Data types of File base type can be passed a *default browser extension*. The browser in the Data dialog will start browsing for files of this type. Obviously this browser can still browse for any other types of files as well. When defining the default extension, the extension's period can be given, but this is not mandatory.

Data types of Directory base type have no default extension; a browser will search for directories only.

When checking the status of a file any combination of three criteria is possible:

- Date/Time. Checks whether the last modification time of a file has changed. A change is not necessarily later in time; an earlier time is also an indication that data have changed;
- Size. This is the number of bytes in a file;
- Checksum. The checksum is a value, which is sensitive to the sequence of the bytes in a file. So when two bytes in a file are swapped, the file's size is unchanged, but the checksum is most likely to be different.

When checking a directory all files in the directory and every sub-directory are tested. This implies that:

- Date-time is the time of the latest modified file in the directory or any of its sub-directories;
- Size is the sum of the sizes of all files together in the directory and its sub-directories;
- The checksum algorithm is performed over all the files in the directory and its sub-directories.

In most cases the date/time checking of a file or even a directory is sufficient. In situations where the date/time of any files may not be altered when the data are altered, checking for size or even checksum may be necessary.

Status checking of directories can be time-consuming, as every file in the directory and every sub-directory is checked. Calculation of the checksum is the most time-consuming operation, as this requires checking every byte in a file or directory.

A special situation is where none of the date/time, size or checksum flags is checked. In this case ArisFlow only checks for the existence of the data. A change of data is never detected, which will definitely give problems where the Status Checking on Output Data is set in the Execution Preferences dialog.

Finally a *viewer program* may be defined for a data type. The viewer program facilitates the viewing of data contents on the screen by selecting the *View/Data* menu option or toolbar

button . When no or incorrect viewer details are provided ArisFlow will attempt to view (file) data by passing just the filepath to the operating system, so that the operating system default viewer for the data is started. Therefore, in most situations it is not necessary to fill in

anything for *Viewer Command* and *Viewer Program*.

In few cases the default viewer would not lead to the correct viewer being started. The viewer to be started may then be defined by:

- The viewer command string. This is a string passed to the operating system when the viewer is to be started.
- The viewer program. This is the program to be started when viewing data of this type.

The viewer command string may contain pre-defined options, which can be inserted by pressing the  button:

- «program». The full path of the viewer program. This option is required in the command string;
- «data». The full path of the data being viewed. At least one of the data full path, data location or data entry is required in the command string;
- «directory». Any directory alias can be included in the command string. The directory is displayed in directory alias notation;
- «variable». Any variable can be included in the command string. The variable is displayed in variable notation.

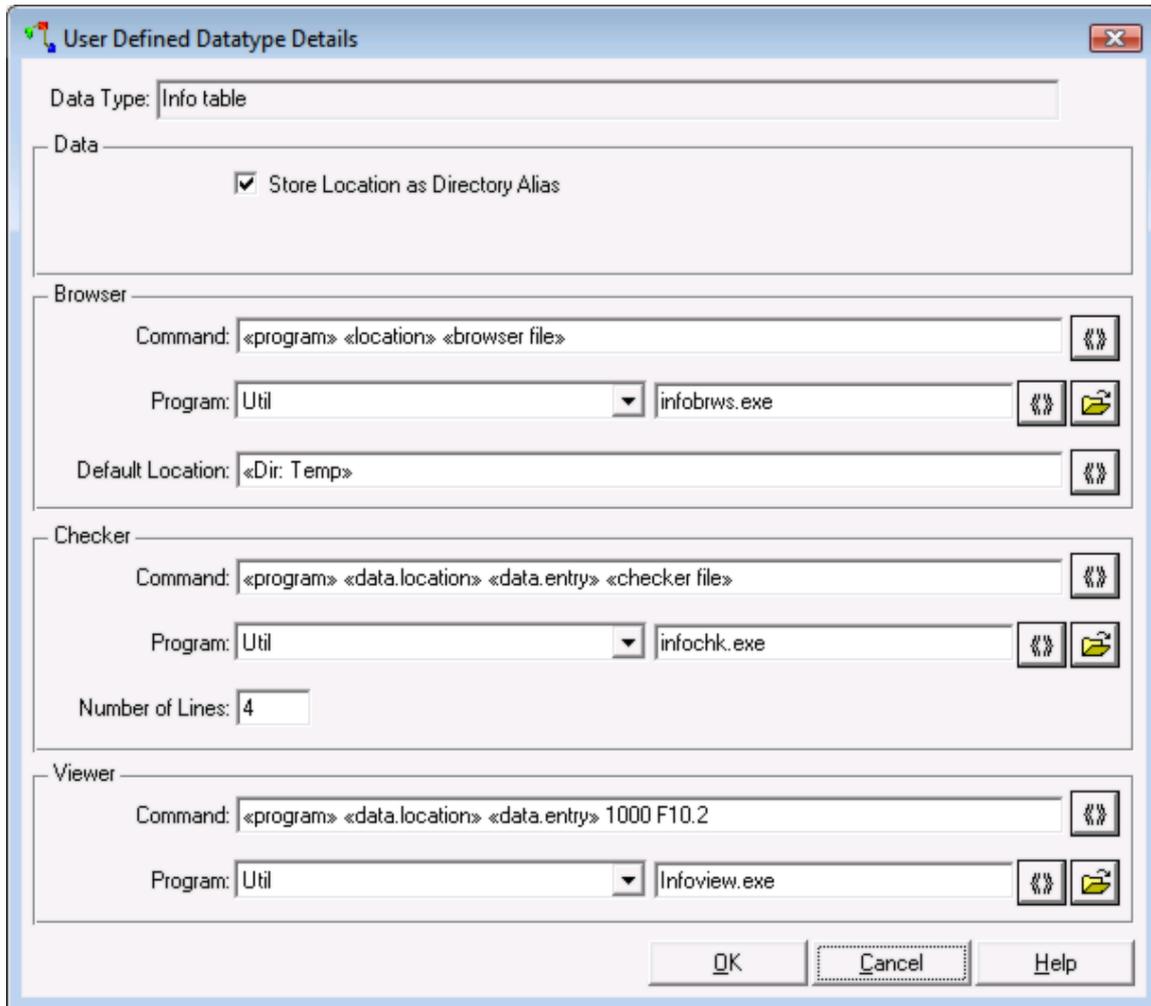
The viewer program can be browsed by pressing the browse button . The browsing starts at the currently selected directory. Its results leave the directory alias and entry variables as much intact as possible. Variables can be inserted in the viewer program entry by pressing the  button, which directly starts the Variable Selection dialog.

The example is the definition of the data type "Text file". This data type is of the *File* base type. The viewer is "notepad.exe" located in the "Windows" directory. In most cases the viewer command string is as displayed. Data are viewed by executing a string that has the full path of "notepad.exe" followed by the full path of the data being viewed. Status checking is very extensive here: every possible criterion is checked for possible changes in the data of the data type "File".

User-defined Data Types

User-defined Data Types

The User-defined Data Type Details dialog is started by selecting the user-defined base type in the [Data Type](#) dialog and pressing the *Details* button.



Definition of the Info table user-defined data type. For this example the location is stored as a string.

The user-defined base type allows the user to define any type of data possible. This is possible because the user-defined data types pass their statuses through other programs (the checker program). ArisFlow starts the checker program and evaluates its results. The often very straightforward implementation of the checker program is the most important requirement in defining a new user-defined data type.

The *Data* section defines properties of the user-defined data:

- Whether the data location is stored as a directory alias.
- The *Separator* string between location and entry for the user-defined data.

These properties and their consequences for the Data involved are described in the User-defined Data section.

The user-defined data type dialog requires the definition of between one and three programs:

- Browser program;
- Checker program;
- Viewer program.

Only the checker program is essential. This checker program determines whether the data exists and whether the data has changed. The other two programs are useful for the user, but not essential in running ArisFlow. Browser, checker and viewer programs are not a part of ArisFlow. The user has to provide these utilities by either building them him/herself or by using programs supplied by either ARIS or a third party.

Browser, checker and viewer descriptions are provided in the different sections. Details of what to enter in these sections are provided in the Specifications of the Browser, Checker and Viewer Program sections.

Each of the Browser, Checker and Viewer sections contains:

- The location of the program in the *Program* field;
- The *Command* field, which describes how the program is started.

The program location is browsed by pressing the *Browse*  button. The *Command* describes the command through which the program is started. A number of pre-defined options, selected by pressing the *Options Selection* button , are useful in describing this command string. These options are described in the Specifications sections described further down. The Special Key Options chapter describes a number of shortcut keys useful when editing fields with script elements, such as the different *Command* fields.

There are two special fields:

- The Default Location defined for the browser program;
- The *Number of Lines* for the checker program.

When browsing for data the browser command string is passed, as defined in the Browser Command section. Part of this command string can be a location. If any data location was already filled in in the Data dialog that location is passed as the location for the browser command. When no location was filled in, the default location is passed in the browser command.

The *Number of Lines* for the checker program describe how many lines in the checker file are relevant for the checker result. The first line describes the status checking result, such as OK, NOK or an error message, and is always relevant.

When the viewer program definition is not correctly assigned (for example when the viewer program is not present on disk) ArisFlow will attempt to view data by simply passing the filename to the operating system. Thus when the operating system recognises the file extension its default viewer is started. This makes it often unnecessary to fill in the viewer program details.

The function and requirements of browser, checker and viewer utilities are described in the following sections:

- Specifications of the Browser Program;
- Specifications of the Checker Program;
- Specifications of the Viewer Program.

A fourth group of utilities is the User-defined Data Access programs. These access programs accept commands sent by ArisFlow (as described in the Action Script) and pass them on to the user-defined data.

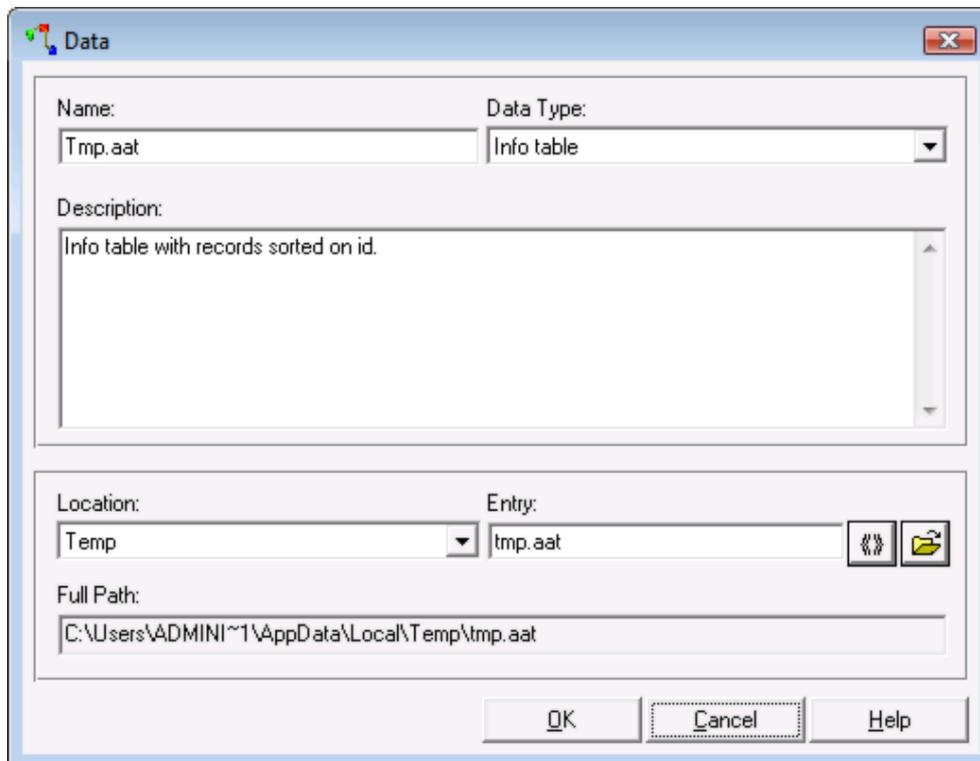
In the dialog example a browser and a checker for the Info data-type are defined. The

browser program *infobrws.exe* simulates how ArcView browses through the directory structures of ARC/INFO info tables. The checker program *infochk.exe* checks and returns the date and size of the info data.

ArisFlow sends its script commands to the *aicInt.exe* program. This program accesses ARC/INFO. The System dialog for the ARC/INFO action type is displayed in the User-defined Data Access Program section.

User-defined Data

When a Data object in the flowchart is double clicked the Data dialog is opened. After a data-type of user-defined base type is selected the Location and Entry fields appear in the Data dialog:



Example of dialog for data with user-defined data type, where location is stored as a string.

Data of the user-defined base type, are defined by two fields:

- Location;
- Entry.

How the location field appears in the Data dialog depends upon whether the *Store Location as Directory Alias* field was checked in the User-defined Data-type dialog. There are two possibilities:

- The location is stored as a string. The above-mentioned checkbox was not checked.
- The location is stored as a directory alias. The above-mentioned checkbox was checked. The directory alias can be selected from a directory alias list.

The full path of the data can be used in the Action Script of the action to which the user-defined data is connected. When the location is stored as a directory alias the default directory

separator, either backslash (\) or forward slash (/), is inserted between the location and entry. When the location is stored as a string the Separator string defined in the User-defined Data-type dialog is used. The *Full Path* field in the data dialog displays the full path.

During execution some actions may use server directories, as defined in the Directory Alias dialog. Where user-defined data have the location stored as a directory alias the Server Path of that directory is inserted into the Action Script.

When the user presses the *Browse*  button the browser program defined in the User-defined Data-type dialog is started. This browser should return the location and the entry of the user-defined data. When the data is stored as a directory alias ArisFlow will attempt to match the location returned with the directories in the Directory Alias List. If any alias matches that directory alias appears in the *Location* field. If no directory matches the user has the chance to define the directory alias for the location that was returned.

The string location and entry of the data may contain variables. These are inserted through the Variable Selection dialog when the  button is pressed.

When checking the status of user-defined data the procedure is as follows:

- ArisFlow checks whether all the relevant fields (data type, location and entry) have been filled in.
- Where the location is stored as a directory alias ArisFlow checks whether the directory actually exists. Where the location is stored as a string nothing is checked here.
- The checker program is started. It should return whether the data are present and the status of the data as described in the Specifications of the Checker Program section.

Specifications of the Browser Program

When the user presses the *Browse*  button in the Data dialog of user-defined data the browser program is started. ArisFlow starts this program with parameters as defined in the User-defined Data-type dialog.

The purpose of the browser program is to return the location and the entry of the data, which are filled in for the location and entry fields in the data dialog. These values are returned in a temporary browser file written by the browser program. ArisFlow supplies the name of this browser file to the browser program as one of its parameters.

ArisFlow assumes that the browser program executes until the temporary browser file is detected. It is not possible to press the *Browse* button in the Data dialog until ArisFlow has detected the temporary file. Therefore it is recommended that a browser program write something, preferably *NOK*, to the temporary file when the browser program is cancelled.

The browser file may contain up to three lines:

- Result of the browsing;
- The location of the data;
- The entry of the data.

The first line returns the browsing result. There are three possibilities:

- OK;
- NOK;

- Error text.

If the first line in the browser file reads *OK* and the second and third lines return correct location and entry of the data the location and entry fields in the data dialog are updated. If the first line in the browser file reads *NOK* the browser was cancelled. Anything written on second or consecutive lines is ignored. If any other text (not *OK* nor *NOK*) or nothing is written on the first line an error message is displayed.

A browser program usually requires some parameters when it is started. These parameters should be supplied in the browser command line defined in the *User-defined Data-type* dialog. For the browser program it is important to note that every parameter is always enclosed by double quotes. Parameters are defined as *pre-defined options*. The possible parameters are:

- «program». The full path specified for the browser program. This is required;
- «location». The start-up location at which the browser program starts browsing;
- «browser file». The name of the temporary file, written by the browser program;
- «directory». Starts the *Directory Alias Selection* dialog enabling the insertion of a directory alias in the browser command string;
- «variable». Starts the *Variable Selection* dialog enabling the insertion of a variable in the browser command string.

Make sure the parameters in the command line are in the same order as the parameters that the browser program expects. The «program» is normally the first parameter that the browser program expects.

The use of a «location» is recommended as a start-up directory for the browsing, but it is not required. When the user specifies the «location» option in the browser command line the browser program will be passed the parameter:

- The current location filled in for the location field in the *Data* dialog of the user-defined data;
- If no current location defined for the data: the *Default Location* defined for the browser in the *User-defined Data-type* dialog. This default may contain a directory alias reference at its first position;
- If no default location was filled in and the data has no current location an empty location, enclosed in double quotes, is passed to the browser.

The «browser file» is mandatory. ArisFlow creates a filename of a non-existing file in the *TEMP* directory and passes this filename to the browser as defined in the browser command line. ArisFlow starts reading the browser file soon (not immediately) after it detects the existence of it. Therefore the browser should write its results to the temporary file immediately after opening the file, or ArisFlow will have read the file before the browser program wrote any results. After the file was evaluated it is removed, unless the *Remove after Execution* option was unchecked in the *File Preferences* dialog.

Finally if no browser command line or browser program is specified in the user-defined

data-type dialog ArisFlow will not start a browser when pressing the *Browse*  button in the *Data* dialog for data of that type. In this case the location and entry field values can only be altered in these fields. Therefore a user-defined data type does not require a browser program, but it can make life much easier for the ArisFlow user.

Specifications of the Checker Program

The checker program describes the [state of user-defined data](#). This state check is performed

when the [Execute/Check Status](#)  toolbar button is pressed; when [location](#) and [entry](#) have been (re-) defined in the [Data](#) dialog; during the [execution of the flowchart](#) and possibly when an ArisFlow project is opened. This makes it obvious that the definition of a checker program is mandatory for in the [User-defined Data-type](#) dialog.

The purpose of the checker program is to return the [status](#) of the [user-defined data](#). These data are defined as the location and entry in the [Data](#) dialog, and may be passed as parameters when ArisFlow starts the checker program. The checker program should write a [temporary file](#) with the results of the check. ArisFlow supplies the name of this checker file to the checker program as one of its parameters.

The [checker file](#) should contain:

- The checker program result on the first line;
- The description of the user-defined data status on subsequent lines.

The checker program can return three possible results described on the first line of the checker file:

- OK;
- NOK;
- Error text.

If data exists the first line should always read *OK* and nothing else. The subsequent lines then describe the state of the data. The total number of lines (including the first *OK* line) in the temporary result file should be equal to the value filled in for the *Number of Lines* field in the [User-defined Data-type](#) dialog. If this number of lines is increased all data of this datatype become [not-up-to-date](#). If the checker file contains more lines than nominated, the surplus is considered irrelevant by ArisFlow, and hence not tested for possible changes. If the checker file contains fewer lines than nominated a message is displayed and the data become [not-up-to-date](#).

It is possible to nominate just one line. In that case the checker detects only changes between *OK* (existing), *NOK* (not existing) and error (location not existing) status. A change in the data is not detected, which may give problems where the Status Checking on Output Data is set in the [Execution Preferences](#) dialog.

If user-defined data do not exist, but can be created, the first line in the checker file should read *NOK*. In this case [project input-data](#) appear grey (undefined) in the flowchart. But other data appear red ([not-up-to-date](#)) in the flowchart. This situation is similar to [file or directory type data](#), where the existence of the directory also indicates that data can be created if it is not project input-data.

If the first line in the checker file reads neither *OK* nor *NOK*, or if the checker file is not created, the data are assumed to be [undefined](#) and will appear grey in the flowchart. ArisFlow will give a message with the contents of the first line. This checker should return an error result:

- If the user-defined data location is incorrect. This is similar to [file or directory type data](#), where the directory alias refers to a not-existing directory;
- If the checker command line definition in the [User-defined Data-type](#) dialog is not correct.

The checker program requires some parameters when it is started. These parameters should be supplied by the checker command line defined in the [User-defined Data-type](#) dialog. For the checker program it is important to note that every parameter is always enclosed by double quotes. Parameters are defined as [pre-defined options](#). The possible parameters are:

- «program». The [full path](#) specified for the checker program;
- «data.location». This is the [location](#) field in the [data](#) which is to be checked;
- «data.entry». This is the [entry](#) field in the [data](#) which is to be checked;

- «data.fullpath». This is the [full path](#) of the [data](#) which is to be checked;
- «checker file». The name of a temporary file to be written by the checker program;
- «directory». Starts the [Directory Alias Selection](#) dialog enabling the insertion of a [directory alias](#) in the browser command string;
- «variable». Starts the [Variable Selection](#) dialog enabling the insertion of a [variable](#) in the browser command string.

Make sure the parameters in the command line are in the same order as the parameters that the checker program expects. The «program» parameter is normally the first parameter that the checker program expects.

The [location](#) and [entry](#) should define the [data](#) being checked. It is obvious that these parameters can be passed to the checker program. The «checker file» is mandatory when calling the checker program. ArisFlow generates a filename of a non-existing file in the [TEMP](#) directory and passes this filename to the checker as defined in the checker command line.

ArisFlow starts reading the [checker file](#) after the checker program has terminated, and the checker file has been created. After the checker file was evaluated it is removed, unless the *Remove After Execution* option was unchecked in the [File Preferences](#) dialog.

Specifications of the Viewer Program

The viewer program is much more straightforward than the browser and checker programs. It displays the contents of the user-defined data. Often the viewer can be an already existing application.

ArisFlow starts the viewer program when the user presses the [View/Data](#) toolbar button . The currently selected data is viewed. The viewer program is started according to the definition in the viewer command line defined in the User-defined Data-type dialog. For the viewer program it is important to note that every parameter is always enclosed by double quotes. Parameters are defined as pre-defined options. The possible options are:

- «program». The full path of the viewer program is substituted for this option;
- «data.location». This is the location of the data, which are to be viewed;
- «data.entry». This is the entry field of the data, which are to be viewed;
- «data.fullpath». This is the full path of the data which is to be checked;
- «directory». Starts the Directory Alias Selection dialog enabling the insertion of a directory alias in the browser command string;
- «variable». Starts the Variable Selection dialog enabling the insertion of a variable in the browser command string.

The «program» is normally the first parameter of the viewer program. It is not uncommon to pass extra parameters, usually program options. These options can be inserted in the command line definition.

If no viewer command line or viewer program is defined the data path is send to Windows, which will start the default viewer associated to the path's extension. If that fails Windows will ask for the viewer program to be associated to the path's extension (Open With dialog).

User-defined Data Access Program

The Action Script defines how the action's input data are read and output data are created. It may be possible to directly manipulate these data. But, certainly with user-defined data, special utilities for accessing these data may be required.

ArisFlow is not designed to directly manipulate user-defined data. It requires other programs to perform this manipulation. ArisFlow sends its commands to the program. This program can be:

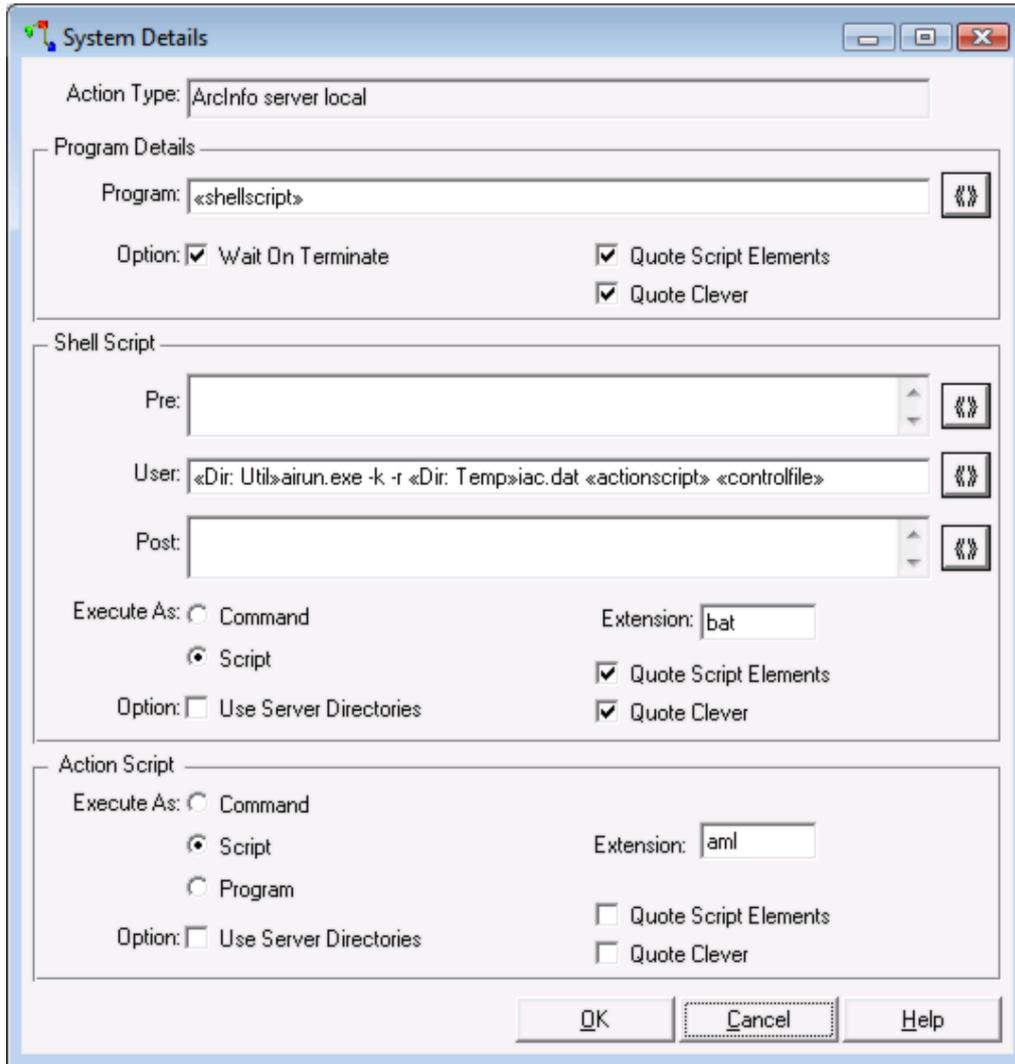
- A DDE server;
- A command-passing program. This program is executed for every command line and receives this command line as a parameter.

DDE servers just receive commands and execute them. The action type in the Action dialog has base type DDE. The command lines in the action script should follow the guidelines set out for the DDE server.

A command-passing program is started for every command line in the Action Script and receives this command line as an input parameter. The command-passing program then interprets and executes the command. How the command line is passed is defined in the [Action Type](#) of the Action whose script is executed. This action type has base type System. The *Program* field or the *User* field defines how the command line is passed as a parameter to the command-passing program.

Often the DDE server or the command-passing program already exists. The user then only has to follow the guidelines of this program when defining the action type and the action script using this program. Sometimes the user should build the program accessing the data.

A command-passing program is *airun.exe*, a utility that enables the access of ARC/INFO by ArisFlow. The system action type dialog screen for accessing *airun.exe* looks as follows:



System Details dialog of Arc/Info action type showing the usage of the command passing program *airun.exe*.

The operating system receives a command, which starts the execution of the shell script file (a batch file, with *.bat* extension). This script file contains one single line, which starts the *airun.exe* program located on the "util" (utilities) directory. This program has two options and three parameters, which are described in detail in the *airun.txt* file located in the utilities directory. The options are:

- *-k*: the *aml* is not removed after execution (keep);
- *-r*: makes the program display error messages and ask for a return after an error was detected.

The parameters of the *airun.exe* program are:

- The file *iac.dat*, which is written on the "shared" directory. This file, created by Arc: [iacopen], contains the RPC information.
- The action script file. This is an ARC/INFO *.aml* file;
- The name of a temporary control file. The usage of a control file is always recommended; it makes ArisFlow wait until the execution of each command is completed and checks whether the command was executed successfully.

Note that quotes are automatically inserted in the shell's *.bat* file, whereas user has chosen to enclose the action script elements in the *.aml* file himself, where required. This is described in the Quote Script Elements section.

Import/Export of Definitions

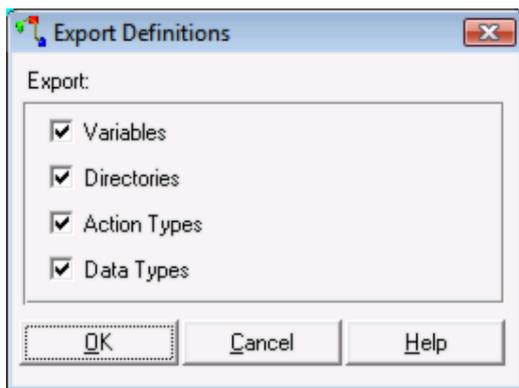
Import / Export of Definitions

It is possible to exchange definitions (variables, directory aliases, action types and data types) between ArisFlow projects. There are two main mechanisms for importing/exporting definitions between ArisFlow projects:

- Import / Export Through Definition Files using the File/Import and File/Export menu options;
- Importing From Definitions Windows by pushing the *Import* button in the Action Type List, Data Type List, Directory Alias List or Variables List dialogs.

Import / Export Through Definition Files

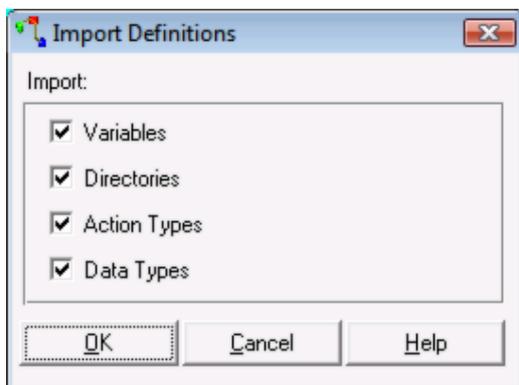
When the [File/Export](#) menu option is selected all definitions (variables, directory aliases, action types and data types) in an ArisFlow project are written to an ASCII text file. This file is called the definitions file. The following dialog appears:



Dialog for export of definitions.

When OK is pressed ArisFlow asks for the path of the definitions file. When any of the checkboxes is unchecked all definitions of that definition type are not exported.

When the [File/Import](#) menu option is selected the following dialog appears:



Dialog for import of definitions.

Again, when OK is pressed ArisFlow asks for the path of the definitions file. Unchecking any of the checkboxes means all definitions of that definition type are not exported.

Importing definitions through definitions files is most useful when starting a new project. Through this mechanism many useful definitions can be imported by a simple procedure. When one or a few definitions should be imported into an existing project it is recommended to use the Importing From Definitions Windows mechanism instead.

The definitions file is an ASCII file. Therefore the user can edit a definitions file quite easily. If the file becomes corrupted, ArisFlow will display what is wrong when importing the definitions from that file. When a definition already exists, ArisFlow will tell so, and ask the user whether the definition should be overwritten. Always be very careful with overwriting definitions, as the current definition may be an important definition in the project.

The contents of the definition file obey the following rules:

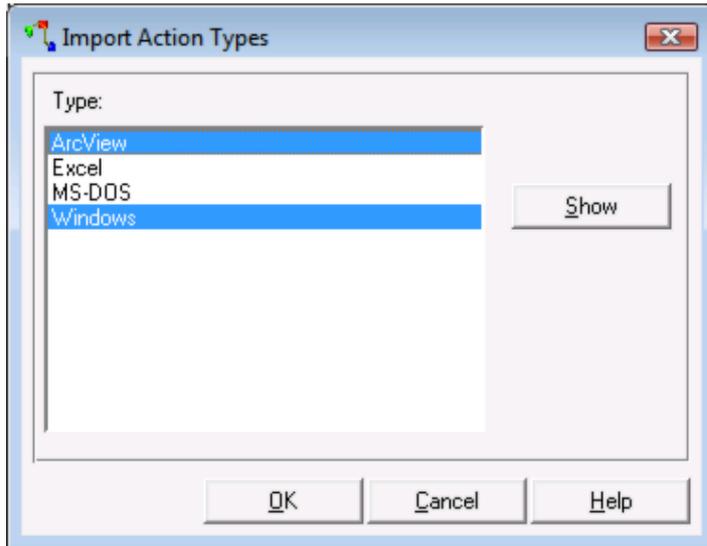
- Each object is declared by the keyword *Variable*, *Directory Alias*, *Action Type* or *Data Type*. A colon and the name of the object follow the keyword. When importing definitions declarations can be in any order;
- A value is assigned to an object by defining: `FieldName = Value`. The = sign is required in this assignment;
- The FieldName is the name, which appears in the related dialog (see [Variables](#), [Directory Aliases](#), [Action Types](#) and [Data Types](#) sections). A fieldname must be preceded by a group name (the text written in the box surrounding the field concerned) if a fieldname appears multiple times in the related dialog, otherwise the group name is optional;
- Checkboxes can only have values On and Off. All other values are as they are displayed in the related dialog;
- Values which have multiple lines use continuation markers, which are an extra » character at the end of the line being continued and also a » character at the beginning of the next line;
- Most assignments are optional when importing definitions. When an assignment to a certain field is missing the default value is assigned to that field. For checkboxes the default value is On. For other field types the defaults are obvious;
- For action types and data types the name and base type are defined first. After the base type is defined all other field values can be defined;
- When importing definitions a variable or directory alias should have been declared before it can be used in a value definition. The exception when another variable is used in a variable's definition or when a root directory is used for the definition of another directory alias. The variable or root directory should be declared inside the same block of variables or directory aliases as the variable or directory alias using it.

It is not (yet) possible to import all definitions from an ArisFlow project file (.afd file) at once.

Importing From Definitions Windows

It is possible to import one or more actions types, data types, variables or directory aliases by pressing the *Import* button in the Action Type List, Data Type List, Variables or Directory Aliases dialogs.

After the *Import* button is pushed a "Select File" dialog appears, asking for an ArisFlow project file . The "Import Action Types" dialog will then display all action types that were defined in the selected (or exporting) ArisFlow project:



Import action types dialog with two action types selected.

It is possible to select more than one action type in the *Type* listbox. Clicking the action type for the first time will select it, clicking it again will unselect the action type.

When the OK button is pressed the definitions of all selected action types are exported from the exporting project into the current ArisFlow project. If a name clashes with a name already in the current project "_2" is appended to the name of the imported action type.

When the *Show* button is pushed the definition of the last action type clicked on is displayed in a read-only [Action Type](#) dialog.

The import dialog displayed for data types, directory aliases and variables is similar to the import dialog for action types. When importing data types, action types or directory aliases definitions it is possible that such a definition uses a directory alias or variable. In that situation the user is asked whether that definition should also be imported, or, if the definition already existed, whether it should be overwritten.

The direct import from definitions windows is recommended for importing one or a few definitions in an existing ArisFlow project. In a new ArisFlow project it is usually quicker to import (many) definitions through a definitions file.

Page Set-up

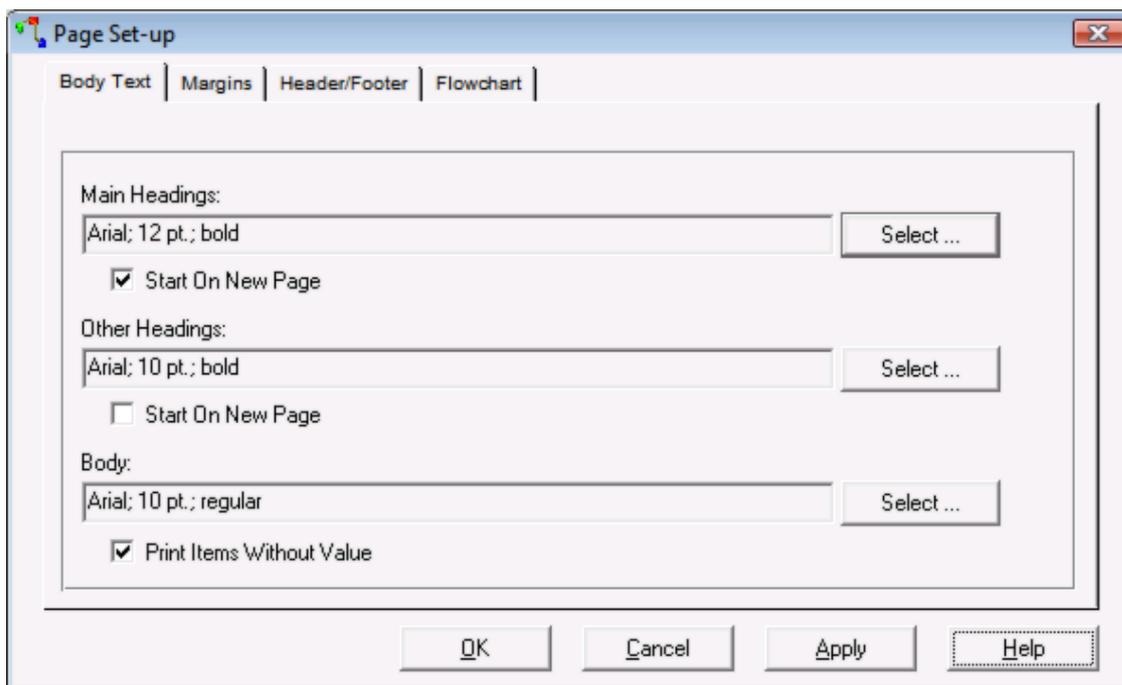
Page Set-up

Selecting the [File/Page Set-up](#) menu option starts the Page Set-up dialog. This dialog describes the layout of a page. The [Print](#) dialog defines which pages will be printed. The Page Set-up dialog has four tabs:

- [Body Text](#), enabling selection of fonts when printing text.
- [Margins](#), enabling setting the paper margins.
- [Header/Footer](#), defining text displayed in header and footer of each page, including the possible inclusion of page numbers.
- [Flowchart](#), defining flowchart settings such as font, printing as one page, page numbering order, etc.

The [Page Grid Lines](#) in the flowchart set page sizes.

Body Text



Body text tab with default font settings.

The Page Set-up Body Text tab allows the user to select fonts when printing ASCII text. Fonts are defined for:

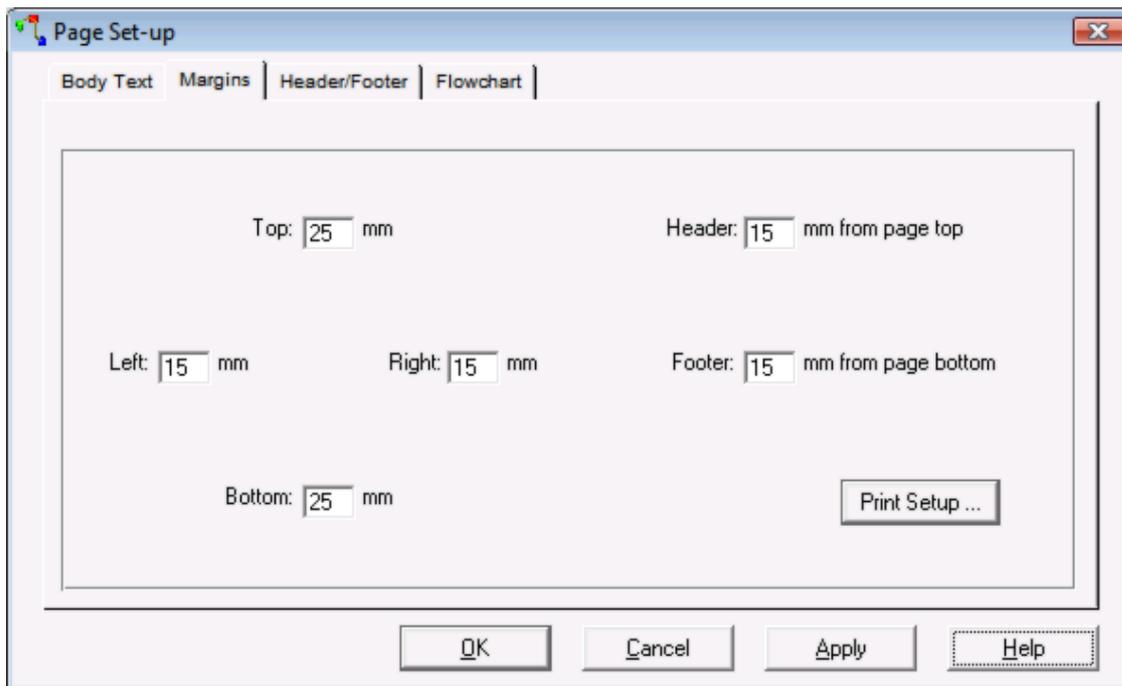
- Heading1. These are the main categories, as listed under Print Items in the Print dialog;
- Heading2. These are main categories which are supposed to be one level below the main categories of Heading1;
- Body. This is all the other text in the print.

Pressing a *Select* button enables setting the related font size and type.

The following options can be selected:

- *Start On New Page*. Sections which are of Heading1 or Heading2 categories appear on new page if this option is checked for the respective section;
- *Print Items Without Value*. When this option is checked every item is printed, whether the item has any value assigned to it or not. If this option is not checked only items and values of items with values assigned are printed.

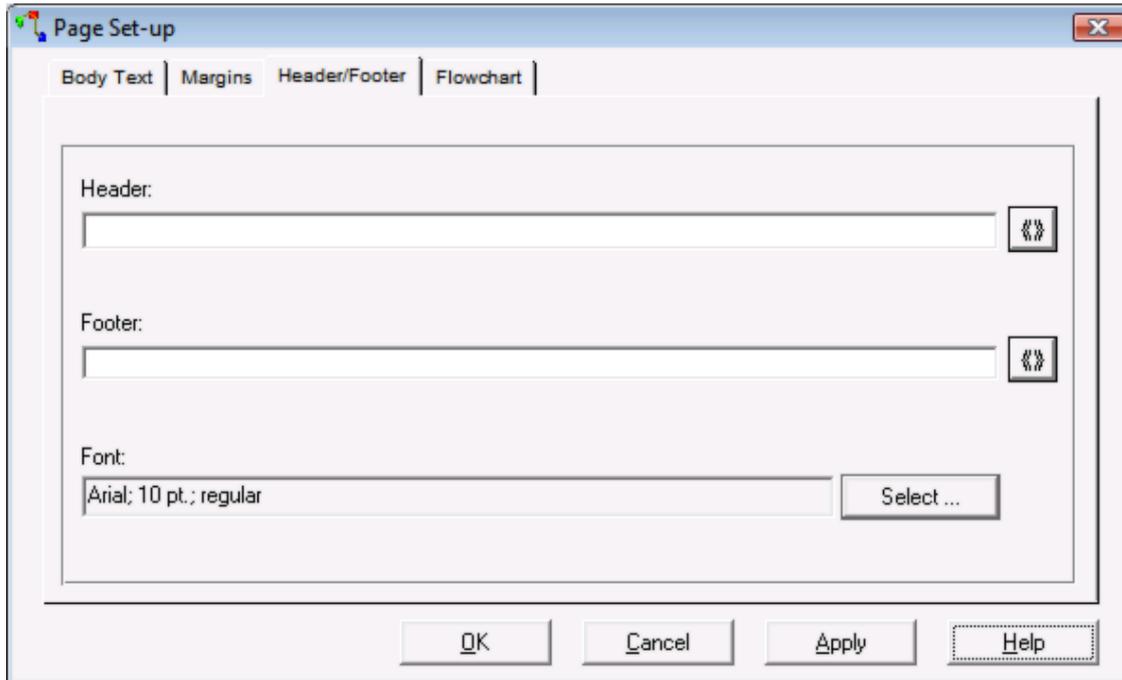
Margins



Margins tab with its default settings.

The Page Set-up Margins tab allows setting of the margins from the edges in the printed page. Text and flowchart pages have the same margins. Header and footer margins should be a good deal less than top and bottom margins respectively, otherwise the header or footer may overlap with text or with a part of the flowchart printed. The *Print Set-up* button allows setting and also checking of the printer and page sizes. Knowing these page sizes may be useful in setting the page margins.

Header/Footer

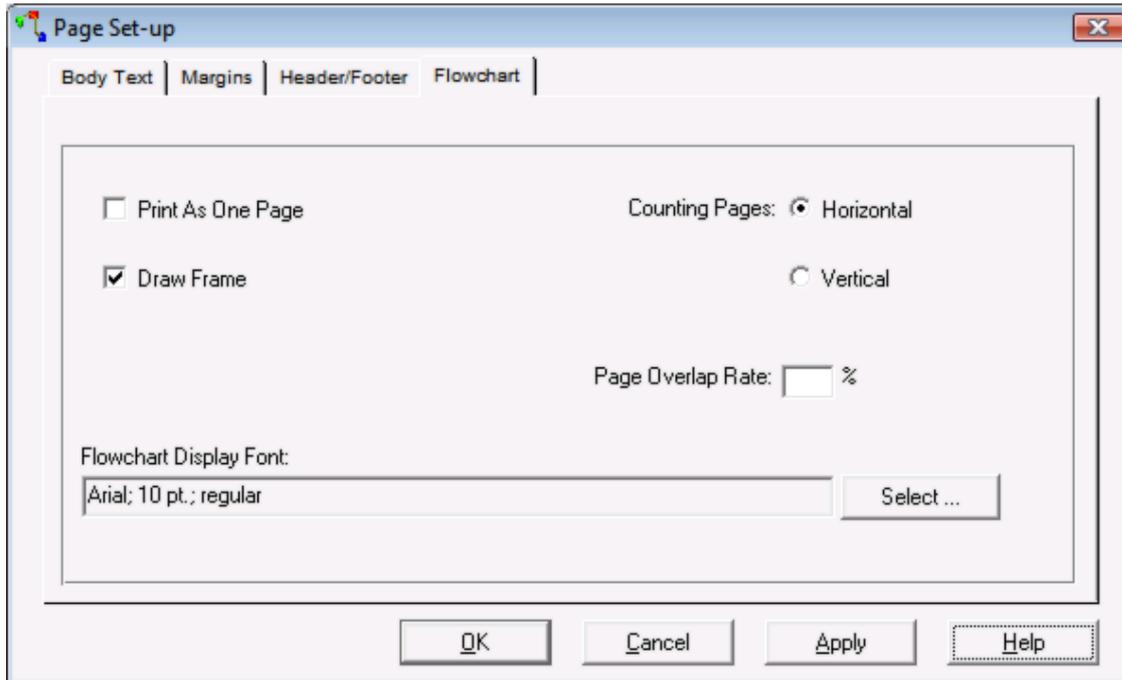


Header/Footer tab.

The [Page Set-up](#) Header/Footer tab describes what the header text at the top of the page and the footer text at the bottom of the page will look like. This text is centred at the header or footer position of every printed page. Both header and footer set-up can include project meta-data which are described in the [Properties](#) section. Project properties, page numbers, [variables](#) values and [directory alias](#) local paths can be included in the header or footer text by pressing the respective [pre-defined option selection](#) buttons  and then inserting the required pre-defined options.

Pressing the Select button and selecting new font characteristics can change the font of header and footer text.

Flowchart Tab



Flowchart tab with its default settings.

The [Page Set-up](#) Flowchart tab defines the appearance of the flowchart when it is printed. It describes five features:

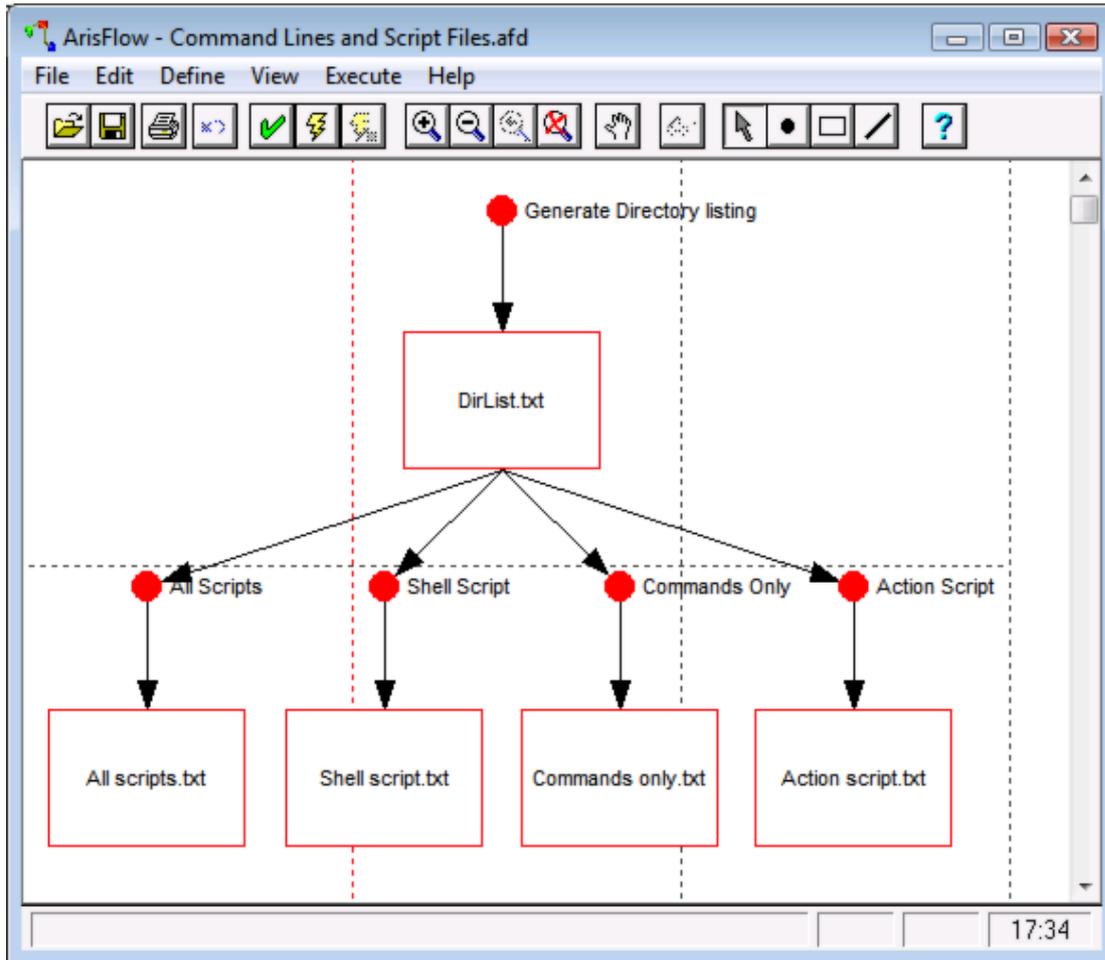
- Print as one page. Setting this option makes sure that the flowchart is always printed on exactly one page;
- Draw Frame. Draws a rectangle around the flowchart shown on every flowchart page;
- Page overlap rate. The percentage of overlap between every page. The right and bottom sides of the flowchart displayed stretch further than indicated by the print grid. The percentage indicates how much of the flowchart is shown on more than one page;
- Counting pages. Defines whether the printed page following a previously printed page is primarily located to the right (counting horizontally) or below (counting vertically) that page;
- Flowchart font.

Page overlap rate and counting pages direction are of importance only if the flowchart is printed on more than one page.

Pressing the Select button and selecting new font characteristics can change the flowchart font. The selected font becomes the font in the current flowchart display after the new settings in the flowchart tab are accepted. When printing the font is scaled onto the printed-paper according to the scaling of the flowchart [Page Grid Lines](#).

Page Grid Lines

The page grid lines define what part of the flowchart is printed on one page. By selecting the menu option [View/Show Page Grid](#) the page grid lines are shown in the flowchart. These are dashed black horizontal and vertical lines. Each rectangle displayed is a portion of the flowchart, which fits on one page. If the *Print as one page* option is checked in the Flowchart Tab, there will be one horizontal and one vertical grid line showing, which will exactly enclose the flowchart.



Example of page grid lines in flowchart, where the dashed red line is the grid line currently being shifted by the user.

If the flowchart does not fit on a single page, multiple lines are shown. The flowchart printed page projection can then be adjusted by moving the grid lines.

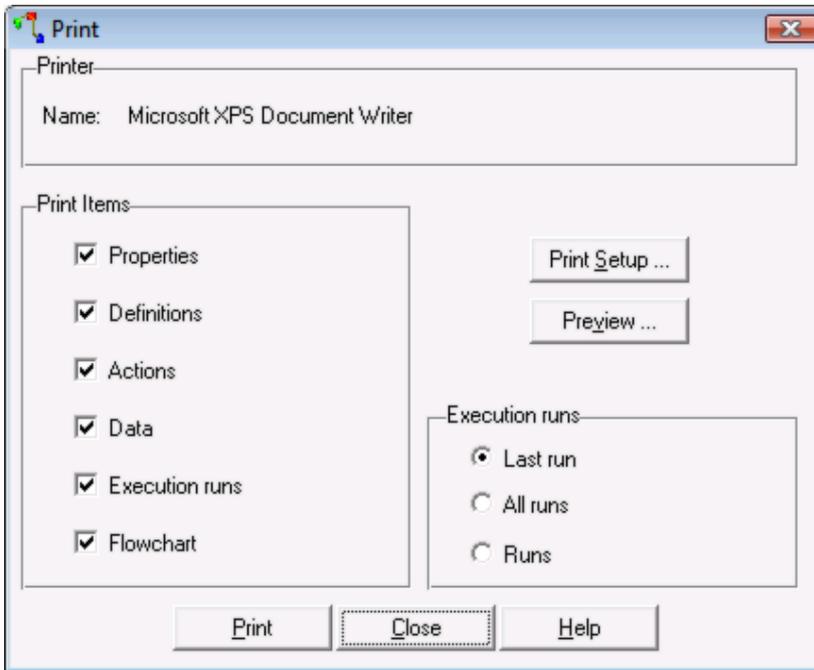
Select Cursor (☞), then clicking the left mouse-button on a grid line and moving the grid line with the mouse button pressed down. The whole grid will shift accordingly. The line moved appears in red. The grid page will conform to a minimum size. An attempt to move a grid line beyond what is represented by this minimum size will result in stopping the grid movement.

Finally the menu [View/Set Default Page](#) will make the grid rectangles on the screen have the size of the printed page, reduced by the margins. In this case the flowchart is projected 1:1 onto the printed page.

Print

Print

Selecting the [File/Print](#) menu option starts the Print dialog.



Print dialog defining which items will be printed.

The Print Items define which pages will be printed. When pressing the *Print* button the items checked are printed. One or more print items should be selected before the printing can be performed:

- [Properties](#). The project meta-data as described in the Properties section;
- [Definitions](#). The settings of special directories and environment variables. The definitions of variables, directory aliases, action types and data types in the project;
- [Actions](#). The actions in the project;
- [Data](#). The data in the project;
- Execution runs. Displays all or part of the contents of the [Execution Log File](#).
- The [Flowchart](#). The flowchart displayed is printed on one or more pages.

When execution runs is checked the contents of the [Execution Log File](#) are printed. This either prints:

- Last run. Only execution results of the last run described in the execution log file will be printed;
- All runs. All the contents of the execution log file will be printed;
- Runs. The contents of the numbers of the execution runs (which are the page numbers shown when the Execution Log File is displayed) described are shown. When no value is assigned in the first column the log of all runs starting from the first run is printed. When no value is assigned in the last column the log of all runs up to the last execution run is printed.

Selecting the [Page Set-up](#) menu option from the [File menu](#) enables setting print options, such as [text fonts](#), [paper margins](#), [header and footer lines](#) (including page numbers), and [flowchart](#)

[characteristics](#) (frames, fonts, etc.).

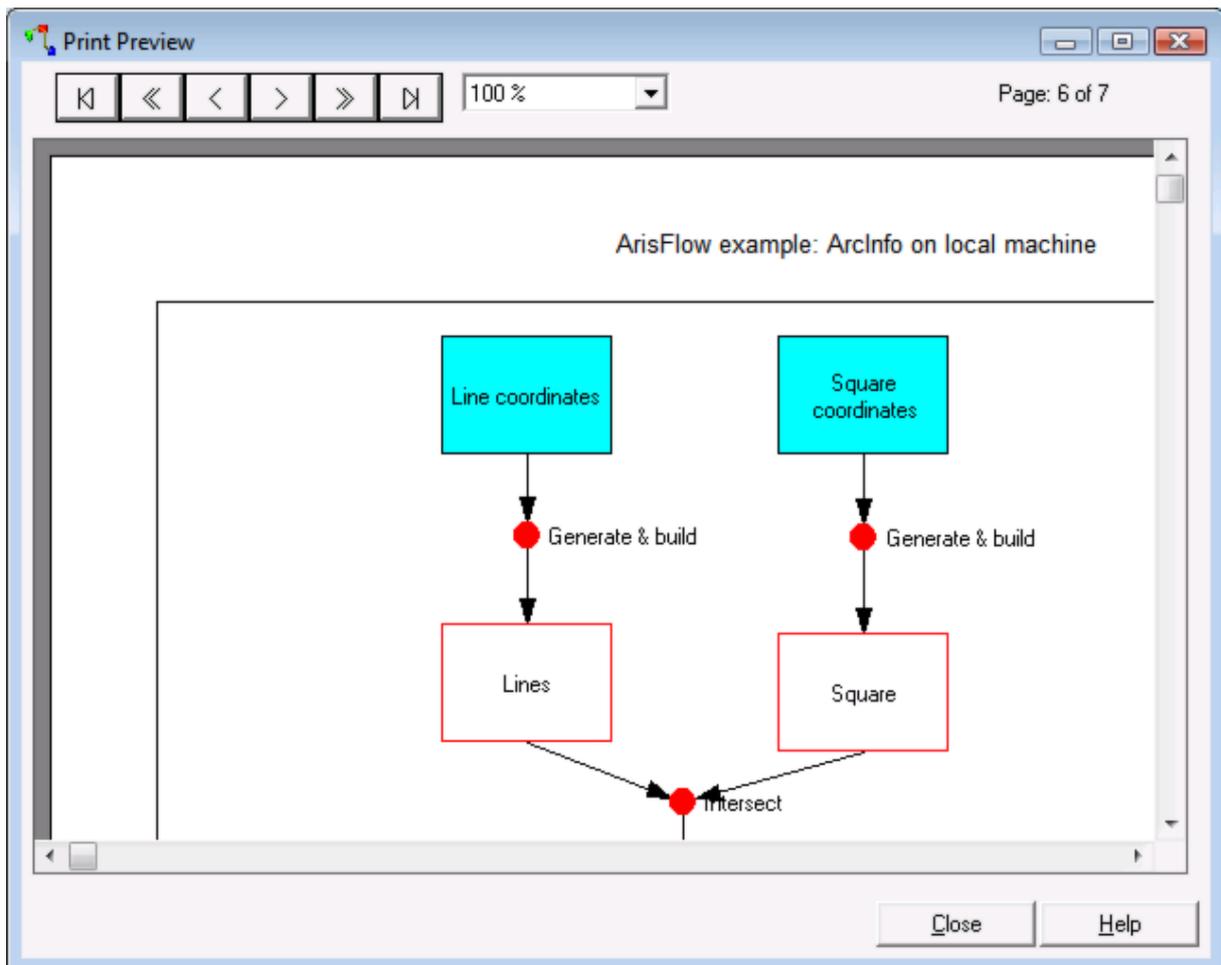
The *Print Set-up* button defines printer settings, such as which printer, portrait/landscape, paper size. Pressing the *Preview* button displays a [Preview](#) of the printed pages on the screen. Pressing the *Print* button creates a print. It is recommended to preview the print before actually printing.

When the *Print* or the *Close* button is pressed the print items selected are saved for the next time when the Print or [Print To File](#) dialog is displayed.

It is possible to write text and flowchart to a text and a bitmap file, instead of printing them. This is achieved by selecting the [Print to File](#) option in the [File menu](#).

Print Preview

The Print Preview window shows what a printed page will look like on paper. Pressing the *Preview* button in the [Print](#) dialog opens the Print Preview window.



Print Preview window.

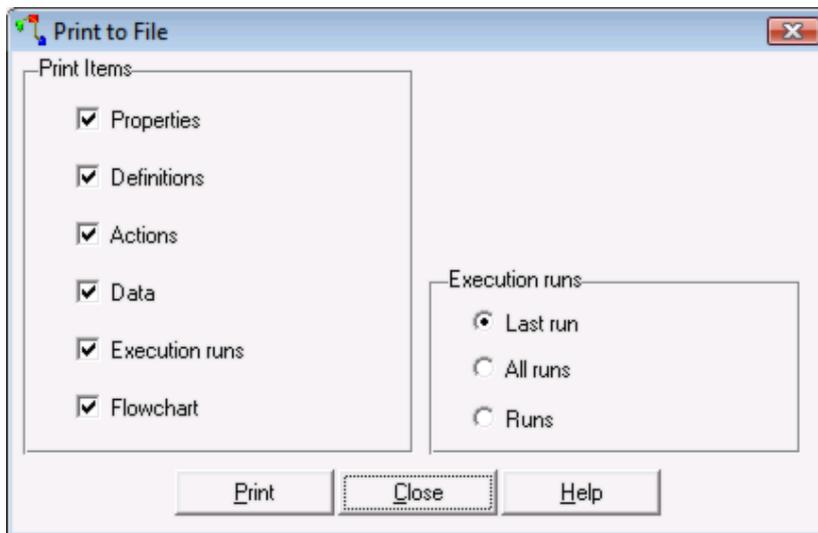
The buttons at the top of the Print Preview window allow the user to go through the different printed pages. These buttons indicate from left to right:

-  Move back to first page;
-  Move back rapidly (multiple pages);
-  Move back one page;
-  Move forward one page;
-  Move forward rapidly (multiple pages);
-  Move forward to last page.

Next to these buttons there is a percentage box in which the user can choose the scale of the page shown in the Print Preview window.

Print To File

Selecting the [File/Print To File](#) menu option starts the Print To File dialog.



Print To File dialog defining which items will be written.

The items written are selected in a similar manner as in the Print dialog. When the *Print* button is pressed the user must select one or two files:

- ASCII Text file, which always has .txt extension. Properties, definitions, actions, data and execution results are written to this file, so that the path of file is asked for when any of these options is checked.
- Windows Meta file, which always has .wmf extension. The path of this file is asked for when the flowchart option is selected. The flowchart is then printed into this file.

When the *Print* or *Close* button is pressed the print options are saved for the next time when the Print or Print to File dialog is displayed.

Windows Meta file is a format used for storage of vector and bitmap-format graphical data in memory or in disk files. Windows Meta files describe how graphical data were recorded. The actual file format used is the Aldus Placeable Windows Meta File format, which is not the standard Windows Meta file format. However most programs (e.g. MS-Word, Paintshop Pro), which import Windows Meta Files, will only recognise this Aldus Placeable WMF format.

The flowchart is copied 1:1 into the Windows Meta file. If the flowchart was zoomed in each object is printed with size as appears on the screen at that moment. It is noted that ArisFlow allows zooming to the default size by the [ViewZoom Default](#) menu option. Also zoom in and zoom out is possible by a standard factor 2.

Preferences

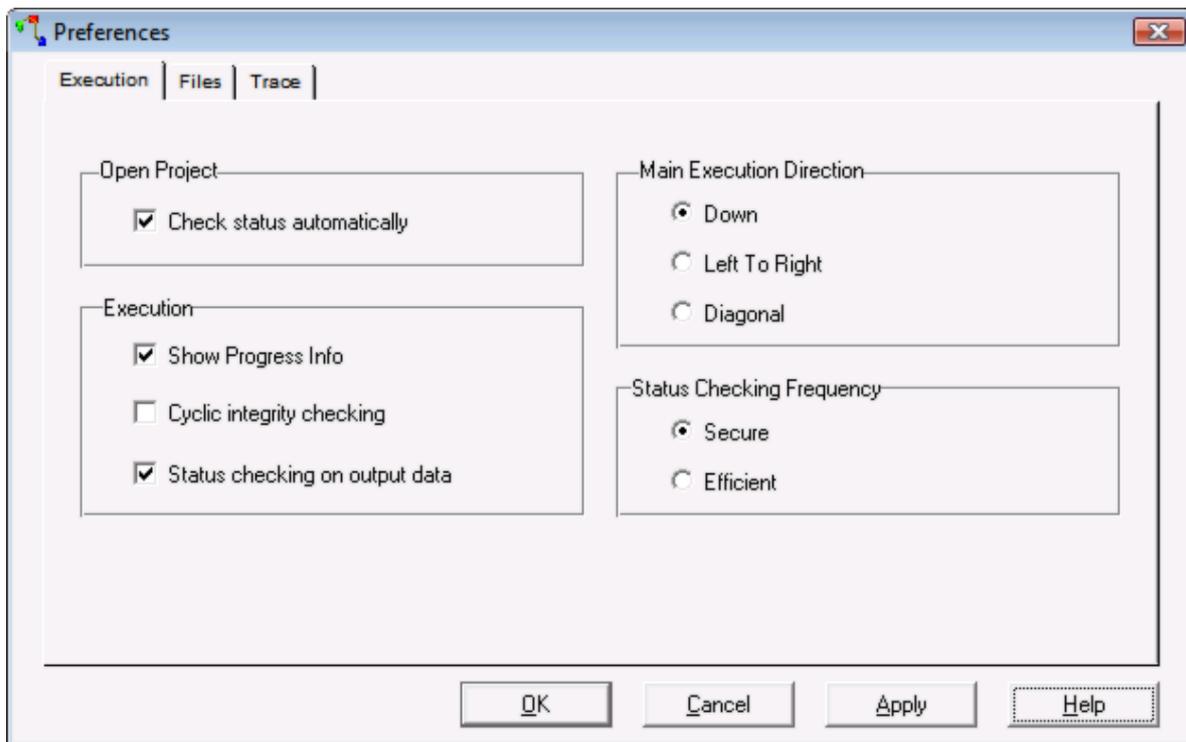
Preferences

The user can set preferences by selecting the [File/Preferences](#) menu option. The preferences dialog has three tabs:

- [Execution Preferences](#);
- [File Preferences](#);
- [Trace Preferences](#).

Execution Preferences

Execution preferences are started through the [File/Preferences](#) menu option.



Execution tab with its default settings.

The Preferences dialog initially starts with the *Execution* tab, which enables the setting of six options. They are:

- Check status automatically when starting an ArisFlow project;
- Cyclic integrity checking;
- Show progress info;
- Status checking on output data;
- Main execution direction;
- Status checking frequency.

The first option is *Check status automatically* when opening the ArisFlow project. If this is set, the

opening of a project will automatically lead to a status check. Thus the status represented should always be correct when opening such a project.

When *Show Progress Info* is set the Execution Progress window is displayed during execution or checking status of the flowchart. This is the default setting.

Cyclic integrity checking assigns iterative properties to the ArisFlow execution as described in the Cyclic Integrity Checking section. Marking the cyclic integrity option will automatically set the *Status Checking Frequency* at "Secure". This is necessary, or the cyclic integrity checking will fail.

When the option *Status Checking on Output Data* is set, this requires that after execution of an action its output data should be different from before action execution. Output data are tested for a change through the criteria defined for the Data Type of the data. This is also described in the Execution of Actions section. Checking this option may give problems where no change detection criteria (data/time, size, checksum for files or directories, one single line for user-defined data) are nominated for the data type of the data.

Main Execution Direction describes the main direction in which the actions in the flowchart are executed. There are three possibilities:

- Down. The main execution direction is from top to bottom. If two or more actions appear at the same height these actions are executed from left to right. This is the default setting;
- Left To Right. The main execution direction is from left to right. If two or more actions appear on the same vertical line these actions are executed from top to bottom. This setting is probably most useful where two or more long and independent vertical chains appear in the flowchart and these chains should be executed one after the other;
- Diagonal. The main execution direction is from the top-left to the bottom-right in the flowchart under an angle of 45 degrees.

Deviations from the main execution direction occur where the execution of an action is delayed until after all its parent actions, also those which are actually positioned behind that action, are executed.

Status Checking Frequency describes how often the status of flowchart elements is checked during the execution process. There are two options:

- Secure. The status of data is always checked prior to the execution of an action to which it is connected.
- Efficient. The status of data is not checked if, during the same execution run:
 - The status of the data was set earlier because the data was output of an action that was executed earlier;
 - The data is project input-data and has already been checked before the execution of another action.

Secure status checking frequency is required where data or program actions are influenced during the execution of an action in a manner that is not represented in the flowchart.

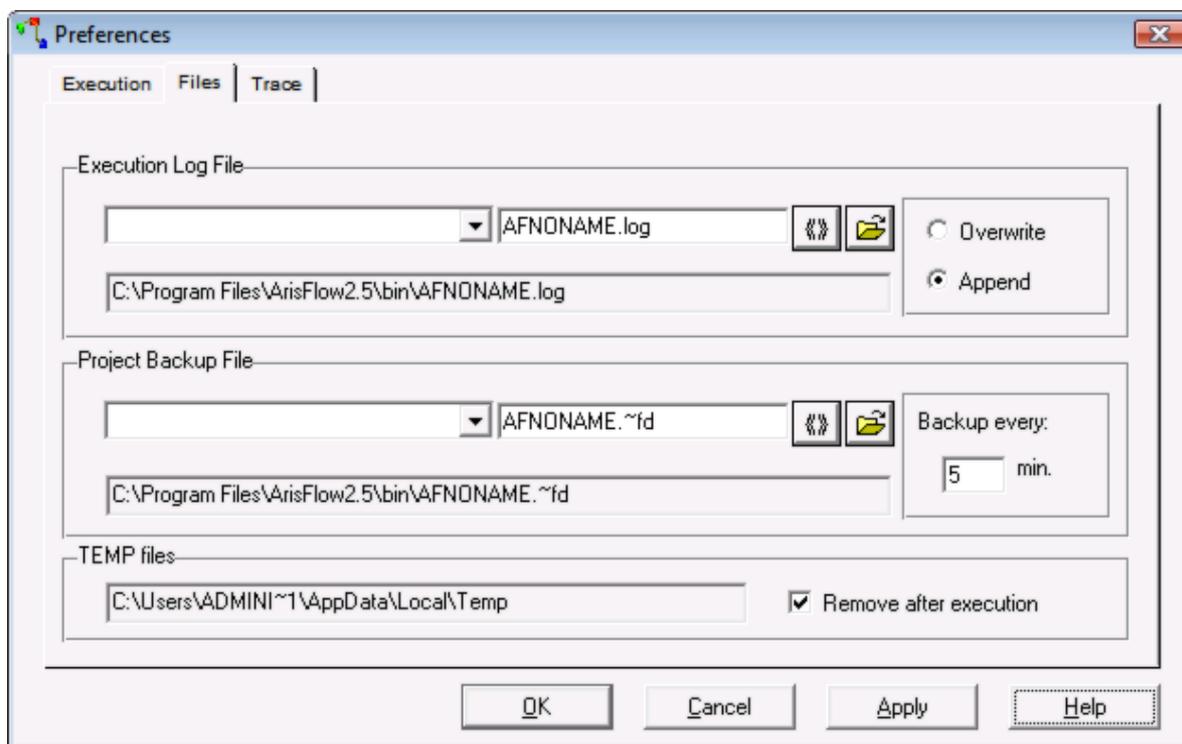
Examples are:

- In cyclic flowcharts where cyclic integrity checking is set.
- Where two different data-objects in the flowchart refer to the same data.
- Where influences outside the processes described in ArisFlow may affect data.

In general secure status checking frequency is recommended. Efficient status checking is only useful in projects where the checking of the status takes a long time, and where the user is absolutely certain that secure status checking frequency is not required.

File Preferences

File preferences are started through the [File/Preferences](#) menu option.



File preferences tab with default settings for "Project" ArisFlow project.

The Preferences dialog contains the *Files* tab, which has three sections:

- Execution log file;
- Project backup file;
- Temporary files.

The *Execution Log File* is an ASCII file created during the execution of ArisFlow. The contents of the execution file can be displayed in ArisFlow by selecting the *View/Log File* menu option. In the default setting the log file has the same name as the project file, but with ".log" extension, and is written to the directory where the ArisFlow dataflow project file is located. However, the user can select another directory and filename to which the log file is to be written. If the user selects a non-existing directory, the execution log file is still written to the work directory.

The execution log file has two possible file properties:

- Overwrite. Each execution run overwrites the results of previous execution runs. Only the results of the last execution run are available for consultation;
- Append. The result of each execution run is appended to the log file. Previous execution run results will therefore remain available for later consultation. This is the default setting.

A *Project Backup File* is created at intervals indicated in the *Backup every* field; provided some project data have been changed since the last project file or backup file was written. In case of power supply failure or other unforeseen failures, the backup file contains very recent project data. When restarting a project by loading the project file make sure the backup file is rapidly saved as another file, otherwise the user will be working with the same project and backup file, which in fact means there is no backup.

By default the backup file has the same name as the project file, with ".~fd" extension, and is written to the directory where the ArisFlow project file is located. However, the user can select

another directory and filename to which the backup file is to be written. If the user selects a non-existing directory, the backup file is still written to the work directory. Writing a backup file is not very time-consuming, therefore frequent backup is recommended.

The *TEMP Files* field shows the directory to which temporary files are written. This directory is:

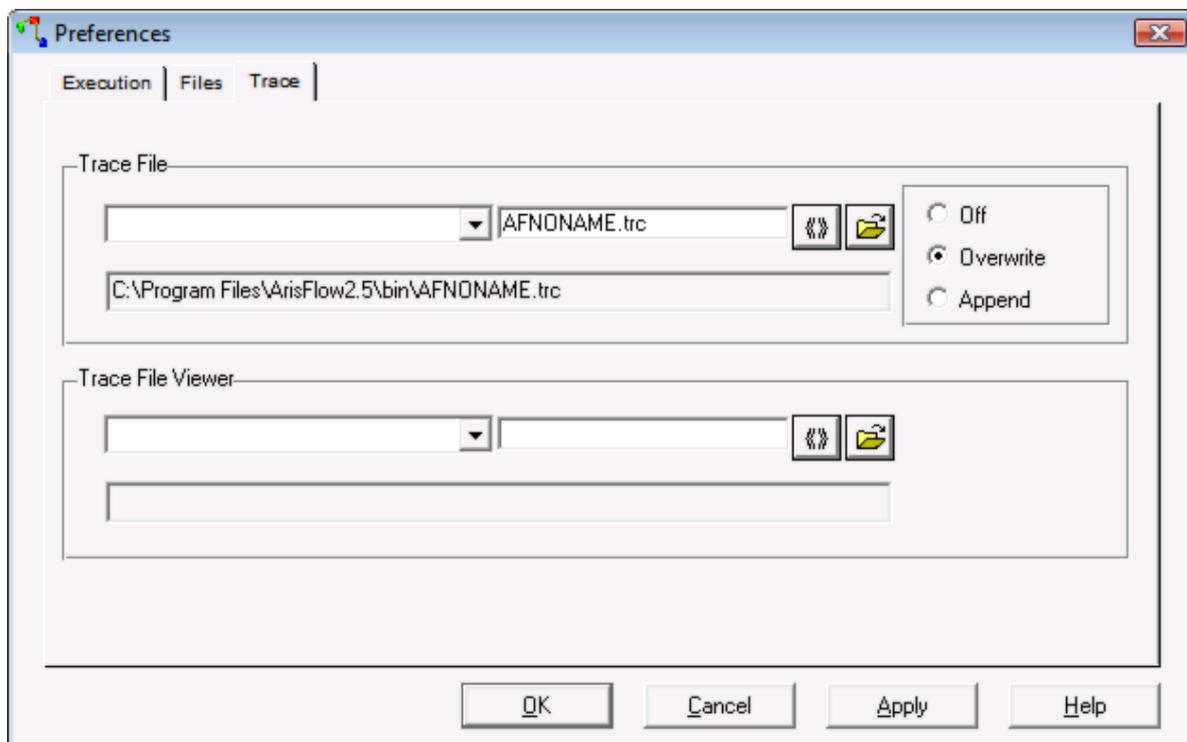
- The directory referred to by the directory alias "Temp". This alias name is capital insensitive. If no directory alias with name "Temp" is defined the temporary files are written to:
- The directory referred to by the environment variable "TEMP". This environment name is capital insensitive. If no TEMP environment is defined the temporary files are written to:
- The project work directory, which is the directory of the ArisFlow project.

If the temporary files should be written to a server directory this is only achieved when the "Temp" directory alias has a server directory defined. In all other cases no server directory is defined and an error message is displayed.

Temporary files are normally deleted when they are not required anymore. Sometimes it may be useful for the user to check the contents of temporary files, e.g. when execution of actions does not yield the expected result. Unchecking the *Remove Temporary Files* option prevents ArisFlow from deleting any temporary files. When temporary files are not removed, the names of temporary files are written to the Execution Log File.

Trace Preferences

Trace preferences are started through the *File/Preferences* menu option. Trace preferences set the path of the trace file and facilitates the definition of a viewer for the trace file.



Trace file preferences tab with default settings for "Project" ArisFlow project.

The *trace file* is used in monitoring the execution and status checking in ArisFlow. Also other

commands passed to the operating system (e.g. the command by which user-defined data browsers and checkers are started) are written to the trace file. There are three options in writing the trace file:

- None. No trace information is written. This option is recommended for normal usage because the writing of trace statements slows down the ArisFlow execution progress;
- Overwrite. A new file is written when this selection is made or when the project is started;
- Append. The trace information is always appended to the existing trace file.

The information written to the trace file consists of:

- What ArisFlow is currently doing;
- Statements passed to the system and DDE servers for execution;
- Temporary files that are being checked by ArisFlow;
- The results of an execution run, of file content checking, or data status checking.

With this and other information (like the Log File, see In Case of Failure section) it should be possible to deduct why a result is not as expected. However, a lot of information is written to the Trace File, making the retrieval of the information related to a certain problem a skill that requires a good knowledge of how to work with ArisFlow.

By default the trace file has the same name as the project file, with ".trc" extension, and is written to the directory where the ArisFlow project file is located. However, the user can select another directory and filename to which the trace file is to be written. If the user selects a non-existing directory, the trace file is still written to the work directory.

The trace file can be viewed by selecting the [menu](#) option View / Trace file. The viewer for the trace file can be defined in the Trace Preferences window in the *Trace File Viewer* box. If no viewer is defined the trace file is displayed in Notepad.

In Case of Failure

When the user resets the system or in case of power supply failure backup files should be present, containing recent changes made to the ArisFlow project. Of course the normal ArisFlow dataflow project file (".afd" extension) should also still be present, but this file does not contain any changes made after the last project save was performed. ArisFlow creates backup files of ".~fd" extension. The user can set backup file options (such as frequency of backup file creation and filename) as described in the File Preferences section.

Browsing a File

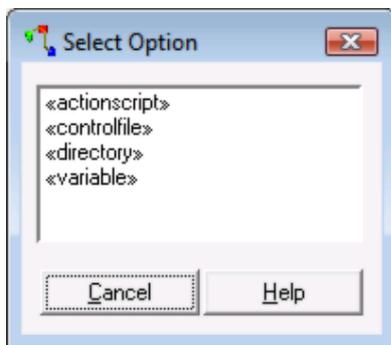
Many dialogs contain one or more browse buttons , which enable the user to browse for a file or directory. Pressing this button starts a file or directory selection dialog. When this dialog is completed, the full path of the file or directory selected is returned.

The full path returned is split up into a location and an entry. The entry is the name of the file or directory. This is the part following the last directory separator in the full path. With files, the extension is part of the file entry. The location is the remainder of the full path. The last directory separator is omitted from the location.

ArisFlow does not deal with location paths, but uses directory aliases instead. The location filled in into the location field is therefore the name of the **directory alias**, not the path. An existing alias with the correct path is usually filled in. If no existing alias has the location path returned by the browsing, ArisFlow will ask the user to make an existing path by showing the [Directory Alias](#) dialog (without showing the directory alias list dialog).

Pre-defined Option Selection

Most dialogs contain pre-defined option buttons . The button is always linked to a field. Pressing the pre-defined option selection button or pressing the Ctrl-O key starts a pre-defined option selection dialog, which may look as follows:



Example of a pre-defined option selection dialog.

A list of options appears. By clicking one of the options, that option is inserted into the field linked to the button. If the user presses *cancel*, no option is inserted. Options cannot be inserted inside other options. In the linked field all pre-defined options included are always enclosed in « and » brackets.

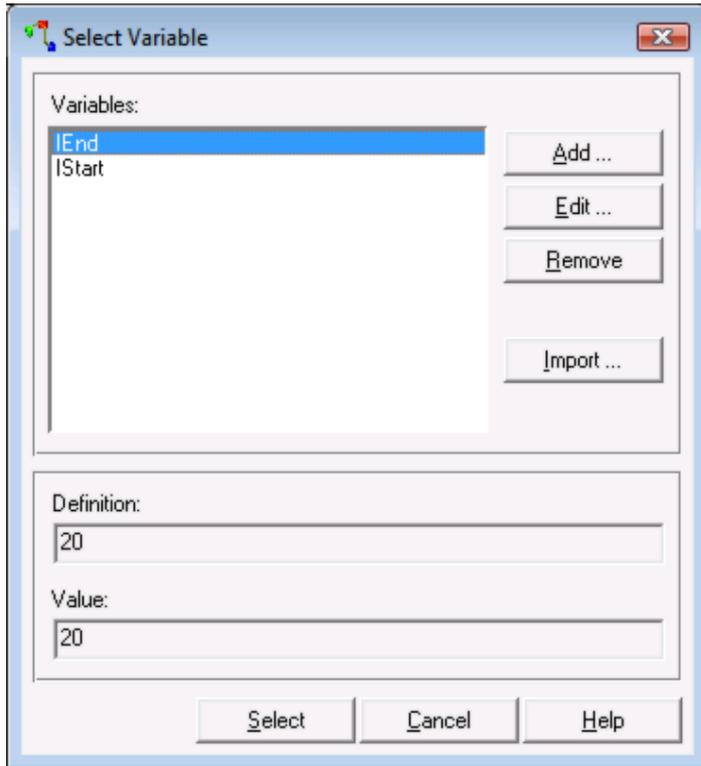
Special options are «variable» and «directory». Selecting either of these options starts the Variable Selection or the Directory Alias Selection dialog.

When the contents of the field with pre-defined options are analysed, the options are substituted by their values. Good examples are the command strings of user-defined data types as described in the User-defined Data Types section. For directories, the local directory path is substituted.

If a command string is passed to the system or to a DDE server an option value (e.g. a directory) containing blanks may cause problems. How this is handled is described in the Quote Script Elements and Quote Clever section.

Variable Selection

Most dialogs contain pre-defined option buttons . Pressing this button leads to a Pre-defined Option Selection dialog, which often includes the «variable» option. After selection of the «variable» option ArisFlow asks the user to select a variable from the variable selection dialog:



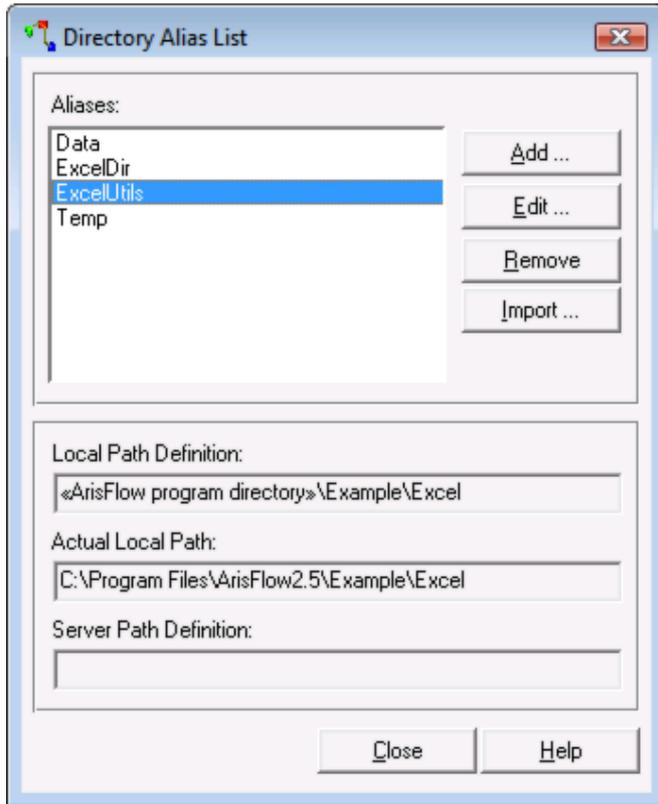
Example of a variable selection dialog.

After selecting a variable and pressing the *Select* button the variable is inserted into the pre-defined option its edit field. The selected variable appears in the script text as follows: «Var: alias», where the "Var:" is fixed and the alias is the name of the variable selected. This is the so-called variable notation.

If is possible to directly create, edit, remove and import [variables](#) from the variable selection dialog, without having to return to the Variable List dialog.

Directory Selection

Most dialogs contain pre-defined option buttons . Pressing this button leads to a Pre-defined Option Selection dialog, which often includes the «directory» option. After selection of the «directory» option ArisFlow asks the user to select a directory alias from the directory selection dialog:



Example of a directory selection dialog.

After selecting a directory alias and pressing the *Select* button the directory alias is inserted into the pre-defined option its edit field. The directory alias selected appears in the text as follows: «Dir: alias», where the "Dir:" is fixed and the alias is the name of the directory alias selected. This is the so-called directory alias notation.

It is possible to directly create, edit, remove and import [directory aliases](#) from the Directory Alias Selection dialog, without having to go to the Directory Alias List dialog.

Quote Script Elements and Quote Clever and Quote Clever

File and directory paths may contain blanks. What a great idea!? This means that where files and directories appear as parameters their paths should be enclosed by double quotes, otherwise the executing system thinks it is dealing with two or more parameters. Very confusing.

In ArisFlow many script parameters may require quotes when passed for execution in a command string or script file. These script elements could be Action Script parameters, parameters in command strings for System and DDE action types and parameters in command strings for File/Directory and User-defined data types.

For browser, checker and viewer commands in File/Directory and User-defined data types no options can be set in ArisFlow. This is not required: these commands are always sent to the operating system and ArisFlow thus implements the clever quoting option described below.

ArisFlow never encloses root directories or environments in the Directory Alias dialog or variables in any dialog in quotes.

For execution scripts two checkbox options define whether script elements are quoted and how script elements are quoted when an action is executed:

- *Quote Script Elements*;
- *Quote Clever*.

Both checkbox options occur:

- In the *System* dialog in the *Program*, *Shell Script* and *Action Script* sections;
- In the *DDE* dialog in the *Server Commands* and *Action Script* sections.

When any of these options is checked this only applies to that section. When the *Quote Clever* option is checked the *Quote Script Elements* option is checked as well.

When the *Quote Script Elements* option is checked it indicates that all script elements, except variables, will be enclosed by double quotes. These script elements are of course location paths of script files, control files, directory aliases, data, etc.

When the *Quote Clever* option is checked as well ArisFlow checks whether directories are part of a so-called extended path. These directories can be:

- Directory aliases included as script elements;
- Locations (.loc as script element type) of data or actions in action scripts. However user-defined data, where the location is not stored as a directory alias are not directories;
- Full paths (.full or default setting) for data of the directory base type.

Besides this directory the extended path may also include:

- Non-blank characters;
- Variable script elements;
- The first data entry (.entry as script element type) behind the extended path's directory.

The extended path is extended until the first blank character or script element that can not be part of the extended path. The double quotes are inserted immediately before this blank character or script element. Note that blanks, which are part of script elements included in the extended path, do not limit the extended path. Thus, if filenames including entries should be part of the extended path, it is an option to define them as variables, so that ArisFlow will put the quotes beyond the blank in that entry.

A directory separator (\ or sometimes /) is always present between the location and the extended path. If necessary ArisFlow inserts the separator when the quote clever option is set.

When creating new action types the *Quote Script Elements* and *Quote Clever* options are set for the *Program* section in the *System* dialog. This is because clever quoting is the logical quotation option for commands send to the operating system. All other quoting options are switched off initially.

Tooltips, Context Menus and Special Key Options in Dialogs

Tooltips are displayed in dialogs. The actual path of a script element is displayed when the cursor stops above this script element. This actual path is substituted for the script element when the script is passed for execution. Also when moving the cursor over directory alias comboboxes in any dialog or over data listboxes or *Program* buttons in the *Action Script* dialog, the actual path of the directory alias or the data is displayed.

In *System* and *Action Script* dialogs it is possible that server directories are used, because the *Use server script* checkbox may be set in the action type's *System* dialog for the *Shell script* or the *Action script* section. In that case the actual path consists of server directories, which then are displayed.

When clicking the right-mouse button a context menu is started. In an edit the options displayed depend upon whether the right-mouse button is clicked on a script element or not.

When the right-mouse button is pressed outside a script element standard options are displayed, such as Cut, Copy, Paste, Delete, Undo and Select All and it is possible to insert new script elements.

When the context menu is started inside a script element certain "standard" options acquire a special meaning. When the options Cut, Copy, Paste or Delete are selected the whole script element is cut, copied, pasted or deleted, not just the character that the right-mouse button is clicked on.

Certain combinations of keys pressed define special key-options in an edit field. The special key combinations available depend upon whether or not the cursor is located inside a script element. The special keys are all control keys, that means that they are activated by pressing the *ctrl* key and at the same time pressing another key. These combinations are:

Ctrl-keyName		Action
A	Select all	Select all text in the edit field
C	Copy	Copy selected text to paste buffer
D	Define	Press <i>Define</i> button
I	Tab	Inserts tab character
J	NewLine	Inserts newline character
O	Option selectionStart	pre-defined option selection dialog
S	Select	Extend selection
V	Paste	Insert paste buffer at cursor position
X	Cut	Remove selected text into paste buffer
Z	Undo	Undo last change

As with context menu options the selected text can be extended to include complete script elements with Copy, Select and Cut special keys. The Define special key combination (Ctrl-D) is only available in the Action Script dialog. The Option selection key (Ctrl-O) is equivalent to pressing the  button. Tabs and Newlines can only be inserted in edit boxes with multiple lines, and not in simple edit lines.

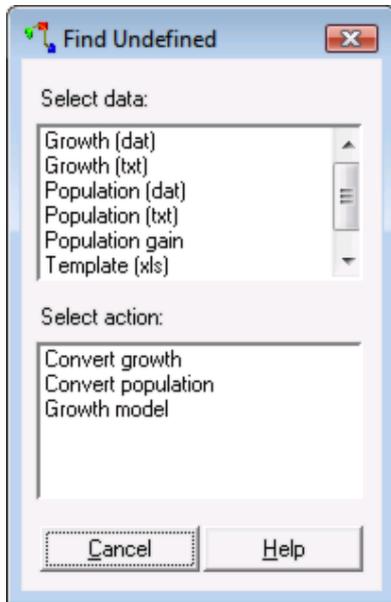
Find Dialogs

The *Edit/Find* menu has three options:

- [Edit/Find Action](#), for searching an action in the flowchart;
- [Edit/Find Data](#), for searching a data in the flowchart;
- [Edit/Find Undefined](#), for searching undefined actions and data (i.e. which are coloured grey)

in the flowchart.

When any of these menu options is selected a dialog window appears displaying all actions, all data or all undefined actions and data. For example when the [Edit/Find Undefined](#) menu option is selected the following dialog is shown:



Example of find undefined dialog.

When any of the listed actions or data is selected that action or data becomes selected in the flowchart, and the flowchart display focuses in on the selected action or data. When *Cancel* is pressed nothing happens.

Input=output Data

A special category of data is input=output data. This is where two data are connected to the same action, one as input data and one as output data, which:

- Have the same [datatype](#);
- Have the same actual path (the same location and entry).

Input=output data is treated as one and the same data. Input=output data are mainly used where existing data (e.g. a database) is updated by a certain action. It is not realistic, and therefore not recommended, to have three or more identical data connected to the same action, but ArisFlow does not prohibit it.

When drawing the flowchart it is not recommended to make the input data of the input=output data also the input data of another action. In that case it is unclear which action is executed first, making it unclear whether the other action is executed with the input=output data before or after the execution of their connecting action. Therefore it is better to only use the output data of the input=output data pair as input data for another action.

Input=output data have the following consequences:

- In the Action Script ArisFlow considers them to be the same data. Therefore if one of the input=output data is named in the action script the other is also considered to be used in the action script. In that case the connections between the action and both data are drawn with a solid line in the flowchart;

- After the action connecting both input=output data is executed the input data receives the same stamp as the output data. If this input data is output to another action that action remains up-to-date.

Finally it is noted certain effects which may appear strange are explained by the sometimes rather surprising behaviour of input=output data. In particular where data become input=output data or are no longer input=output data actions may suddenly become defined or undefined and connections may become defined (solid line) or undefined (dashed line). These effects can occur:

- When data are altered, making two data identical or not identical anymore;
- Where connections are drawn turning two data now connected to the same action into input=output data;
- Where directory alias paths are changed, making some data now have the same actual path (and thus input=output data) or not having the same path anymore.

Check Status

All data and actions are checked:

- When the user selects the *Execute/Check Status* menu option;
- When the user presses the  toolbar button.
- When opening a project that has the Status Checking On Output Data option checked in the Execution Preferences dialog.

As described in the State of Actions and Data section actions and data can have three states:

- *Undefined* actions and data appear grey in the flowchart;
- *Not-up-to-date* actions and data are red;
- *Up-to-date* actions and data are black.

Project input-data are either undefined (grey) or up-to-date (blue).

Data or actions, which are undefined, cannot be executed. The reasons why data or actions may be undefined are described in the State of Actions and Data section. To make data or action definitions defined it is recommended to:

- Check Data or Action dialogs to see whether all required fields were filled in.
- Check whether all data connected to an action were named in the Action Script. If a data is not named in the action script the connection between the data and the action appears as a dashed line in the flowchart.
- Check whether project input-data or action programs are present at their specified locations or whether other data can be generated (i.e. have existing locations).
- Perform a check status with the Trace File being written. The trace file defined in the Trace Preferences section describes the check status result of every flowchart object and why the action or data may be undefined.

Execution

It is possible to execute:

- All actions, by selecting the *Execute/All* menu option or pressing the  toolbar button.

- Only the actions currently selected by selecting the [Execute/Selected Actions](#) menu option or pressing the  toolbar button;
- All actions that should be executed in order to attempt to make all currently selected flowchart data and actions up-to-date by selecting the [Execute/Up To Selection](#) menu option.

The Execution of Action section describes that actions are executed only when the action and its output data are not-up-to-date and when all input data are up-to-date. Execution continues until no more actions can be executed. It is possible to execute actions multiple times by checking the Cyclic Integrity Checking option as described in the Execution Preferences dialog section.

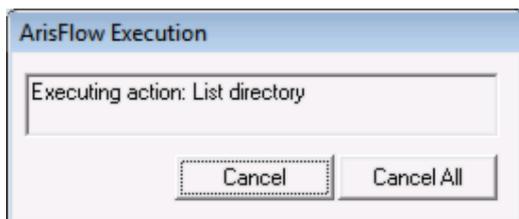
Actions are executed in the sequence set in the Execution Preferences dialog. This sequence is overruled if an action's parent action still requires execution. In that case the execution of the action is delayed until all its parent actions have been executed.

When an action is executing it appears yellow in the flowchart. The Execution Progress window displays the progress of the execution and allows the user to cancel the execution. During execution and when checking status ArisFlow goes into execution mode. This means it is possible to start any dialog in ArisFlow, but most fields cannot be updated.

When an action has executed correctly the action and all its output data become up-to-date (black colour). When the execution fails the action and its output data remain not-up-to-date (red colour). In that case the user has the occasionally difficult task of establishing what went wrong, and attempting to fix the problem. The following strategy is recommended:

- Check the message written to the screen. This message is usually clear enough in describing why the execution of the action failed;
- Check the last run in the Execution Log File. This page describes why the execution failed;
- Check the temporary script files written. When shell or script command lines are passed as script, the script is written to a temporary file. Temporary files are normally removed when not required anymore, unless an action fails. The names of the temporary files are written to the Execution Log File. If the *Remove temporary files* option is switched off (as described in the File Preferences section), temporary script files are never removed after completion of an action, even when ArisFlow detects no irregularities during the execution;
- Check the contents of the Trace File. It is likely that the execution should be re-run with the writing of the Trace File enabled, as described in the Trace Preferences section;
- Check the input data and output data. There may be irregularities in these data, e.g. an input file containing 0 bytes;
- Check the definition of the action types and the Action Script. In particular [Action Type Definitions](#) have many options where selection of the wrong option may cause problems.

Execution Progress



Execution progress showing the action currently being executed.

The Execution Progress window displays what ArisFlow currently is doing. This is usually the execution of a certain action or (sometimes) the status checking of certain data. This window is only displayed when the *Show Progress Info* checkbox, described in the Execution Preferences

section, is checked.

The execution progress window has two buttons:

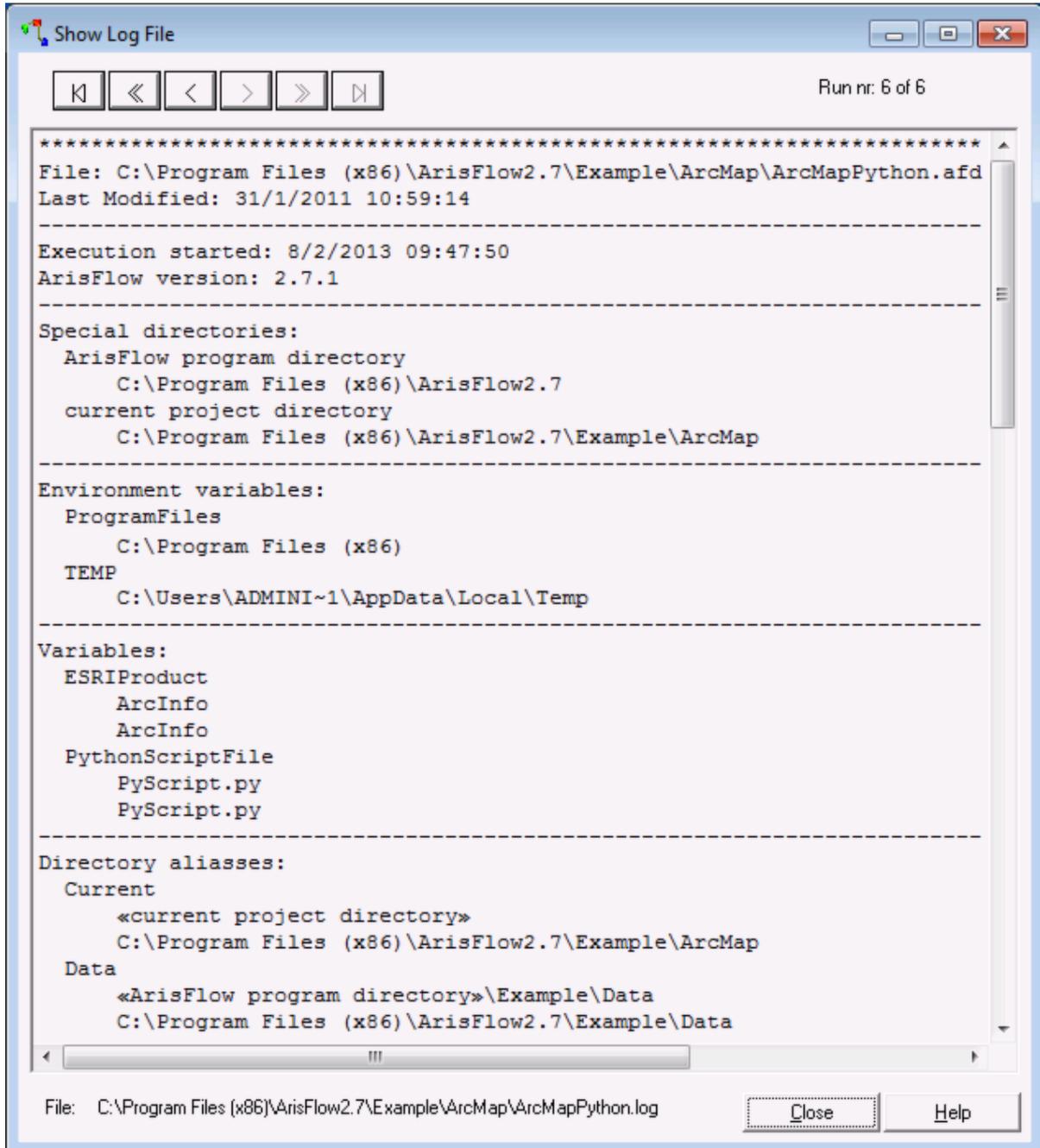
- **Cancel.** The action currently executing is cancelled. ArisFlow will attempt to continue the execution process with another action, if possible. When the cyclic integrity checking option is set in the **Execution Preferences** section, ArisFlow will not attempt to execute the cancelled action once again;
- **Cancel All.** The action currently executing is cancelled. ArisFlow will finish the execution process immediately.

When ArisFlow is executing or checking status it is in execution mode. In execution mode it is not possible to change any flowchart components and most dialogs edit fields.

When the execution of an action is cancelled, this will be written to the Execution Log File as the execution result.

Execution Log File

The [View/Log File](#) menu option displays the log file in a dialog window. The window is sizeable and read-only. For setting the execution log file name and options, the reader is referred to the **Filename Preferences** section.



Example of Execution log file, showing definitions of variables.

The log file describes the results of execution runs. When the user detects something unexpected during an execution run, the execution log file may be helpful in detecting what actually happened. It is also possible to Print execution log results or to Print these into an ASCII file.

Every execution run is written into the log file as one page. The buttons at the top of the Execution Log File window allow the user to go through the different execution run pages. These buttons indicate from left to right:



Move back to first page;



Move back rapidly (multiple pages);



Move back one page;



Move forward one page;



Move forward rapidly (multiple pages);



Move forward to last page. This is the result of the last execution run.

Of each execution the log file describes:

- Which project has been executed; i.e. the ArisFlow-project filename and the last modification date of the ArisFlow dataflow project file before the execution started;
- The version of the ArisFlow program that executed the flowchart;
- The time when the execution run was started;
- The time when the execution run was finished;
- A list of special directories (related to ArisFlow and the current project);
- A list of environment variables (name and path);
- A list of variables (name, definition and value);
- A list of directory aliases; (name, definition and path);
- A list of action types (name, base type, location definition, location value, entry definition, entry value, full path);
- A list of data types (name, base types, location definition, location value, entry definition, entry value, full path);
- Actions executed.

Of the actions executed the log file describes:

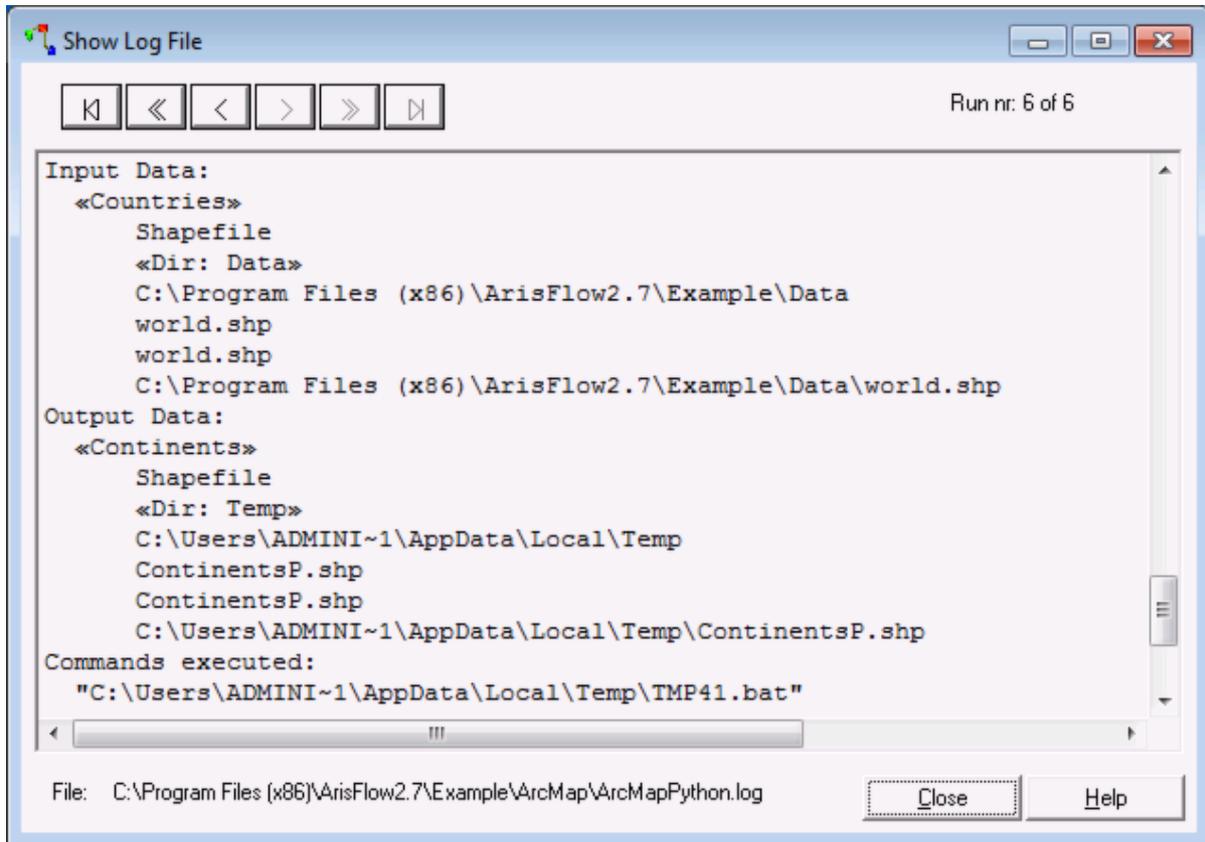
- Action logical name;
- Action type;
- Time when the execution of the action was started;
- Time when the execution of action was ended;
- Action script;
- Program definition (if applicable) with location and entry rows, both as Definition and as Actual Value
- Input data of the action;
- Output data of the action;
- The actual commands executed;
- Execution result;
- Names of temporary script files created by the action, if not removed. Of actions that were not completed successfully the temporary script files are never removed.

Seven characteristics are described of input and output data:

- Logical name of data;
- Data type;
- Location definition;
- Location value (path);
- Entry definition;
- Entry value;

- Full actual path;

In the action script the logical names of the data are used.



Example of Execution log file, showing input and output data.

Execution results are written to the log file. If execution fails, a message is also written to the screen. In the log file two parameters are described with each action execution result:

- How execution of the action was terminated;
- If the execution was not OK: the reason of the failure.

An action may be terminated in four different ways:

- OK. The action has been executed all right and is set up-to-date (black circle in the flowchart);
- Fail. ArisFlow has detected that something went wrong during the execution;
- Cancel. The user has pressed *Cancel* in the Execution Progress window during the execution of the action;
- A Time out. A command being executed or a DDE server being started has not returned a positive result after a pre-set time-out period had expired.

If the termination was not OK, the execution log file will describe why the result was not OK. In case the execution was cancelled or a time-out was reached, the log file displays what ArisFlow was attempting to execute at the moment the execution was stopped, and writes this to the log file.

Finally, when temporary files are not removed (an option in the File Preferences), the log file will name the temporary script files that were used in executing the action.

Execution Parsing

Execution Parsing

This section contains two chapters showing examples of the execution of MS-DOS actions, which highlight:

- **Command Lines and Script Files.** This chapter has examples showing which script files are written and which command lines are sent to the operating system for various settings in the *Shell Script* and *Action Script* sections of the System dialog;
- **Quoting.** This chapter has examples showing how script elements are enclosed in quotes for various settings of the Quote Script Elements and Quote Clever settings in the System dialog.

The flowcharts of both examples are included in the ArisFlow examples sub-directory.

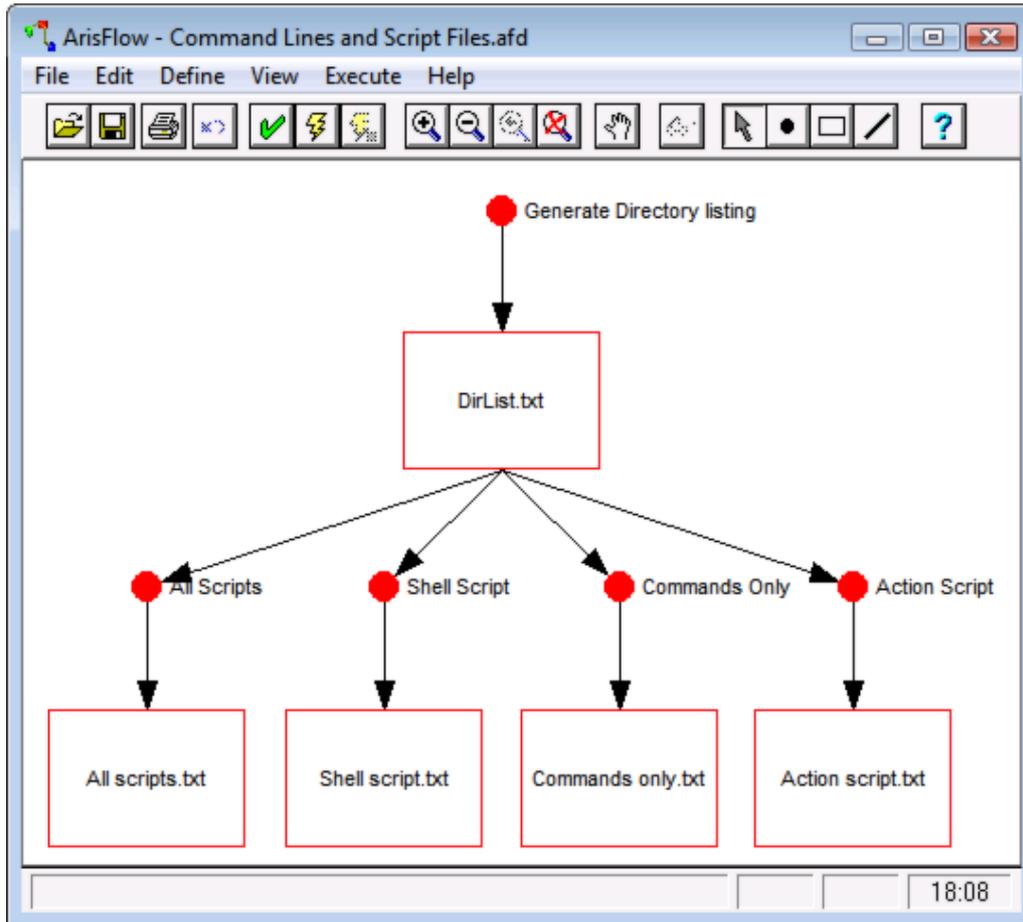
Command Lines and Script Files

This section illustrates the [Execution Parsing](#) mechanism of an action. The example "Command Lines and Script Files.afd", which is located in the "Example\DOS" directory is explained. This section describes which command lines are sent to the operating system in which order, and which lines are written to temporary scripts files.

Four situations with different settings in the actions type's [System](#) dialog are described:

- [Both shell script and action script written to script files;](#)
- [Shell script written to a script file, action script executed as command;](#)
- [Both shell script and action script executed as command;](#)
- [Shell script executed as command, action script written to a script file.](#)

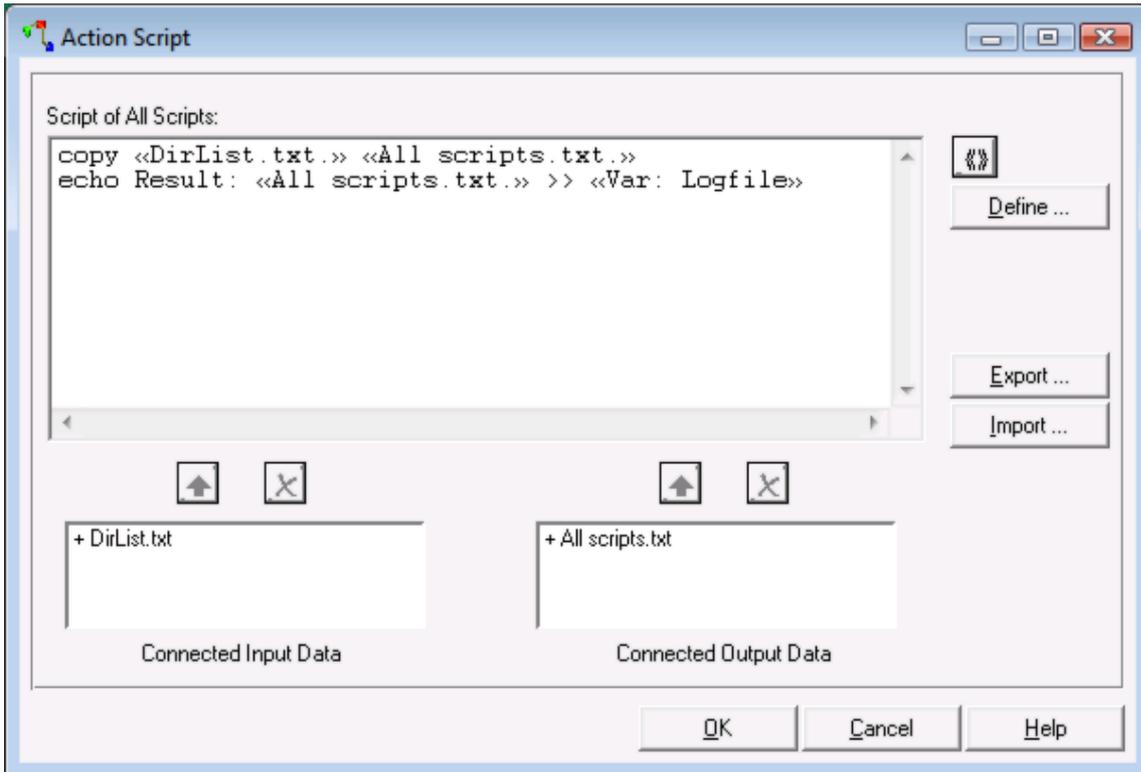
In all Command Lines and Script Files examples the same flowchart is used, which is the "ArisFlow Examples\DOS\Command Lines and Script Files.afd" flowchart:



Flowchart used in Command Lines and Script Files example

The action "Generate Directory Listing" is just a simple action which generates a directory listing and writes it to the "DirList.txt" file.

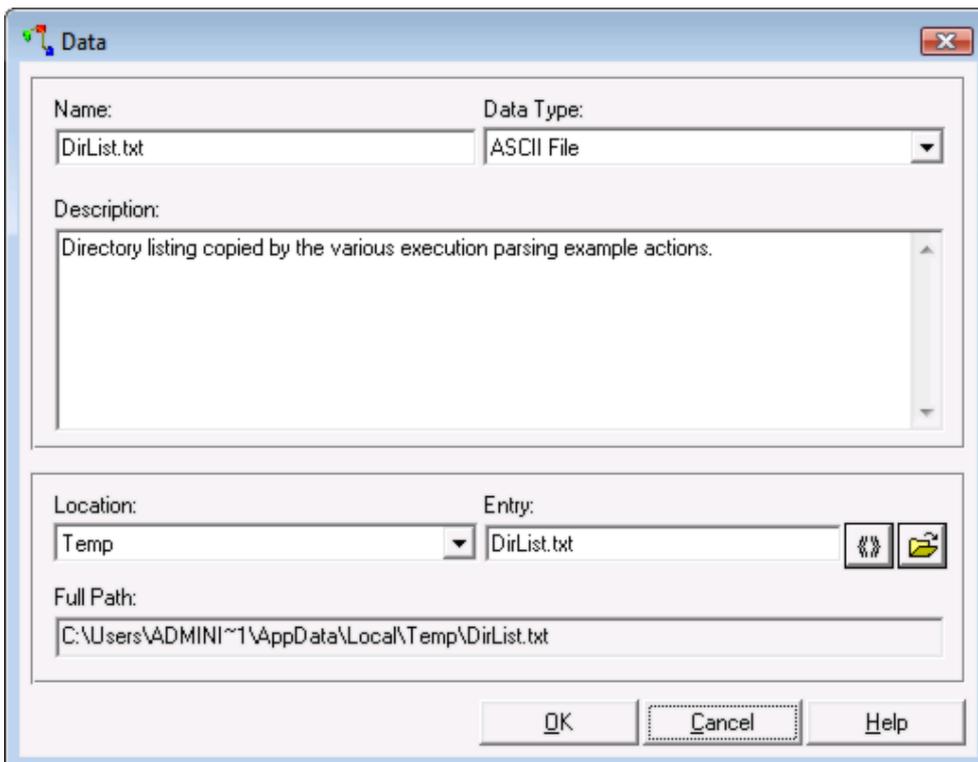
The action "All Scripts" copies the directory listing to the output file "All scripts.txt". After that it writes some kind of logging statement to a logfile. This logfile is defined as the variable "Logfile", which is set as as the quoted file path "«Env: TEMP»\outfiles.txt". The second line may look a bit peculiar, but it should illustrate the merits of this example very well. The [action script](#) of the action "All Scripts" is:



Action script of "All Scripts" action

The actions "Shell Script", "Commands Only" and "Action Script" have the same action script definition as the action "All Scripts". The only differences are the name of the output file and the action type.

The "DirList.txt" input [data](#) looks like:



Details of "DirList.txt" data

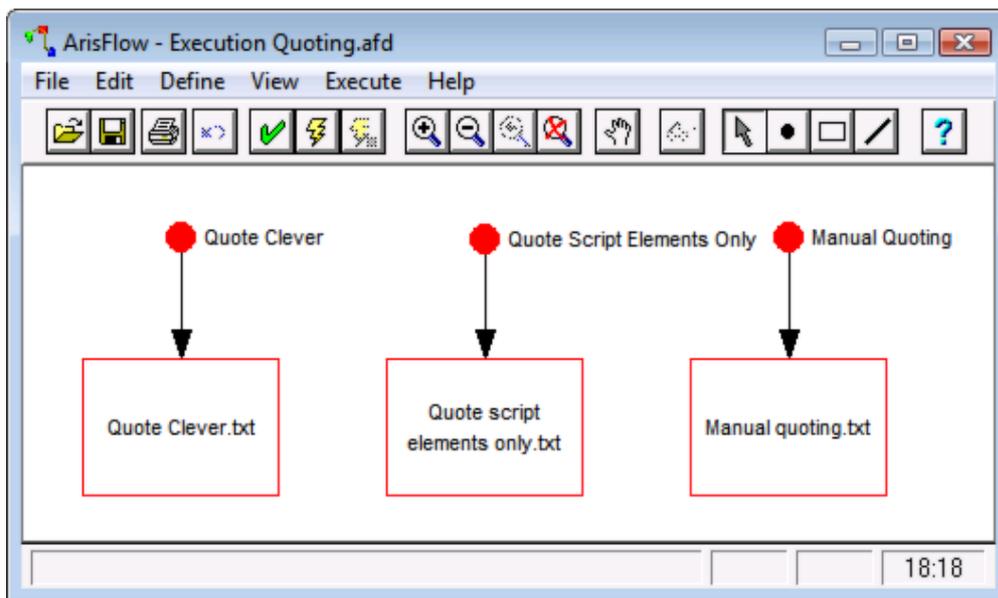
All output data are similar to the "DirList.txt" data. The location and entry of the data is in itself of course not that important, but will be of relevance when the exact commands passed for the various cases are written out.

Quoting

This chapter shows three Execution Parsing examples with different script element quoting settings:

- Quote script elements and quote clever set;
- Quote script elements only;
- No quoting set, quotes inserted manually.

The examples also illustrate extended paths and what happens to [directory](#) separators. The Quoting examples can be found in the "Execution Quoting.afd" flowchart, which is located in the "Example\DOS" directory.



Flowchart used in Quoting examples

These examples concentrate around the shell script definition of each action type. The action scripts and output data are not very relevant and are not displayed here. However the following definitions are important:

- The [variable](#): "dirlist entry", which has the value "Dir list";
- The TEMP directory, which is "c:\temp";
- The [directory alias](#): "ArisFlow directory", which represents the directory "c:\program files\AF".

ArisFlow as DDE Server

ArisFlow as DDE Server

This chapter has the following useful main parts:

- How to interact with ArisFlow through DDE commands.
- ArisFlow Commander.
- ArisFlow DDE commands list.

The next chapter describes the ArisFlow Commander. This is a separate program, installed together with ArisFlow, through which server commands can be sent to ArisFlow using a script file.

All DDE commands are described in detail in the following sections:

- Handling errors occurring during DDE conversation;
- Managing the ArisFlow project;
- Managing the appearance of ArisFlow;
- Executing and checking the status of (parts of) the flowchart;
- Requesting the results of the last or the current execution run;
- Retrieving information about the flowchart and its components;
- Changing flowchart parameters.

Finally there is a list of server commands recognised in ArisFlow versions 1.0.1 and 2.0. ArisFlow still recognises these obsolete commands until versions 2.2 and 3.0 respectively.

How to interact with ArisFlow through DDE commands

ArisFlow can act as a DDE server. A client can access ArisFlow using the Dynamic Data Exchange (DDE) protocol. This protocol provides a mechanism to send DDE commands to ArisFlow. These commands can be:

- **Transactions.** A transaction makes ArisFlow perform certain activities.
- **Requests.** A request attempts to retrieve data from ArisFlow. If the request is successful ArisFlow will return the requested data to the client.

Before sending any transactions or requests to ArisFlow the client should start a DDE conversation with the ArisFlow server by providing the server name and topic to ArisFlow. The server name should always be *ArisFlow*. The topic is capital insensitive and can be any of the following:

- *System*. Any ArisFlow process;
- *New*. Any process with a new and unchanged flowchart. This topic is useful when testing whether a new flowchart has started up, and can be used by the ArisFlow Commander (see ArisFlow Commander Topic Handling section);
- The full path of the flowchart with forward and backward directory separators. Any ArisFlow process with the specified flowchart loaded (e.g. "D:\flowchart\myproject.afd", but then also "D:/flowchart/myproject.afd");
- The entry of the flowchart path. Any ArisFlow process with the specified flowchart loaded (e.g. "myproject.afd").

Therefore every new and unchanged flowchart responds when the topic is *New* or *System*. A

new flowchart, that has been altered only responds to the *System* topic. A flowchart, that exists on disk (or has been saved) responds when the topic is *System*, or the full path or entry of the flowchart. Note that only one instance of ArisFlow will act on the commands send, even when more than one ArisFlow process would answer to the specified server name and topic conversation.

If ArisFlow is executing and accepts the topic it will acknowledge the conversation to the client. After this acknowledgement the client can communicate with that ArisFlow project through the DDE commands described in the ArisFlow DDE Command List section.

Commands send to ArisFlow and request results returned by ArisFlow are always character strings. Commands are enclosed between [and] brackets and are capital insensitive. Some commands require that parameters be passed with them. The parameter list should be enclosed between (and) brackets and (if required) separated by commas. If no parameters are passed the (and) brackets are optional. A parameter may be enclosed by double quotes ("). This is required where confusion occurs, which is where a comma could be interpreted as parameter separator, or where extra blanks at the beginning or at the end of the parameter are required. Double quotes are not required in the rare situation where a variable used in a parameter contains commas; the commander will put quotes around the parameter in the command send to ArisFlow.

After it has received a transaction command or a request ArisFlow will attempt to start the command and acknowledge it has done so. The acknowledgement is invalid if:

- The command or request has not been send as a character string.
- ArisFlow is currently busy executing another command.

If ArisFlow is busy it will return an "invalid" acknowledgement. The client should then attempt to perform the same command after a short waiting period, in order to give ArisFlow time to finish what it is currently doing. In particular opening a file and checking the flowchart status are activities which make that ArisFlow could be busy for a while, thus delaying the acceptance of consecutive commands.

After an [ExecuteAll], [ExecuteAction], [ExecuteUpToAction] or [ExecuteUpToData] command was passed ArisFlow will immediately reply with a valid acknowledgement of the command and the start the execution. The advantage of this rapid acknowledgement is that ArisFlow is then still available for monitoring the execution progress through commands such as the [GetFlowchartExecuteStatus] request. ArisFlow can handle these requests during the execution process. It is recommended to check whether the execution of the ArisFlow flowchart has finished through these execution-monitoring requests before sending consecutive commands to ArisFlow.

An example of a client, which can send DDE commands to ArisFlow, is the ArisFlow Commander utility, which is described in a next section.

ArisFlow DDE Commands

ArisFlow DDE Command List

The following table lists all server commands and possible parameters. The type is a code, which means:

T Transaction
R Request

Optional parameters are inclosed between { and } characters. Detailed information of each command is provided in later sections.

Type	Command	Parameters	Alternative command
T	AppendActionDescription	(ActionName,Description)	
T	AppendDataDescription	(DataName,Description)	
T	AppendDirAliasDescription	(AliasName,Description)	
T	AppendVariableDescription	(VariableName,Description)	
T	AppendFlowchartDescription	(Description)	
T	AppendToFile	(FilePath,Text, {NrNewLine})	
T	CancelExecute		
T	CheckStatus		
T	CreateDirAlias	(AliasName,LocalPath)	
T	CreateVariable	(VariableName,Value)	
T	ExecuteAction	(ActionName)	
T	ExecuteAll		
T	ExecuteUpToAction	(ActionName)	
T	ExecuteUpToData	(DataName)	
R	GetActionCount		
R	GetActionDescription	(ActionName)	
R	GetActionExecuteEndTime	(ActionName)	
R	GetActionExecuteStartTime	(ActionName)	
R	GetActionExecuteStatus	(ActionName)	
R	GetActionName	(Index)	
R	GetActionStatus	(ActionName)	
R	GetActualPath	(PathDefinition)	
R	GetArisFlowPath		
R	GetDataCount		
R	GetDataDescription	(DataName)	
R	GetDataDirAlias	(DataName)	
R	GetDataEntry	(DataName)	
R	GetDataFullPath	(DataName)	

R	GetDataLocation	(DataName)	
R	GetDataName	(Index)	
R	GetDataStatus	(DataName)	
R	GetDirAliasCount		
R	GetDirAliasDescription	(AliasName)	
R	GetDirAliasName	(Index)	
R	GetDirAliasLocalPath	(AliasName)	GetDirAliasPath
R	GetDirAliasLocalPathDefinition	(AliasName)	GetDirAliasPathDefinition
R	GetDirAliasNotation	(AliasName)	
R	GetDirAliasServerPath	(AliasName)	
R	GetDirAliasServerPathDefinition	(AliasName)	
R	GetEnvironmentNotation	(Environment)	
R	GetErrorMessage		
R	GetFlowchartDescription		
R	GetFlowchartExecuteEndTime		
R	GetFlowchartExecuteStartTime		
R	GetFlowchartExecuteStatus		
R	GetFlowchartStatus		
R	GetFlowchartTitle		
R	GetLogfileName		
R	GetProgramDirAlias	(ActionName)	
R	GetProgramEntry	(ActionName)	
R	GetProgramFullPath	(ActionName)	
R	GetProgramLocation	(ActionName)	
R	GetProjectPath		
R	GetVariableCount		
R	GetVariableDescription	(VariableName)	
R	GetVariableName	(Index)	
R	GetVariableNotation	(VariableName)	
R	GetVariableValue	(VariableName)	
R	GetVariableValueDefinition	(VariableName)	
T	MessageBox	(Title,Text)	
T	Open	(ProjectName)	
T	Quit		
T	RelocateDirAliasLocalPath	(AliasName,LocalPath)	RelocateDirAliasPath
T	RelocateDirAliasServerPath	(AliasName,ServerPath)	
T	RelocateVariableValue	(VariableName,Value)	
T	Save		
T	SaveAs	(ProjectName)	

T	SetActionDescription	(ActionName,Description)
T	SetAppMaximise	
T	SetAppMinimise	
T	SetAppRestore	
T	SetAppShowOff	
T	SetAppShowOn	
T	SetAppTopOff	
T	SetAppTopOn	
T	SetDataDescription	(DataName,Description)
T	SetDataDirAlias	(DataName,AliasName)
T	SetDataEntry	(DataName,Entry)
T	SetDataFullPath	(DataName,FullPath)
T	SetDataLocation	(DataName,Location)
T	SetDDEDebugFilename	(FileName)
T	SetDDEDebugOff	
T	SetDDEDebugOn	
T	SetDirAliasDescription	(AliasName,Description)
T	SetDirAliasLocalPath	(AliasName,LocalPath) SetDirAliasPath
T	SetDirAliasServerPath	(AliasName,ServerPath)
T	SetFlowchartDescription	(Description)
T	SetFlowchartTitle	(Title)
T	SetLogfileName	(FileName)
T	SetProgramDirAlias	(ActionName,AliasName)
T	SetProgramEntry	(ActionName,Entry)
T	SetProgramFullPath	(ActionName,FullPath)
T	SetProgramLocation	(ActionName,Location)
T	SetVariableDescription	(VariableName,Description)
T	SetVariableValue	(VariableName,Value)
T	SetWindowPosition	(X,Y)
T	SetWindowSize	(Width,Height)
T	ShowErrorMessagesOff	
T	ShowErrorMessagesOn	
T	ShowProgressInfoOff	
T	ShowProgressInfoOn	
T	WriteToFile	(FilePath,Text, {NrNewLine})
T	ZoomAll	

Handling errors occurring during the DDE conversation

During a DDE conversation errors can occur. This could be because commands are not recognised by the ArisFlow server or because command parameters are incorrect. ArisFlow has three mechanisms to notify DDE errors:

- ArisFlow can display error messages on the screen. This is not the most convenient error handling method (the user has to press "OK" for every message). Normally messages are not displayed. The [\[ShowErrorMessageOn\]](#) command switches on the message display;
- The client can enquire the result of the last DDE command handled ArisFlow through the [\[GetErrorMessage\]](#) request;
- DDE commands and requests received by ArisFlow can be written to the DDE debug file. This is an ASCII text file. Results returned by requests as well as errors detected by ArisFlow are also written to this DDE debug file. Its filename is declared by the [\[SetDDEDebugFilename\]](#) command. After the [\[SetDDEDebugOn\]](#) command ArisFlow starts writing to the DDE debug file.

The following commands implement these error-handling mechanisms:

[GetErrorMessage]

Returns possible error result in the command or request passed prior to this request. If no error occurred an empty string ("") is returned. Otherwise the error message is returned.

[SetDDEDebugFilename(FileName)]

The DDE debug file name is set to FileName. The FileName may contain an environment variable. This command should always be passed before any [\[SetDDEDebugOn\]](#) command can be passed to the server.

An Error is generated:

- When no file with FileName could be generated (i.e. when its directory does not exist).
- When a non-existing environment variable is used.

[SetDDEDebugOff]

Makes that no more information is written to the DDE debug file, until the [\[SetDDEDebugOn\]](#) command is passed. The current processor time is written to the DDE debug file.

[SetDDEDebugOn]

Starts with writing commands passed to an ASCII file. The file has the filename, which was set by the [\[SetDDEDebugFilename\]](#) command. In the file will be written:

- Every transaction and request which is passed to the ArisFlow server;
- For requests: the result that was returned to the client;
- Possible errors that were detected in the command that was passed.

Request results and errors are indented by one tab position. The string "Returned:" is written before the request result. The string "Error:" is written before a possible error message.

Every time when information is written to the DDE debug file this file is opened and the information is appended to the file. The debug file is closed immediately after the information is written, making it possible to examine its contents even when the DDE process performs not as expected.

Each time the [\[SetDDEDebugOn\]](#) command is passed a header is written to the DDE debug file. This header includes the current time. The [\[SetDDEDebugOff\]](#) command or the termination of ArisFlow ensures that no more information is written to the DDE debug file.

[ShowErrorMessageOn]

Makes ArisFlow display message boxes. These messages mainly involve errors detected in DDE commands received by ArisFlow and errors detected during the execution of an ArisFlow project. Usually it is not very useful to display messages when ArisFlow is managed through server commands. The only exception may be during the execution of flowchart actions. In that case the client could pass a `[ShowErrorMessagesOn]` command immediately before the execute command is passed to ArisFlow, and a [\[ShowErrorMessagesOff\]](#) directly after the execute command is passed to ArisFlow.

[ShowErrorMessagesOff]

ArisFlow will not display any message boxes. This is the normal setting. Compare [\[ShowErrorMessagesOn\]](#).

When passing the `[ShowErrorMessagesOff]` command it is recommended to pass this after every [\[Open\]](#) command, to ensure that every project opened does not display error messages.

Managing the ArisFlow project

The ArisFlow project is managed by the following commands:

[AppendToFile(FilePath,Text,NrNewLine)]

Appends Text to file with FilePath. The FilePath may contain an environment variable in environment notation, which can be derived using the [\[GetEnvironmentNotation\]](#) request. If the file with FilePath does not yet exist a new file is created. A new file is written using the [\[WriteToFile\]](#) command. NrNewLine is optional. If a value is passed that number of newline characters is written after the Text is written to the file.

An Error is generated:

- When the file cannot be written (i.e. when the directory in FilePath does not exist);
- When the FilePath contains a not-existing environment.

[Open(ProjectName)]

Opens ArisFlow project with the input project name. The project name may include an environment variable in environment notation, which can be derived using the [\[GetEnvironmentNotation\]](#) request. If ProjectName has no extension ".afd" is taken as the extension for the project file. All open dialogs are closed by ArisFlow. If the [\[ShowErrorMessagesOff\]](#) command was passed before the `[Open]` command the [Undefined Environments](#) and Directory Alias Repair dialogs are never shown.

An Error is generated:

- When the project file cannot be opened;
- When the ProjectName path contains a not-existing environment.

[MessageBox(Title,Text)]

Displays message box with title and text. The ArisFlow user should press the "OK" button for acknowledgement of the message box command and continuation of ArisFlow. The title is optional. A standard title is displayed when the message box command has only one single parameter. Quotes are not required, except for when the Title or the Text contains commas, which could be mistaken for parameter separators.

[Save]

Saves the ArisFlow project, into the current project file.

[SaveAs(ProjectName)]

Saves the ArisFlow project, into the project file with ProjectName. The project name may include an environment variable in environment notation, which can be derived using the [\[GetEnvironmentNotation\]](#) request. If ProjectName has no extension ".afd" is taken as the extension for the project file.

An Error is generated:

- When the project file cannot be written (i.e. when the directory in ProjectName does not exist).
- When a non-existing environment variable is used.

[Quit]

Exit from ArisFlow without saving any changes. Use [\[Save\]](#) before the [Quit] command if changes should be saved.

[WriteToFile(FilePath,Text,NrNewLine)]

Writes Text to file with FilePath. The FilePath may contain an environment variable in environment notation, which can be derived using the [\[GetEnvironmentNotation\]](#) request. If the file with FilePath does already exist it is overwritten. The text can be appended to a file using the [\[AppendToFile\]](#) command. NrNewLine is optional. If a value is passed that number of newline characters is written after the Text is written to the file.

An Error is generated:

- When the file cannot be written (i.e. when the directory in FilePath does not exist);
- When the FilePath contains a not-existing environment.

Managing the appearance of ArisFlow

The client can manage the appearance of ArisFlow. The following commands are available:

[SetAppMaximise]

Makes ArisFlow cover the entire screen. The original size is restored by the [\[SetAppRestore\]](#) command.

[SetAppMinimise]

Minimises the ArisFlow display. The original size is restored by the [\[SetAppRestore\]](#) command.

[SetAppRestore]

Resets the ArisFlow display to its original size.

[SetAppShowOff]

ArisFlow runs in the background. The ArisFlow application is not shown on the screen.

[SetAppShowOn]

ArisFlow is shown on the screen.

[SetAppTopOff]

When showing the ArisFlow appearance may (partly) be obscured by other applications.

[SetAppTopOn]

When showing ArisFlow always appears as the top-most application on the screen.

[SetWindowPosition(X, Y)]

The top-left corner of the ArisFlow application appears at the input (X,Y) pixel coordinates of the screen.

An [Error](#) is generated:

- When either X or Y is not 0 or a positive integer number.

[SetWindowSize(Width, Height)]

The ArisFlow display (which includes menu display, scrollbars, status line) is resized to input Width and Height.

An [Error](#) is generated:

- When either Width or Height is a positive integer number.

[ZoomAll]

Makes whole flowchart fit into the screen display of ArisFlow.

Executing or checking the status of (parts of) the flowchart

The client can initiate execution or checking the status of some or all actions in the flowchart by using the following commands:

[CancelExecute]

Cancels the execution of the flowchart.

An Error is generated:

- When the flowchart was not executing.

[CheckStatus]

Checks the status of all flowchart elements.

[ExecuteAction(ActionName)]

Executes the action with ActionName. The progress of the execution of the flowchart can be monitored through the [\[GetActionExecuteStatus\]](#) request.

An Error is generated:

- When no action with ActionName exists.

[ExecuteAll]

Executes all actions (if possible). The progress of the execution of the flowchart can be monitored through the [\[GetFlowchartExecuteStatus\]](#) request.

[ExecuteUpToAction(ActionName)]

Executes action with ActionName and all its parent actions, thus attempting to make the ActionName action up-to-date. The progress of the execution of the flowchart can be monitored through the [\[GetActionExecuteStatus\]](#) request.

An Error is generated:

- When no action with ActionName exists.

[ExecuteUpToData(DataName)]

Executes all parent actions of the data with DataName, thus attempting to make the DataName data up-to-date. The progress of the execution of the flowchart can be monitored through the [\[GetActionExecuteStatus\]](#) request.

An Error is generated:

- When no data with DataName exists.

[ShowProgressInfoOff]

During execution or status checking the progress (usually the action executed or data checked) of the execution or data checking is not displayed in a window. This flag is linked to the *Show Progress Info* checkbox in the Execution Preferences section.

[ShowProgressInfoOn]

During execution or status checking the progress (usually the action executed or data checked) of the execution or data checking is displayed in a separate window. This flag is linked to the *Show Progress Info* checkbox in the Execution Preferences section.

Requesting the results of the last or the current execution run

The client can retrieve the results of the last execution run of ArisFlow. If ArisFlow is currently executing the results of the execution run are returned. Results for individual actions and for the whole flowchart can be requested. These requests are:

[GetActionExecuteEndTime(ActionName)]

Returns the time at which the execution of the action with input ActionName was finished. Returns an empty string ("") if this action was not executed during the last execution run or if the action is still executing.

[GetActionExecuteStartTime(ActionName)]

Returns the time at which the execution of the action with input ActionName was started. Returns an empty string ("") if this action was not executed during the last execution run.

[GetActionExecuteStatus(ActionName)]

Returns the result of the last execution run for the action with input ActionName, which is useful in monitoring the progress of the [\[ExecuteAction\]](#) command. The result of this request can be:

- *None*. The action was not executed during the last execution run;
- *OK*. The action was executed and its status was set to up-to-date;

- *Error*. During the execution of the action ArisFlow detected something went wrong. The error can be retrieved from the Execution Log File;
- *Cancelled*. The user pressed the "cancel" button during the execution of the action;
- *Executing*. The action is currently being executed;
- Empty string (""). No action with input action name exists.
An Error is generated:
- When no action with ActionName exists.

[GetFlowchartExecuteEndTime]

Returns the time at which the last execution of the flowchart was finished. Returns an empty string ("") if the flowchart was never executed or if the flowchart is currently executing.

[GetFlowchartExecuteStartTime]

During flowchart execution this request returns the time at which the execution was started. Otherwise the time at which the last execution run of the flowchart was started is returned. Returns an empty string ("") if the flowchart was never executed yet.

[GetFlowchartExecuteStatus]

Returns the result of the last (or current) execution run. This is useful when monitoring the progress of the [\[ExecuteAll\]](#), [\[ExecuteUpToAction\]](#) and [\[ExecuteUpToData\]](#) commands. The result of this request can be:

- *None*. The flowchart has never been executed yet;
- *OK*. ArisFlow is currently not executing. All actions that were attempted for execution during the last execution run executed correctly;
- *Error*. ArisFlow is currently not executing. One or more actions attempted for execution did not execute correctly (i.e. GetActionExecuteStatus would return Error of Cancelled for one or more actions);
- *Executing*. The flowchart is currently being executed.

[GetLogfileName]

Returns the name of the log-file for the project execution results.

[SetLogfileName(fileName)]

Results of execution runs will be written to the log file with fileName. The filename definitions could include a directory alias, written in the directory alias notation by using the [\[GetDirAliasNotation\]](#) request. If no directory alias is included the location field of the log file will not be assigned and therefore the log file will be written to the work directory.

An Error is generated:

- When no file with fileName could be generated (i.e. when the directory of fileName does not exist).

Retrieving information about the flowchart and its components

Information about the flowchart and its components is retrieved by the following requests:

[GetActionCount]

Returns the number of actions in the flowchart.

[GetActionDescription(ActionName)]

Returns the description field value of the action with ActionName. Descriptions with multiple lines are returned with a single end-of-line (ASCII character 13) character separating two lines. The [SetActionDescription] command enables the server to alter the description of the action.

An Error is generated:

- When no action with ActionName exists.

[GetActionName(Index)]

Returns the name of the Index-th action in the flowchart. Index should be a value between 1 and the number of actions in the flowchart, which can be retrieved by the [GetActionCount] request.

An Error is generated:

- When Index is outside the specified range.

[GetActionStatus(ActionName)]

Returns the status of the action with ActionName as described in the State of Action and Data section. Possible results are:

- UpToDate;
- NotUpToDate;
- Undefined;
- Executing. The action is currently executing;
- Empty string (""). No action with ActionName exists.

An Error is generated:

- When no action with ActionName exists.

[GetActualPath(PathDefinition)]

- Returns the actual path for the path definition passed as parameter. The path definition may contain environments in environment notation, variables in variable notation and directory aliases in directory alias notation. The actual path does not have to exist.

An Error is generated:

- When PathDefinition is not correct.

[GetArisFlowPath]

- Returns the actual path of the ArisFlow base directory. This directory is the same as the directory in «ArisFlow program directory», which can be filled in as root directory of the local path of directory aliases.

[GetDataCount]

Returns the number of data objects in the flowchart.

[GetDataDescription(DataName)]

Returns the description field value of the data with DataName. Descriptions with multiple lines are returned with a single end-of-line (ASCII character 13) character separating two lines. The [SetDataDescription] command enables the server to alter the description of the data.

An Error is generated:

- When no data with DataName exists.

[GetDataDirAlias(DataName)]

Returns the name of the directory alias that is the location of the data object with DataName. If the data has no directory alias assigned to it an empty string ("") is returned.

An Error is generated:

- When no data with DataName exists.

[GetDataEntry(DataName)]

Returns the value of the entry field for the data object with DataName. If the data has no entry assigned to it an empty string ("") is returned.

An Error is generated:

- When no data with DataName exists.

[GetDataFullPath(DataName)]

Returns the full path for the data object with DataName. If the data has no datatype assigned an empty string ("") is returned.

An Error is generated:

- When no data with DataName exists.

[GetDataLocation(DataName)]

Returns the actual location of the data object with DataName. This is either the directory path for files and directories or the location string for user-defined data. If the data has no location assigned to it an empty string ("") is returned.

An Error is generated:

- When no data with DataName exists.

[GetDataName(Index)]

Returns the name of the Index-th data in the flowchart. Index should be a value between 1 and the number of actions in the flowchart, which can be retrieved by the [GetDataCount] request.

An Error is generated:

- When Index is outside the specified range.

[GetDataStatus(DataName)]

Returns the status of the data with DataName as described in the State of Action and Data section. Possible results are:

- UpToDate;
- NotUpToDate;
- Undefined;
- Empty string (""). No data with DataName exists.

An Error is generated:

- When no data with DataName exists.

[GetDirAliasCount]

Returns the number of directory aliases defined in the project.

[GetDirAliasDescription(AliasName)]

Returns the description field value of the directory alias with AliasName. Descriptions with multiple lines are returned with a single end-of-line (ASCII character 13) character

separating two lines. The [SetDirAliasDescription] command enables the server to alter the description of the directory alias.

An Error is generated:

- When no directory alias with AliasName exists.

[GetDirAliasName(Index)]

Returns the name of the Index-th directory alias in the project. Index should be a value between 1 and the number of directory aliases in the project, which can be retrieved by the [GetDirAliasCount] request.

An Error is generated:

- When Index is outside the specified range.

[GetDirAliasLocalPath(AliasName)]

Returns the actual path of the directory alias with AliasName. Is equal to the command [GetDirAliasPath(AliasName)].

An Error is generated:

- When no directory alias with AliasName exists.

[GetDirAliasLocalPathDefinition(AliasName)]

Returns the local path definition, as entered in the Directory Alias dialog, of the directory alias with AliasName. Is equal to the command [GetDirAliasPathDefinition(AliasName)].

An Error is generated:

- When no directory alias with AliasName exists.

[GetDirAliasNotation(AliasName)]

Returns a string that represents the directory alias AliasName in directory alias notation. This string could be used as (part of) a path parameter in commands such as [SetDirAliasLocalPath], [SetDirAliasServerPath], [SetDataFullPath], [SetProgramFullPath] and [SetLogfilename].

An Error is generated:

- When no directory alias with AliasName exists.

[GetDirAliasServerPath(AliasName)]

Returns the server path of the directory alias with AliasName.

An Error is generated:

- When no directory alias with AliasName exists.

[GetDirAliasServerPathDefinition(AliasName)]

Returns the server path definition, as entered in the Directory Alias dialog, of the directory alias with AliasName.

An Error is generated:

- When no directory alias with AliasName exists.

[GetEnvironmentNotation(Environment)]

Returns a string that represents Environments in environment notation. This string could be used as (part of) a path parameter in [SetDirAliasLocalPath], or as part of the full path in the [Open], [SaveAs], [AppendToFile] and [WriteToFile] commands.

An Error is generated:

- When Environment does not exist.

[GetFlowchartDescription]

Returns the description field value of the project, as described in the **Properties** section. Descriptions with multiple lines are returned with a single end-of-line (ASCII character 13) character separating two lines. The **[SetFlowchartDescription]** command enables the server to alter the description of the flowchart description.

[GetFlowchartTitle]

Returns the title field value of the project, as described in the **Properties** section. The **[SetFlowchartTitle]** command enables the server to alter the description of the flowchart description.

[GetFlowchartStatus]

Returns the status of the flowchart. Possible results are:

- *UpToDate*. All actions and data are up-to-date;
- *Executing*. The flowchart is currently executing;
- *NotUpToDate*. Flowchart is not executing. No actions or data are undefined. One or more actions or data are **not-up-to-date**;
- *Undefined*. Flowchart is not executing. One or more actions or data are undefined;
- *Empty*. Flowchart has no actions or data.

[GetProgramDirAlias(ActionName)]

Returns the alias name of the directory alias, which is the location of the action object with ActionName. If the action has no directory alias assigned to it an empty string ("") is returned.

An Error is generated:

- When no action with ActionName exists;
- When the action type is not a program.

[GetProgramEntry(ActionName)]

Returns the value of the entry field for the program action with ActionName. If the program has no entry assigned to it an empty string ("") is returned.

An Error is generated:

- When no action with ActionName exists;
- When the action type is not a program.

[GetProgramFullPath(ActionName)]

Returns the full path of the program action with ActionName.

An Error is generated:

- When no action with ActionName exists;
- When the action type is not a program.

[GetProgramLocation(ActionName)]

Returns the name of the directory alias for the program action with ActionName. If the program has no location assigned to it an empty string ("") is returned.

An Error is generated:

- When no action with ActionName exists;
- When the action type is not a program.

[GetProjectPath]

Returns the actual path of the project directory. This directory is the same as the directory in «current project directory», which can be filled in as root directory of the local path of directory aliases.

[GetVariableCount]

Returns the number of variables defined in the project.

[GetVariableDescription(VariableName)]

Returns the description field value of the variable with VariableName. Descriptions with multiple lines are returned with a single end-of-line (ASCII character 13) character separating two lines. The [SetVariableDescription] command enables the server to alter the description of the variable.

An Error is generated:

- When VariableName is not an existing variable.

[GetVariableName(Index)]

Returns the name of the Index-th variable in the project. Index should be a value between 1 and the number of variables in the project, which can be retrieved by the [GetVariableCount] request.

An Error is generated:

- When Index is outside the specified range.

[GetVariableNotation(VariableName)]

Returns a string that represents variable in variable notation. This string could be used as (part of) a path parameter in directory, data and action location, entry and full paths.

An Error is generated:

- When VariableName is not an existing variable.

[GetVariableValue(VariableName)]

Returns the actual value of variable with VariableName. The values of any variables, included in the variable's value definition, are expanded in the actual value. Compare [GetVariableValueDefinition], which returns the definition of the variable value.

An Error is generated:

- When VariableName is not an existing variable.

[GetVariableValueDefinition(VariableName)]

Returns the definition of variable with VariableName. If the definition includes any other variables, these variables are returned in the value string in variable notation. Compare [GetVariableValue], which returns the actual variable value.

An Error is generated:

- When VariableName is not an existing variable.

Setting path definitions

When changing flowchart parameters the commands [CreateDirAlias], [RelocateDirAliasLocalPath], [RelocateDirAliasServerPath], [SetDirAliasLocalPath], [SetDirAliasServerPath], [SetDataEntry], [SetDataFullPath], [SetDataLocation], [SetProgramEntry], [SetProgramFullPath] and

[SetProgramLocation] change the definition of a path. This path may include:

- Variables, in variable notation. These variables may occur in every one of the abovementioned commands. A variable in variable notation can be obtained using the [GetVariableNotation] command;
- Root Directories, in directory alias notation. These may occur in any of the command involving directory aliases (local and server path). A directory alias in directory alias notation can be obtained using the [GetDirAliasNotation] command;
- Environments, in environments notation. These may occur in the directory alias commands involving local paths, i.e. [CreateDirAlias], [RelocateDirAliasLocalPath] and [SetDirAliasLocalPath], An environment in environment notation can be obtained using the [GetEnvironmentNotation] command.

Errors may be generated:

- When the variable, root directory or environment does not exist in the project;
- When the path definition contains « or » markers, which do not contain correct variable, environment or root directory definitions;
- When the path definition contains a root directory that turns that definition into a "circle".

Changing flowchart parameters

The client is able to alter some parameters in the ArisFlow project. Directory aliases, entries and locations referred to by data and actions, meta-data and log-file names can be altered.

[AppendActionDescription(ActionName,Description)]

Appends the Description string to the description field of the action with ActionName. An open line is inserted into the action's description field when the Description parameter is omitted. An Error is generated:

- When no action with ActionName exists.

[AppendDataDescription(DataName,Description)]

Appends the Description string to the description field of data with DataName. An open line is inserted into the data's description field when the Description parameter is omitted. An Error is generated:

- When no data with DataName exists.

[AppendDirAliasDescription(AliasName,Description)]

Appends the Description string to the description field of [directory alias](#) with AliasName. An open line is inserted into the directory alias's description field when the Description parameter is omitted. An Error is generated:

- When no directory alias with AliasName exists.

[AppendFlowchartDescription(Description)]

Appends Description to the description field of the flowchart, which is described in the Properties section. An open line is inserted into the flowchart's description field when the Description parameter is omitted.

[AppendVariableDescription(VariableName,Description)]

Appends the Description string to the description field of [variable](#) with VariableName. An open line is inserted into the variable's description field when the Description parameter is omitted.

An Error is generated:

- When no variable with VariableName exists.

[CreateDirAlias(AliasName,LocalPathDefinition)]

Creates new directory alias with input AliasName and local path definition. When setting the path definition this may include variables, root directories and environments.

An Error is generated:

- When a directory alias with AliasName does already exist, except when the existing directory alias has the same actual path as the proposed local path;
- See "Setting Path Definitions" section.

When input local path is not present on disk the alias is still accepted.

[CreateVariable(VariableName,Value)]

Creates new variable with input VariableName and variable Value. When setting the path definition this may include other variables.

An Error is generated:

- When a variable with VariableName does already exist, except when the existing variable has the same value as the proposed value;
- See "Setting Path Definitions" section.

[RelocateDirAliasLocalPath(AliasName,LocalPathDefinition)]

Relocates the local path definition of the directory alias with AliasName to local path definition.

When setting the path definition this may include variables, root directories and environments.

The [RelocateDirAliasLocalPath] command affects all actions and data, which contain this directory alias. The status of these objects is checked and possibly updated. Actions and data, which were up-to-date, will normally remain up-to-date. Objects become undefined when files or directories referred to in these objects become non-existing. [\[SetDirAliasLocalPath\]](#) always sets data and actions, containing the directory alias, either not-up-to-date or undefined.

An Error is generated:

- When no directory alias with AliasName exists;
- See "Setting Path Definitions" section.

When input local path is not present on disk the directory alias is still accepted, making all actions and data containing the directory alias become undefined.

[RelocateDirAliasServerPath(AliasName,ServerPathDefinition)]

Relocates the server path definition of the directory alias with AliasName to the server path definition. When setting the path definition this may include variables and root directories.

The [RelocateDirAliasServerPath] command affects all actions and data, which contain this directory alias and use server directories. The status of these objects is checked and possibly updated. Actions and data, which were up-to-date, will normally remain up-to-date. Objects become undefined when files or directories referred to in these objects become non-existing. [\[SetDirAliasServerPath\]](#) always sets data and actions, containing the directory alias, either not-up-to-date or undefined.

An Error is generated:

- When no directory alias with AliasName exists;
- See "Setting Path Definitions" section.

[RelocateVariableValue(VariableName,Value)]

Relocates the value definition of the variable with VariableName to Value. When setting the value

definition this may include other variables.

The `[RelocateVariableValue]` command affects all actions and data that contain either this variable or a directory alias which uses this variable. The status of these objects is checked and possibly updated. Actions and data, which were up-to-date, will normally remain up-to-date. Objects become undefined when files or directories referred to in these objects become non-existing. By comparison `[SetVariableValue]` always makes data and actions, not-up-to-date or undefined.

An Error is generated:

- When no variable with VariableName exists;
- See "Setting Path Definitions" section.

[SetActionDescription(ActionName,Description)]

Sets the description field of the action with ActionName to the Description string. The action's description is reset when the Description parameter is omitted.

An Error is generated:

- When no action with ActionName exists.

[SetDataDescription(DataName,Description)]

Sets the description field of the data with DataName to the Description string. The data's description is reset when the Description parameter is omitted.

An Error is generated:

- When no data with DataName exists.

[SetDataDirAlias(DataName,AliasName)]

The location field of the data with DataName is set at the directory alias with AliasName. If the data is user-defined and has location stored as a string its location field is set to the full path of the directory alias with aliasname. If data was up-to-date it becomes not-up-to-date (or undefined) as the data is now related to other physical data.

The location can be assigned to the data as a string through the `[SetDataLocation]` command.

An Error is generated:

- When no data with DataName exists;
- When the data has no location field (i.e. the datatype was not yet assigned);
- When the server is the run-time ArisFlow Executable Type.

[SetDataEntry(DataName,Entry)]

The entry field in data with DataName is altered to Entry. The entry definition may contain variables. If the data was up-to-date it becomes not-up-to-date (or undefined) as the data is now related to other physical data.

An Error is generated:

- When no data with DataName exists;
- When the data has no entry field (i.e. the datatype was not yet assigned);
- When the server is the run-time ArisFlow Executable Type;
- See "Setting Path Definitions" section.

[SetDataFullPath(DataName,FullPath)]

Assigns FullPath to the location and entry fields of the data with DataName. The full path may contain variables. If the full path contains a directory alias at its first position, in directory alias notation, this is assigned as the location of the data. Compare with the `[SetDataDirAlias]` command. The `[GetDirAliasNotation]` command is useful in converting a directory alias to the directory alias notation.

If the data location is stored as a directory alias, but the full path contains no directory alias reference, the location part of the full path is converted to a matching directory alias. If no

directory alias matches an error is returned.

If the data was up-to-date it becomes not-up-to-date (or undefined) as the data is now related to other physical data.

An Error is generated:

- When no data with DataName exists;
- When the data has not yet a datatype assigned;
- When the data location is stored as a directory alias and no existing directory alias has the same actual path as the location assigned;
- When the server is the run-time ArisFlow Executable Type;
- See "Setting Path Definitions" section.

[SetDataLocation(DataName,Location)]

The location field in data with DataName is altered to Location. The location is a path string that may contain variables. Files, directories and sometimes user-defined data store the data location as a directory alias. In that case ArisFlow attempts to assign the directory alias that matches the location string's actual path, and returns an error if no matching directory alias is defined. For assigning directory aliases however the [\[SetDataDirAlias\]](#) command is more appropriate. If the data was up-to-date it becomes not-up-to-date (or undefined) as the data is now related to other physical data.

An Error is generated:

- When no data with DataName exists;
- When the data has no location field (i.e. the datatype was not yet assigned);
- When the data location is stored as a directory alias and no existing directory alias has the same actual path as the location to be assigned;
- When the server is the run-time ArisFlow Executable Type;
- See "Setting Path Definitions" section.

[SetDirAliasDescription(AliasName,Description)]

Sets the description field of the [directory alias](#) with AliasName to Description. The directory alias's description is reset when the Description parameter is omitted.

An Error is generated:

- When no directory alias with AliasName exists.

[SetDirAliasLocalPath(AliasName,LocalPathDefinition)]

Sets the local path definition of the directory alias with AliasName to the local path definition. When setting the path definition this may include, and generate errors for, variables, root directories and environments.

The [SetDirAliasLocalPath] command affects all actions and data, which contain this directory alias. The status of these objects is checked and possibly updated. Actions and data, which were up-to-date, will usually become not-up-to-date. Objects become undefined when files or directories referred to in these objects become non-existing. By comparison [\[RelocateDirAliasLocalPath\]](#) does not change the status of up-to-date data and actions, unless it becomes undefined.

An Error is generated:

- When no directory alias with AliasName exists.
- When the local path definition string contains either an unknown or incorrectly described environment, variable or root directory alias;
- When the ServerPathDefinition string contains a root directory which turns the definition into a "circle";
- See "Setting Path Definitions" section.

When input local path is not present on disk the directory alias is still accepted, making all actions and data containing the directory alias become undefined.

[SetDirAliasServerPath(AliasName,ServerPathDefinition)]

Sets the server path definition of the directory alias with AliasName to the server path definition. When setting the path definition this may include, and generate errors for, variables and root directories.

The [SetDirAliasServerPath] command affects all actions and data, which contain this directory alias and use server directories. The status of these objects either remains undefined or becomes not-up-to-date. [\[RelocateDirAliasServerPath\]](#) leaves data and actions up-to-date.

An Error is generated:

- When no directory alias with AliasName exists;
- See "Setting Path Definitions" section.

[SetFlowchartDescription(Description)]

- Sets the description field of the flowchart, described in the Properties section, to Description. The flowchart's description is reset when the Description parameter is omitted. The [GetFlowchartDescription](#) request retrieves the flowchart description.

[SetFlowchartTitle(Title)]

- Sets the title field of the flowchart, described in the Properties section, to Title. The [GetFlowchartTitle](#) request retrieves the flowchart title.

[SetProgramDirAlias(ActionName,AliasName)]

The location field of the program action with ActionName is set at the directory alias with AliasName. If the action was up-to-date it becomes not-up-to-date (or undefined), as the program is now another executable.

The location can be assigned to the action as a string through the [\[SetProgramLocation\]](#) command.

An Error is generated:

- When no action with ActionName exists;
- When the action type is not a program;
- When the server is the run-time ArisFlow Executable Type.

[SetProgramEntry(ActionName,Entry)]

The entry field in program actions with ActionName is altered to Entry. The entry definition may contain variables. If the action was up-to-date it becomes not-up-to-date (or undefined), as the program is now another executable.

An Error is generated:

- When no action with ActionName exists;
- When the action type is not a program;
- When the server is the run-time ArisFlow Executable Type;
- See "Setting Path Definitions" section.

[SetProgramFullPath(ActionName,FullPath)]

Assigns FullPath to the location and entry fields of the action with ActionName. The full path may contain variables. If the full path contains a directory alias, in directory alias notation, at its first position this is assigned as the location of the action. Compare with the [\[SetProgramDirAlias\]](#) command. The [\[GetDirAliasNotation\]](#) command is useful in converting a directory alias to the directory alias notation.

If the full path contains no directory alias reference, the location part of the full path is converted to a matching directory alias. If no directory alias matches an error is returned. If the action was up-to-date it becomes not-up-to-date (or undefined), as the program is now

another executable.

An Error is generated:

- When no action with ActionName exists;
- When the action type is not a program;
- When no existing directory alias has the same actual path as the location assigned;
- When the server is the run-time ArisFlow Executable Type;
- See "Setting Path Definitions" section.

[SetProgramLocation(ActionName,Location)]

The location field in program action with ActionName is altered to Location. The location should be a full path, which may contain variables. For assigning directory aliases however the [\[SetProgramDirAlias\]](#) command may be more appropriate. If the action was up-to-date it becomes not-up-to-date (or undefined), as the program is now another executable.

An Error is generated:

- When no action with ActionName exists;
- When the action type is not a program;
- When the server is the run-time ArisFlow Executable Type;
- See "Setting Path Definitions" section.

[SetVariableDescription(VariableName,Description)]

Sets the description field of the [variable](#) with VariableName to Description string. The variable's description is reset when the Description parameter is omitted.

An Error is generated:

- When no variable with VariableName exists.

[SetVariableValue(VariableName,Value)]

Sets the value definition of the [variable](#) with VariableName to Value. When setting the value definition this may include other variables.

The [\[SetVariableValue\]](#) command makes all actions and data, which contain this variable or a directory alias that uses this variable, **not-up-to-date**. Objects become undefined when files or directories referred to in these objects become non-existing. By comparison [\[RelocateVariableValue\]](#) leaves the status of data and actions unchanged.

An Error is generated:

- When no variable with VariableName exists;
- See "Setting Path Definitions" section.

Obsolete commands

The following DDE-server commands of ArisFlow Version 1.0.1 are now obsolete. ArisFlow will support these commands up to and including version 2.1. All of these commands have been replaced by other command names. The obsolete commands are:

- [Close], which is replaced by [\[Quit\]](#);
- [Maximise], which is replaced by [\[SetAppMaximise\]](#);
- [Minimise], which is replaced by [\[SetAppMinimise\]](#);
- [SetDirAlias], which is replaced by [\[SetDirAliasLocalPath\]](#);
- [UpdateDirAlias], which is replaced by [\[RelocateDirAliasLocalPath\]](#).

The following DDE-server commands of ArisFlow Version 2.0 are now obsolete. ArisFlow will

support these commands up to version 3. All of these commands have been replaced by other command names. The obsolete commands are:

- [GetDirAliasString], which is replaced by [\[GetDirAliasNotation\]](#);
- [RedefineDirAliasPath], which is replaced by [\[SetDirAliasLocalPath\]](#).

Utilities

Utilities are executables specifically designed to support ArisFlow. These utilities may be:

- User-defined Data Browsers, which enable the user to define a location and entry of User-defined Data;
- User-defined Data Checkers, which check the status of user-defined data;
- Data viewers which enable the user to display the contents of Files, Directories or User-defined data;
- Utilities specifically designed to perform certain tasks in programs called in actions. A utility can be used in DDE server programs to facilitate access by ArisFlow;
- Access utilities designed for accessing user-defined data or even other data. Actions use this utility by sending execution commands to it. The utility manipulates the data.

The ArisFlow program is installed together with many useful utilities. Details of the installed utilities can be found in the .txt or .hlp files installed together with the specific utility and in the ArisFlow Utilities section of this Help. It is likely that over time more and more useful utilities will be developed. The latest developments will be displayed on the ArisFlow web site ("www.arisflow.com").

ArisFlow users may also define their own utilities. The user-defined data utilities can be laid out as described in the User-defined Data Type section. The requirements of utilities supporting other programs depend very much upon the characteristics of the programs that these utilities should support.

Usage Hints

Although already described somewhere in the Help a few hints may be very useful to the user:

- Removing connections from the flowchart. Connections cannot be selected and cannot be removed directly. To remove a connection double click on the action it is connected to. In the Action dialog press the *Script* button, starting the Action Script dialog. In either the Input Data or Output Data listbox select the data that the connection is connected to. Press the remove button  and the connection is removed from the flowchart and everywhere in the action script.
- If you want to know where certain variables, directory aliases, action types or data types are used in a project: Open the dialog by selecting the relevant menu option in the [Define](#) menu. Select the object. Press the *Remove* button. This will actually remove the object if it is not used anywhere in the project, so watch out! But if the object is used anywhere a message is displayed, telling exactly where the object is used.
- Selecting all actions or data connected to a data or action. Press the Control key. With left mouse-button click on data or action. All connected actions or data become selected as well.

- Wiping a value from a combobox. In every dialog the value entered in a combobox can be wiped by clicking on the combobox, thus making the combobox the active window (the item becomes selected text), and then hitting the *delete* or *backspace* key.

ArisFlow Commander

ArisFlow Commander

The ArisFlow Commander is a very flexible utility that is used for driving ArisFlow, by sending DDE commands to ArisFlow. It can be useful in ArisFlow projects:

- Where different regions are involved;
- When different scenario's are run;
- Where various alternatives are executed;
- Where different versions of data or programs (used in actions) are involved;
- Where many similar flowcharts are created.

The ArisFlow Commander is described in the following sections:

- **Starting the ArisFlow Commander.** The ArisFlow Commander can be started from Windows or from the command prompt.
- **ArisFlow Commander Program.** This section describes how to work with the ArisFlow Commander and how the commander program interacts with ArisFlow;
- **ArisFlow Commander File.** This is the text file, which defines the commands to be sent to ArisFlow. In fact the main task for the user of the ArisFlow Commander is to create this ArisFlow Commander File.
- **ArisFlow Commander Topic Handling.** By passing the right topic to ArisFlow the Commander Program should always communicate with the right flowchart, also when when more instances of ArisFlow are running at the same time.
- **Commander Errors.** When the commander detects any errors when parsing or executing the ArisFlow Commander File these errors are written to the Commander Error file.
- **Examples of ArisFlow commander files.**

The following ArisFlow sections are useful when using the ArisFlow Commander:

- **ArisFlow as a DDE server.** This section describes how to use ArisFlow as a DDE server;
- **ArisFlow DDE-commands list.** This section has a list of all the commands that are recognised by ArisFlow.

Starting the ArisFlow Commander

The ArisFlow Commander program is located in the ArisFlow *util* directory. It can be started:

- From the windows start-up menu.
- By double clicking on an [ArisFlow commander file](#). These files have extension .acf. The commander file is automatically loaded into the [ArisFlow Commander Program](#).
- From the command prompt.

When starting the [ArisFlow Commander Program](#) from the command prompt it is possible to pass some parameters with it, but this is not required. The following parameters can be passed to the ArisFlow commander program:

- Execution options.
- The path of the [ArisFlow commander file](#), with extension .acf.
- The path of the ArisFlow commander error file, to which any errors detected in the ArisFlow commander file are written.

The first filepath is always the commander file. If a second filepath is passed this is the error

file.

The ArisFlow Commander program can be started with the following (case-insensitive) execution options:

- -C. After completing execution of the flowchart the commander program exits automatically.
- -R. The ArisFlow Commander immediately starts executing the ArisFlow Commander file, which is passed as the next parameter.

For example the command:

```
afcommander -c -r listafdirs.acf error.txt
```

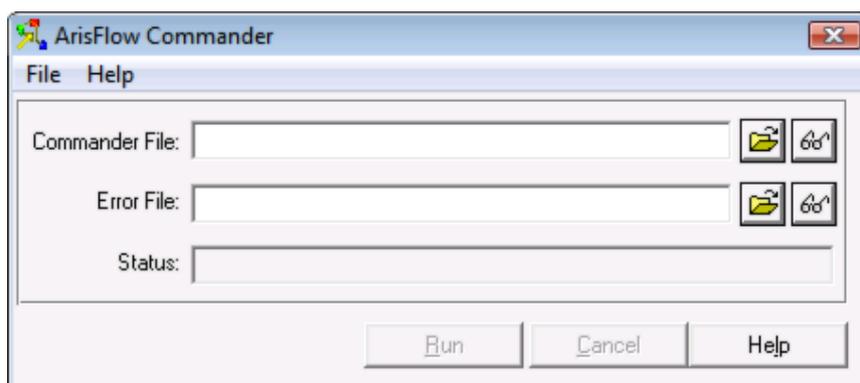
starts the afcommander with [commander file](#) *listafdirs.acf*. The [commander program](#) immediately starts executing (-r option). After completion of the execution the commander program exits (-c option). Any errors in parsing or executing the DDE commands are written to the *error.txt* file.

ArisFlow Commander Program

The ArisFlow Commander is a program, which reads a script file (the ArisFlow Commander File). This file describes DDE commands to be sent to ArisFlow. It is possible to define many different runs with different parameters in the ArisFlow Commander File.

The ArisFlow Commander is designed to communicate with ArisFlow. It can start ArisFlow, connect with ArisFlow, send commands to ArisFlow, and check when the execution of a command has completed. After completion of a command the ArisFlow Commander sends its next command to ArisFlow, and checks whether that command has been executed correctly and evaluates the result of a request to ArisFlow. In short, all these complicated tasks in running a client-server protocol are done by the ArisFlow Commander. The user only has to describe the commands to be send to ArisFlow (in a text file) and press the *Run* button.

The ArisFlow Commander has the following appearance:



Start-up screen of ArisFlow Commander program.

The Commander File path must be entered. Pressing a *Browse*  button browses for this the filename. The Error File path is not always required. The Error File is a text file, written by the commander when an error is detected in parsing the Commander file or when the execution of a DDE command by ArisFlow leads to an error. If an error occurs and the name of the Error File is not supplied the commander displays an error message.

Pressing the *View*  button enables the viewing and editing of the ArisFlow Commander file or the Error file using Notepad.

It is possible to have the Commander File field and also the Error File fields already filled in, as described in the *Starting the ArisFlow Commander* section. It is even possible to have the commander starting its execution immediately (i.e. that the *Run* button is pushed automatically) when the Commander program is started. In that case the commander may execute entirely in the background and even close automatically.

The *Status* field is a read-only field, displaying the command currently send to ArisFlow.

When the *Run* button is pushed the ArisFlow Commander starts to develop the outputs defined in the ArisFlow Commander File. If an error is detected this run is terminated. The commander run is also terminated when the *Cancel* button is pressed.

When the *Run* button is pushed the ArisFlow Commander:

- Connects to ArisFlow. If no connection is established the Commander attempts to start ArisFlow first.
- Develops the commands described in the ArisFlow Commander File.
- Sends the commands to ArisFlow. All server commands that can be send to ArisFlow are described in the ArisFlow DDE Command List section.

The ArisFlow commander ensures that commands are send to the right ArisFlow flowchart, by using the right server topic in its communication with ArisFlow. The ArisFlow Commander Topic Handling section describes this in detail.

After a command is send to ArisFlow the ArisFlow Commander performs the following steps:

- Wait for an acknowledgement by ArisFlow.
- If a command is not executed (because ArisFlow is busy) the command is send again after a short time period.
- If a command is executed the ArisFlow Commander checks whether any errors occurred during its execution. The [\[GetErrorMessage\]](#) request is send to ArisFlow. An error message is written to the error file and the generation is ceased.
- After the [\[ExecuteAll\]](#), [\[ExecuteAction\]](#), [\[ExecuteUpToAction\]](#) or [\[ExecuteUpToData\]](#) command is passed to ArisFlow the ArisFlow Commander also sends the [\[GetFlowchartExecuteStatus\]](#) or [\[GetActionExecuteStatus\]](#) request. The ArisFlow Commander continues with the next command when the result returned by ArisFlow is not *Executing* anymore.
- Send the next command to ArisFlow if no error was detected.
- Stop the DDE conversation and stop a possible execution in ArisFlow if an error is detected or if the user presses the *Cancel* button.

The ArisFlow Commander performs all these actions automatically at the right time and in the right order. It is the user's task to describe the right commands in the ArisFlow Commander File and press the *Run* button.

ArisFlow Commander File

ArisFlow Commander File

The ArisFlow Commander File is a text file, which usually has the `.acf` file extension. It defines all the commands that are to be send to ArisFlow, and that consequently drive ArisFlow. All text in the ArisFlow Commander File is capital insensitive. The ArisFlow Commander Program parses the commander file and, if the syntax is found correct, sends the DDE-Server Commands described by the ArisFlow Commander File to ArisFlow. After a command is send to ArisFlow its result is checked as described in the ArisFlow Commander Program section.

The user writes the commander file. Examples are provided in the Examples of ArisFlow commander files section. The Commander File can be edited using Notepad by pressing its  button in the ArisFlow Commander Program screen. Do not forget to save the Commander File before executing it.

The Commander File is sub-divided into sections. Each section is started by a simple line containing the type of section enclosed between [and] brackets. Each section type may occur multiple times in a commander file, although most simple Commander files may contain only a single [Variables] and a single [ServerScript] section.

The ArisFlow Commander Program recognises five section types. Any other parameter enclosed between [and] brackets will lead to an error message. Each section type is described in a separate chapter. The section types recognised are:

- [Variables]
- [ServerScript]
- [Output]
- [OutputList]
- [Commentary]

A section is started by any of the five above-mentioned section type definitions and extends until the next section's definition.

Commentary lines and blank lines are allowed in the ArisFlow Commander File and are a good way of making the commander file more understandable. The types of commentary lines are described in the Commentary section.

The basics of each section is described in a single chapter. Next to the section descriptions there are also chapters describing:

- Variable Values;
- Multiple-part Commander Files;
- Execution Commands;
- ArisFlow Commander Topic Handling;
- Errors.

The [ServerScript] and [Variables] chapters describe basic commands and variable declarations in these sections. There are also three more advanced chapters describing:

- Assignments to variables in the [ServerScript];
- Control structures in the [ServerScript];
- Array variables.

Finally there are chapters with very illustrative examples of:

- A single-part ArisFlow Commander File;

- A multiple-part ArisFlow Commander File;
- Use of control structures.

[Variables]

Variables are always declared in a [Variables] section. Multiple [Variables] sections are allowed in a [commander file](#). A variable is valid after it is declared through the remainder of the commander file. A variable that is used anywhere in the commander file should have been declared somewhere earlier in the commander file.

There are two types of variables:

- Normal variables, described in this chapter;
- [Arrays](#). These are more advanced and are described in a separate chapter.

A variable declaration appears on a single line. There are two types of declarations:

- A simple declaration, containing only the name of the variable;
- An assignment. A value is assigned to the variable as well.

A variable declaration always starts with the variable name. This variable may contain letters, numbers, blanks and most other characters except for the characters: " \ ^ & * % \$ # @ ! + - = / < > : ; , , () { }.

A simple declaration only requires the variable name. For example the declaration line:

```
DataName
```

declares the variable *DataName*. Of course the variable name is capital insensitive.

With an assignment the variable name is followed by an = sign and also contains a command line definition. This definition is the original value of the variable. This original value is valid for the variable as long as it is not overwritten in the [ServerScript] or an [Output] section. For example the declaration line:

```
DataName = Output data
```

declares the variable *DataName* and assigns its original value *Output data* to it.

Commandlines can use other variables. Where variables are used these are always enclosed in % markers, for example in the assignment declaration:

```
FileName = %FileDirectory%\%FileEntry%
```

This declares the variable *FileName*. The value of this variable is made up of three components, namely the variable *FileDirectory* (which apparently assigns its directory location), the character \ and the variable *FileEntry* (which apparently assigns *FileName*'s entry location). Every time when the value of variable *FileDirectory* or *FileEntry* is changed the value of *FileName* is updated as well.

When using a variable in the value assignment of another variable the variable used may of course not be dependent upon the value of the variable that it is defining, otherwise the ArisFlow commander will generate an error message.

A variable can be declared only once in the [commander file](#). Thus is not allowed to declare the same variable in the same or even in different [Variable] sections in the commander file.

[Variable value](#) assignments are explained further in a separate chapter.

Finally, the character # is the literal character. In any command line the character following the # is taken literally, which could be useful where for example a % character or a # character is required in a command definition. Of course # characters may not occur in variable names.

[ServerScript]

A [ServerScript] section defines the transactions and requests that are to be send to ArisFlow. Each command appears on a single line in the ArisFlow Commander File. Blank lines are allowed, but are not send to ArisFlow. All possible commands are listed in the ArisFlow DDE Command List section.

A [ServerScript] section may contain assignments to variables and control structures. These more advanced features are explained in separate chapters. Assignments and control structures do not lead to communication with ArisFlow, but they can influence which commands are send to ArisFlow in which order.

The lines in a [ServerScript] section may contain references to variables. These variable references are always enclosed between percentage markers (%). When a line containing variables is send to ArisFlow the current value of each variable is substituted in the command line that is send to ArisFlow. Also the commander puts [and] brackets around the command, so that these brackets are not required in the commander file. For example when the command line:

```
ExecuteUpToData (%DataName%)
```

is send to ArisFlow the current value for *DataName* is substituted. If that value was for example *Output data* ArisFlow will receive the DDE transaction:

```
[ExecuteUpToData(Output Data)]
```

Requests return information from ArisFlow to the client requesting the information, in this case the ArisFlow Commander Program. The result of the request must be assigned to a variable, otherwise the ArisFlow commander thinks it is dealing with a transaction. The result of this request can then be used in other server commands. For example the line:

```
DirAlias = GetDataDirAlias(Last Data)
```

sends the following request to ArisFlow:

```
[GetDataDirAlias(Last Data)]
```

ArisFlow returns the name of the directory alias that is the location of the data *Last Data* and the ArisFlow commander will then assign this value to the variable *DirAlias*. Of course *DirAlias* should have been declared earlier in a [Variables] section. Notice that the assigned variable *DirAlias*, left of the '=' sign, is not enclosed by percentage markers.

Again the # character is the literal character in command definitions, so that any character following the # (such as a % or a #) is taken literally instead of it being part of a variable or something else.

NOTE:

Assignments in a ServerScript containing special characters like () [] may result in errors

when it's intended purpose is not for a list or function, but rather a part of a text or path. This is feature by design as the ArisFlow Commander cannot distinguish intended and unintended use of special characters in a variable interpretation.

Example:

When ArisFlow is installed in C:\Program Files (x86)\ArisFlow2.8

```
ArisFlowPath=GetArisFlowPath  
  
ExamplePath=%ArisFlowPath%\Example\AFCommander
```

or

```
ArisFlowPath=C:\Program Files (x86)\ArisFlow2.8
```

will result in an error on "(".

Workaround:

```
ArisFlowPath=C:\Program Files #(x86#)\ArisFlow2.8
```

or set the conflicting assignment in the Output section.

[Output]

[Output] sections provide a mechanism to have the same sequence of commands and requests being send to ArisFlow multiple times, but with different values subsituted for the [variables](#). [Output] sections are a feature of the ArisFlow commander, which enables version management and execution of the flowchart with different parameters. An alternative to [Ouput] sections could be the use of [control structures](#) and [assignments](#) in the [\[ServerScript\]](#). Notice however that the mechanism of assigning values to variables in an [Output] section differs from assignments described in the [Assignments](#) section.

An [Output] section belongs to the last [\[ServerScript\]](#) section that was defined before the [Output] section, and is always listed after that [ServerScript] section. Each output section is started by the word [Output], followed by a number of lines that [assign values to variables](#). The variables should of course have been declared earlier in any [\[Variables\]](#) section. Each [Serverscript] section can have none, one or more than one [Output] section.

As an example a [ServerScript] could be defined as:

```
[ServerScript]  
  
%ExecuteUpTo%(%Until What%)
```

The [ServerScript] can be followed by [Output] sections:

```
[Output]  
  
ExecuteUpTo = ExecuteUpToAction  
Until What = First Action  
  
[Output]  
  
ExecuteUpTo = ExecuteUpToData
```

Until What = Output Data

Of course both the *ExecuteUpTo* and the *Until What* variables should have been declared earlier in any [\[Variables\]](#) section. The lines of this [\[ServerScript\]](#) are now executed once for each [\[Output\]](#) section. The values defined in the [\[Output\]](#) section are substituted into the variables *ExecuteUpTo* and *Until What* and then send to ArisFlow, which thus receives the DDE transactions in this order:

```
[ExecuteUpToAction(First Action)]
[ExecuteUpToData(Output Data)]
```

[\[Output\]](#) sections are not required. If a [\[ServerScript\]](#) section is not followed by any [\[Output\]](#) section the [\[ServerScript\]](#) commands are executed once, with the original values assigned to the variables that the [\[ServerScript\]](#) uses.

It is also possible to have an "empty" [\[Output\]](#) section that has no variables assigned. In that case every variable in the [\[ServerScript\]](#) starts with its original value defined when the variable was declared in a [\[Variables\]](#) section.

[OutputList]

Suppose we have data for four seasons in two different years. That is 8 sets of data. We can execute an ArisFlow flowchart for each of these 8 seasons' data and thus generate the required output data. Using the [ArisFlow Commander program](#) it is easy to drive ArisFlow for these 8 execution sections. That is one [\[ServerScript\]](#) section with 8 [\[Output\]](#) sections. But defining 8 [\[Output\]](#) sections is a bit tedious, in particular because always the same years and same seasons return in these sections. And it becomes much worse when more years are added. [\[OutputList\]](#) sections overcome this problem. In this example we can define the [commander file](#):

```
[Variables]
DataDir = c:\data\%year%\%season%

[ServerScript]
SetDirAliasLocalPath(Data Directory, %DataDir%)
ExecuteAll

[OutputList]
[Output]
Year = 2001
[Output]
Year = 2002

[OutputList]
[Output]
Season = Spring
[Output]
Season = Summer
[Output]
Season = Autumn
[Output]
Season = Winter
```

By defining [\[OutputList\]](#) sections the number of [\[Output\]](#) sections is already reduced to 6, and each [\[Output\]](#) section contains only 1 instead of 2 variables, making it much more readable and reducing the risk of making errors in any of the [\[Output\]](#) sections. And adding one year is just adding one [\[Output\]](#) section that only needs to define that specific year.

Each [\[ServerScript\]](#) can contain one or more [\[OutputList\]](#) sections. Of course having only one

section is not very useful. Each [OutputList] section is simply started by the word [OutputList]. Nothing else is required.

Each [OutputList] contains one or more [Output] sections, which define values of [variables](#) as before.

When the [\[ServerScript\]](#) is executed this is done for every [Output] section in the first [OutputList] and for every [Output] section in every other [OutputList] section. In the example above the following 16 commands are send to ArisFlow:

```
[SetDirAliasLocalPath(Data Directory, c:\data\2001\Spring)]
[ExecuteAll]
[SetDirAliasLocalPath(Data Directory, c:\data\2001\Summer)]
[ExecuteAll]
[SetDirAliasLocalPath(Data Directory, c:\data\2001\Autumn)]
[ExecuteAll]
[SetDirAliasLocalPath(Data Directory, c:\data\2001\Winter)]
[ExecuteAll]
[SetDirAliasLocalPath(Data Directory, c:\data\2002\Spring)]
[ExecuteAll]
[SetDirAliasLocalPath(Data Directory, c:\data\2002\Summer)]
[ExecuteAll]
[SetDirAliasLocalPath(Data Directory, c:\data\2002\Autumn)]
[ExecuteAll]
[SetDirAliasLocalPath(Data Directory, c:\data\2002\Winter)]
[ExecuteAll]
```

Notice that commands are send first with the first [Output] section of the first [OutputList] for every [Output] section in the second [OutputList] and then with the second [Output] section of the first [OutputList] for every [Output] section in the second [OutputList]. Also remember that [OutputList]'s are very powerful, so that using many lists with many [Output] sections could lead to a [ServerScript] being executed a lot of times.

Finally the following remarks are made:

- A [\[ServerScript\]](#) section is either followed by:
 - [OutputList] markers, which of course contain [Output] sections;
 - [Output] sections. In that case there is no [OutputList] and the [OutputList] marker cannot follow any [Output] section that belongs to that specific [ServerScript];
- It is possible to have an [\[Output\]](#) section in any [OutputList] lists with no [variable assignments](#);
- A [variable](#) assigned a value in an [Output] section in one [OutputList] cannot be assigned a value anywhere in another [OutputList] belonging to the same [ServerScript]. Otherwise an [error](#) message follows, as it is of course not clear what value that variable must have in that [ServerScript].

Commentary

Commentary is allowed in the [ArisFlow commander file](#). Commentary is ignored when parsing the file. There are three ways of writing commentary:

- The beginning of the commander file is always commentary, until the first [\[Variables\]](#) or [\[ServerScript\]](#) section is encountered;
- The [Commentary] section marker marks the beginning of a commentary section. Each line in this section is ignored until a [Variables], [ServerScript], [\[Output\]](#) or [\[OutputList\]](#) section definition is encountered;

- In any line in any other section. All text following two forward slashes (//) is treated as commentary and is ignored.

Assignment of values to Commander Variables

Variables defined in the ArisFlow commander file (which are different from variables defined in ArisFlow!) are assigned values:

- When declared in the [Variables] section. This is the original value of the variable;
- When a value or another value is assigned in an [Output] section. This value overrules the original value of the variable until the [ServerScript] has finished with that [Output] section;
- In requests and assignments defined in the [ServerScript]. This value overrules any values assigned in any [Variables] section and is valid until overruled or until another values is set in another [Output] section.

In the first two cases the variable value may depend upon the values of other variables. It is important to know that if any of these variable's values change the dependent variable's value changes as well. Assignments defined in the serverscript section may also depend upon other variables, but these values do not change when any of the determining values change.

The value of a variable assigned in an [Output] section is valid until it is overwritten in the [ServerScript] section or the [Output] section is finished. After that the variable's value is reset to its original value, defined in the [Variables] declaration, or to the value that is assigned in the next [Output] section that is executed.

A variable assigned a value in a [ServerScript] request remains valid until it is overruled. This could be very useful for commands like [\[GetDirAliasNotation\]](#), [\[GetVariableNotation\]](#) and [\[GetEnvironmentNotation\]](#). Usually these commands always yield the same value for the same flowchart. Therefore in commander files with multiple [Output] sections it is good practice to perform these requests once in a separate [ServerScript] initialisation section, that is executed only once.

Finally, when a variable is used but has no value assigned to it, this will lead to an error message during the execution of the commands.

Multi-part Files

An [ArisFlow commander file](#) may contain more than one [\[ServerScript\]](#) section. Each [ServerScript] section describes server commands being send to ArisFlow. The commands of one [ServerScript] section may have to be send to ArisFlow once. The commands of other sections may be send to ArisFlow more than once, with different [variable values](#) assigned in different [\[Output\]](#) sections. This makes multiple [ServerScript] sections very useful. An example is described in the [Example of a multi-part ArisFlow Commander File](#) section.

Execution Commands

The execution commands [\[ExecuteAction\]](#), [\[ExecuteAll\]](#), [\[ExecuteUpToAction\]](#) and [\[ExecuteUpToData\]](#) may take much longer to complete than any other commands send to ArisFlow. But the [ArisFlow Commander Program](#) should not send any other commands to ArisFlow, while it is still executing. Therefore when the ArisFlow Commander Program sends an execution command to ArisFlow this will be followed by execution status requests [\[GetFlowchartExecuteStatus\]](#) until ArisFlow returns that the execution of the action or the flowchart was completed. Only after this will the ArisFlow Commander Program send the next command to ArisFlow.

This waiting for the termination of execution is automatically performed when the commander sends an [\[ExecuteAll\]](#), [\[ExecuteAction\]](#), [\[ExecuteUpToAction\]](#) or [\[ExecuteUpToData\]](#) command to ArisFlow.

Assignments

In the [\[ServerScript\]](#) of an ArisFlow commander file the line:

```
TempEnv = GetEnvironmentNotation(TEMP)
```

sends the request [\[GetEnvironmentNotation\(TEMP\)\]](#) to ArisFlow. ArisFlow returns the result («*Env: TEMP*») and assigns that to TempEnv.

When the ArisFlow commander program encounters the line:

```
DirAlias = Output Dir
```

This is not a request for ArisFlow. Instead the value "Output Dir" is to be assigned to the commander variable *DirAlias*, and this is what happens after the commander has verified that [\[Output Dir\]](#) is not a request. This verification is done only once; the second time around the Commander Program knows it is dealing with an assignment to a variable.

Just like with requests the expression to the right of the = sign may contain variables. These variables should be enclosed between percentage markers (%).

But with assignments the expression may also contain operators and brackets, to perform arithmetic calculations. The following operators are recognised:

Operator	Name	Operand types	Priority	Unary/Binary	Result
!	not	any	6	unary	0 or 1
-	negate	number	6	unary	number
*	multiply	number	5	binary	number
/	divide	number	5	binary	number
+	add	number	4	binary	number
-	subtract	number	4	binary	number
<	is less than	number	3	binary	0 or 1
<=	is not more than	number	3	binary	0 or 1
>	is more than	number	3	binary	0 or 1
>=	is not less than	number	3	binary	0 or 1
=	is equal to	any	2	binary	0 or 1

!=	is not equal to	any	2	binary	0 or 1
&	and	any	1	binary	0 or 1
	or	any	1	binary	0 or 1

Important features of this table are:

- Operand types. Operands are always strings. However some operators require these strings to represent an integer or float number. Other operators accept any strings as operands.
- Priority. Higher priority operators are performed first. Priority can be overruled using brackets (and).
- Unary/binary. Unary operators have only one operand, which is positioned to the right of the operator. Binary operators require one operand to its left and one to its right. Some operators (e.g. the -) can be both unary and binary.
- Result. The result is always a string. With arithmetic operators these result strings always represent a number. With other operators the result is a string that is either "0" (tested condition is false) or "1" (tested condition is true).

For example the assignment:

```
A = %B%+7
```

adds 7 to the value of B and assigns that value to A. According to the table the operand *B* must be a number, otherwise an error message is generated.

The assignment:

```
A = %B%-7*%C%
```

multiplies the value of C by 7, then subtracts that from B. Because the multiplication has higher priority (see table) it is performed before the subtraction. Priority precedence can be overruled using brackets:

```
A = (%B%-7) *%C%
```

deducts 7 from the value of B and then multiplies that by the value of C.

For the *divide* operator (/) it is important what type its operands are. If both operands are integer numbers the result is an integer number:

```
A = 5/2
```

assigns the value 2 to A, as both operands (5 and 2) are integers. If one or both operands are floats the result is a float number. For example the assignment:

```
A = 5./2
```

assigns 2.5 to A, as 5. is written as a float number (recognised by the decimal point, a "."). Division by 0 always leads to an error.

Unary operators have only one operand, positioned to the right of the operator and possibly enclosed in brackets. It is possible to have multiple unary operators in sequence. These are then executed from right to left. For example:

```
A = -!%B%
```

first performs the not operator on the value of B. This may yield "0" or "1". That value is negated, so that A is assigned "0" or "-1". Alternatively:

```
A = !~%B%
```

first negates B and then performs the not operator, so that A is assigned "0" or "1".

The not (!) operator is unary, and yields "0" or "1". In the assignment:

```
A = !%B%
```

A is assigned:

- "1" when B is an empty string (no value).
- "1" when the value of B is "0".
- "0" with every other value of B.

The and (&) and or (|) operators have very low priority. That means that any other calculation or comparison is normally performed before these are performed. These operators are of course very useful in control structures. In the expression:

```
A = %B% & %C%
```

A is assigned:

- "1" when both B and C have a value that is not equal to "0".
- "0" when either B or C has a value equal to "0".
- "0" when either B or C is an empty string.

With assignment:

```
A = %B% | %C%
```

A is assigned:

- "1" when either B or C has a value that is not equal to "0".
- "0" when both B and C either have a value equal to "0" or have no value at all (empty string).

There are 6 comparison operators, yielding "0" or "1". These outcomes can be used with control structures. The operators <, <=, >= and > compare only numbers, and the outcome is "1" when the operand numbers match the expression and "0" otherwise. The operators = and != compare strings and yield "1" when both operand strings are equal, respectively not equal to each other, and "0" otherwise. The = and != comparisons are of course very tricky when comparing non-integer numbers: due to rounding off in calculations two "equal" numbers may differ very little in a remote decimal, giving rise to non-intended results.

Control Structures

In a simple [ServerScript] in an ArisFlow commander file all statements are executed sequentially, and DDE transactions and requests are sent to ArisFlow. Request results are assigned to commander variables. Assignments may perform arithmetic operations and assign results to variables. A [ServerScript] can even be executed multiple times with different variable settings using [Output] sections. This is sufficient in many situations.

The use of control structures however provide a powerful extension, by conducting the order in which the statements in the [ServerScript] are executed. There are two basic types of control structures defined in the commander:

- Conditions, where a number of lines can be executed or skipped, depending upon the

condition.

- Repetitions, where a number of lines are executed 0, 1 or more times depending upon a condition.

There are five reserved words that define control structures. These words should be the first word in a line and maybe followed by a condition. They are:

- If
- Else
- ElseIf
- While
- End

A condition always starts with the word *If*. A condition line should also contain an expression, otherwise an error is generated. This condition line is followed by statements lines and extends a line with the word *End* is encountered. For example the sequence:

```
If %J% < 0
    ExecuteAll
End
```

The *If* is followed by an expression. This expression could have values:

- "0", or empty string. In that case the condition is said to be *false* and the statements up to the *End* are not executed.
- Any other value. The condition is *true* and the statements up to the *End* are executed.

In the example the expression `%J% < 0` yields "0" or "1", as explained in the Assignments chapter, so that when the value of J is a number less than 0 the [\[ExecuteAll\]](#) statement is executed. When J is a number equal to or larger than 0 the [\[ExecuteAll\]](#) statement is skipped. When J is not a number an error is generated, as both operands of the < operator should be numbers (see chapter Assignments).

Statements following an *Else* are executed when the condition is *false*, and are skipped when a condition is *true* and the statements following the *If* were executed. The *Else* just forms a line by itself in the commander file (excluding commentary), it does not contain a condition. *Else* may only occur between lines containing the matching *If* and *End* statements. In the following sequence:

```
If %J% < 0
    ExecuteAll
Else
    J = 4
End
```

There are two possibilities:

- The condition `%J% < 0` is *true*, i.e. the value of J is less than 0. In that case the [\[ExecuteAll\]](#) transaction is sent to ArisFlow. After that the statements following the *Else* up to the *End* are skipped.
- The condition is *false*, i.e. J equals 0 or is more than 0. In that case the statements following the *If* are ignored, but the statements following the *Else* up to the *End* are executed, so that J is set to 4.

ElseIf combines the *Else* and *If* statements. It should be followed by a condition. An *ElseIf* may only occur when a previous line contained an *If* control structure that has not yet been followed by a line containing an *Else* or *End* statement. There maybe more *ElseIf* lines, as long as no matching *End* or *Else* comes between them.

That *ElseIf* condition is only tested when the matching *If* statement and all previous *ElseIf* statements yielded *false*. If the condition is *true* the statements following the *ElseIf* are

executed, until the next *Elseif*, the *Else* or the *End* is encountered. If the *Elseif* condition is *false* the next *Elseif* condition maybe tested, or lines are skipped until the matching *Else* or *End* line. For example in the following sequence:

```

If %J% < 0
  ExecuteAll
ElseIf %J% < 2
  J = 4
ElseIf %K% < 5
  J = GetDirAliasCount
Else
  ExecuteUpToAction(%ActionName%)
End

```

The statement [ExecuteAll] is executed by ArisFlow when J is negative. Otherwise when J has a value between 0 and 2 J it is set at 4 and that is it. In other cases, that is when $J \geq 2$, K is tested and when K is less than 5 the [GetDirAliasCount] request is send to ArisFlow so that the number of directory aliases is assigned to J. Finally when all conditions failed, i.e. when $J \geq 2$ and $K \geq 5$ the [ExecuteUpToAction] statement is send to ArisFlow.

A *While* enables repetition. The repetition section is started by the word *While*, followed by a condition. When the condition is *true* the repetition section is entered and the statements up to the matching *End* are executed. After that the condition is tested again and when it is still *true* the statements inside the repetition section are executed again. This goes on until the condition yields *false*. In that case the execution continues with the first line after the *End* of the *While*. For example the sequence:

```

Year = 2000
While %Year% <= 2003
  Open( %Project Directory%/project%Year% )
  ExecuteAll
  Save
  Year = %Year% + 1
End

```

This sequence starts with setting the variable *Year* at 2000. The *While* is encountered, the condition ($\%Year\% < 2003$) tested and found true, so that the repetition section is entered. The project for the Year 2000 is opened, executed and saved. In the next line ($Year = \%Year\% + 1$) the variable *Year* is increased by one to 2001. The *End* is reached and the *While* condition is tested again and of course still true. So for 2001 the project is also opened, executed and saved and *Year* is increased. The same is performed for the years 2002 and 2003, after which the parameter *Year* becomes 2004 and the *While* condition becomes *false*. The *While* is finished and the commander continues with the first line beyond the *End* statement.

Notice that *Year* is not surrounded by % markers when it is to the left of the '=' sign.

It is possible to nest control structures. For example when only projects for even years must be executed this is achieved as follows:

```

Year = 2000
While %Year% <= %Final Year%
  If %Year% / 2 * 2 = %Year%           // only even years!
    Open( %Project Directory%/project%Year% )
    ExecuteAll
    Save
  End
  Year = %Year% + 1
End

```

This example clearly demonstrates the usefulness of indenting lines when control statements occur. In this case it is easy to see which *End* belongs to which *If* or *While*, by indenting the

lines an extra number of blanks or tab between a *While* and matching *End* and between matching *If*, *Elseif*, *Else* and *End* lines.

The *If* line ensures that the lines up to the end are only performed when *Year* is even (because the integer *Year* divided by 2 is cut off when it is odd, so that after multiplication the expression left of the equal sign is one less than *Year* and the *If* thus yields *false*). The commentary on the line with the *If* is useful, because at first it maybe hard to see that the condition holds for only even years.

The *Year = %Year% + 1* line should of course be outside the *If* segment, otherwise *Year* is not increased during odd years and the *While* condition will be true forever, meaning the commander will continue until the user presses the commander program *Cancel* button.

When executing flowcharts for different months, using the names of the months, it is of course not possible to use a simple variable like *Year* for opening the right flowchart. This problem is solved by using Arrays, explained in a next chapter.

Arrays

In the [ArisFlow commander file](#) it is possible to use one-dimensional arrays as [variables](#). Multiple-dimensional arrays are not possible: the commander is designed to drive ArisFlow and not meant to perform complex arithmetics.

An array is a variable that contains more than one value. Each value is accessed through an index. In the commander file the index must always be a positive integer number (larger than or equal to 1). Each index is linked to a value, which is a string that may represent a number (like the normal variables). The number of elements is not limited, although the use of arrays with a large range of indices (difference between the lowest and highest used index) is not advisable because that could seriously hamper the performance of the commander.

An array is a variable and must be declared in a [\[Variables\]](#) section. A simple declaration of an array without any initial value in the [\[Variables\]](#) section could be:

```
Season[]
```

The array brackets [and] indicate that *Season* is an array.

It is possible to assign values to an array in the [\[Variables\]](#) section. In that case the values cannot be overwritten in any [\[ServerScript\]](#) section anymore, but different values can be assigned in [\[Output\]](#) sections. The array *Season* has initial values in the declaration:

```
Season[] = Spring, Summer, Autumn, Winter
```

This declares the array *Season* of size four and also initialises the first four elements of the array. When initial values are assigned these must be separated by commas. The values assigned to array *Season*[] are:

```
Season[1] = Spring
Season[2] = Summer
Season[3] = Autumn
Season[4] = Winter
```

Here 1, 2, 3 and 4 indicate the index, to which respective values "Spring", "Summer", "Autumn" and "Winter" in array *Season*[] are assigned.

In [\[Output\]](#) sections other values maybe assigned to variables. This is also possible for arrays. For example in many Tropical regions there are only two seasons, assigned in the [\[Output\]](#) section:

```
[Output]
Season[] = Dry, Wet
```

This [\[Output\]](#) section the array Season contains only two elements; the elements Season[3] and Season[4] do not exist anymore.

In the [\[ServerScript\]](#) values can be assigned to (an element of) an array, but only to arrays that have not had initial values assigned in [\[Variables\]](#) of [\[Output\]](#) sections, otherwise it becomes too confusing what the actual values of the array are and an [error](#) is generated. Again it is possible to assign a default value and to assign values.

An array value can be assigned in the [\[ServerScript\]](#) using an index, for example:

```
DaysInMonth[2] = 28
```

The index = 2, and thus the value 28 is assigned to the second element in DaysInMonth[2].

Both the index (for example the 2 in DaysInMonth[2]) and the value assigned (e.g. 28) can be expressions containing all operators defined in the [Assignments](#) chapter. However when a statement is executed by the commander the Index must lead to a positive integer, otherwise an [error](#) is generated. For example the statement:

```
DaysInYear[%Year%] = 365 + (%Year%/4*4 = %Year%)
```

should always have a positive integer index (the year). *DaysInYear* is the variable set in this line (being left of the = sign), and therefore not enclosed in % markers. Year is used in the index, and therefore must be enclosed in % markers, otherwise the index is taken from the literal string "Year", which of course leads to an error message ("Index Year is not a positive integer").

Array element values can be used in declarations of variable values in [\[Variables\]](#) and [\[Output\]](#) sections and in the [\[ServerScript\]](#) sections. When using the value of an element of an array the index must be provided, for example the statement:

```
DirAliasPath = GetDirAliasLocalPath(%DirAlias[%Index%]%)
```

uses array *DirAlias* of which the *Index*-th element (presumably a [directory alias](#) name) is retrieved, so that the directory alias local path can be requested from ArisFlow and assigned to the *DirAliasPath* variable.

The Commander Program also recognises some specific qualifiers with arrays. These qualifiers may only occur in the serverscript section. When qualifiers are used:

- There is no index between the array opening [and closing] brackets.
- The closing bracket is followed by a dot (.
- The qualifier dot is followed by the qualifier name. This name is capital insensitive.

The Commander Program recognises the following qualifiers:

Qualifier	Assign	Description
.clear	no	Removes all elements from the array. Returns '0'.
.firstindex	no	Returns lowest index where array element is assigned.

.initial	yes	Reference to initial value assigned to new elements in array.
.lastindex	no	Returns highest index where array element is assigned.
.nextvalue	yes	Assigns value to next element that was not yet assigned.
.size	no	Returns number of elements that are assigned a value.

Any of these qualifiers are to be used in assignments or requests right of the '=' sign, or (when assigning them is possible), to the left of the '=' sign. For example when clearing the array a variable must be used:

```
Dummy = %DaysInYear[].clear%
```

The *.nextvalue* qualifier is useful when assigning values to the array:

```
ChartStatus[].nextvalue = GetFlowchartStatus
```

assigns the flowchart status at the first index in ChartStatus that was not yet assigned a value.

A final example shows how indices can be used in a repetition [control structure](#):

```
TotDays = 0
Month = %DaysInMonth[].firstindex%
While %Month% <= %DaysInMonth[].lastindex%
    TotDays = %TotDays% + %DaysInMonth[%Month%]%
    Month = %Month% + 1
End
```

This adds the days assigned for every month into variable *TotDays*. Days are added from the first to the last element assigned in *DaysInMonth[]*.

ArisFlow Commander Topic Handling

The ArisFlow Commander ensures that server commands are sent to the right ArisFlow flowchart, by using the right server topic in its communication with ArisFlow, as described in the [How to interact with ArisFlow through DDE commands](#) section.

When the Commander sends its first command to ArisFlow its topic is *System*, unless this is an [\[Open\]](#) command. The *System* topic means that every active ArisFlow flowchart may respond to the commands sent by the Commander. Of course only one flowchart will actually respond. When ArisFlow is not executing the commander starts ArisFlow, and sends its commands to that flowchart.

There are three commands that can be sent to ArisFlow that influence the topic. These are:

- The [\[Open\]](#) command;
- The [\[SaveAs\]](#) command;
- The [\[Quit\]](#) command.

The [\[Open\]](#) command is very useful in sending the right commands to the right flowchart, and not affecting any other flowcharts. When an [\[Open\]](#) command is sent to ArisFlow, the Commander first ensures that the specified flowchart is not already opened, by trying to establish a connection with ArisFlow using the flowchart's full path as topic. If no connection is established everything is fine, and ArisFlow is started with the specified flowchart. If a connection is established, a warning message is written to the Commander Error file and the

server commands are sent to the already opened flowchart.

The [\[SaveAs\]](#) command checks whether no other flowchart with the specified full path is currently opened. If this is the case that flowchart is closed first, and a warning message is written to the Commander Error file. Any unsaved data maybe lost. After this the flowchart is saved with the specified path. The topic used by the Commander is adjusted for the following commands.

The [\[Quit\]](#) command closes the flowchart the Commander is communicating with. After the [Quit] the Commander will revert to flowchart that it was communicating with before it started exchanging commands with the flowchart that was just closed.

Any command other than [Open], [SaveAs] or [Quit] does not affect the topic used by the Commander to access ArisFlow. However, when ArisFlow is not executing a new flowchart is always started. This practice is not recommended, often it is better to ensure commands are sent to the right flowchart by passing an [\[Open\]](#) command first, thus simply opening that flowchart.

Finally the parameter of an [Open] or [SaveAs] command is a file path. This could be:

- The full path of the flowchart;
- Only the entry of the flowchart. In that case the project is opened from or saved at the directory from which the ArisFlow Commander File is executed.

Commander Commander Errors

When errors or warnings are detected by the ArisFlow Commander Program these are written to an error file. The *Error File* is an optional field of the commander program.

The *Error File* displays an error message and the line and position where the error occurred in the Commander File. This *Error File* is only created when warnings or errors occurred. There are two types of errors:

- Errors detected when the commander file is parsed. If no errors are detected at this stage the ArisFlow Commander Program starts sending commands to ArisFlow;
- Errors detected by ArisFlow during the execution of a command. This could be an error detected by the ArisFlow Commander Program with the [\[GetErrorMessage\]](#) request, but also a not assigned or recursive variable. The commander program stops execution and writes the error message to the error file.

Pressing the *View*  button in the ArisFlow Commander Program screen displays the contents of the error file using Notepad.

Examples of ArisFlow commander files

Examples of ArisFlow Commander Files

This section describes three examples of commander files:

- [Example of a simple ArisFlow Commander File](#);
- [Example of a multi-part ArisFlow Commander File](#);
- [Example of an ArisFlow Commander file with Assignments, Control Structures and Arrays](#).

Example of a Simple ArisFlow Commander File

The following is an example of a single [ServerScript] [ArisFlow Commander File](#):

```
[Commentary]
  This is an ArisFlow Commander File.
  Each output generates a different ArisFlow project file for a different
  city.

[Variables]
  Year = 2000

  CityBaseDir
  City
  CityAbbrev
  BaseDir = C:/Project/Soundwalls
  Template = %BaseDir%/Template/Template.afd
  Output = %BaseDir%/City/%CityAbbrev%%Year%.afd
  InputDataLocation = %BaseDir%/

[ServerScript]

  // Open template file.
  Open(%Template%)

  // Set new directory path for CityBase directory alias.
  CityBaseDir = GetDirAliasNotation(CityBase)
  SetDirAliasLocalPath(CityData, %CityBaseDir%/City%/Data)

  // Change entry for data CityInputData.
  SetDataEntry(CityInputData, %CityAbbrev%%Year%.dat)

  // Execute the flowchart.
  ExecuteAll

  // Save flowchart under the name of Output variable.
  SaveAs(%Output%)

[Output]
  City = Amsterdam
  CityAbbrev = ams

[Output]
  City = Rotterdam
  CityAbbrev = rot

[Output]
  City = Utrecht
```

```
CityAbbrev = utr
```

When running this commander file three project files are generated, namely one for Amsterdam, one for Rotterdam and one for Utrecht. First the Amsterdam commands are send to ArisFlow:

[\[Open\(C:/Project/Soundwalls/Template/Template.afd\)\]](#)

This command opens the template file.

[\[GetDirAliasNotation\(CityBase\)\]](#)

This is a useful request to pass to ArisFlow. Its result is «Dir: CityBase», which is used in the next command:

[\[SetDirAliasLocalPath\(CityData, «Dir: CityBase»/Amsterdam/Data \)\]](#)

The CityDir directory alias for this project is altered to the above defined directory alias definition (with CityBase as root directory).

[\[SetDataEntry\(CityInputData, ams2000.dat \)\]](#)

Changes the entry in the CityInputData data to ams2000.dat.

[\[ExecuteAll\]](#)

Performs the execution of the flowchart.

[\[GetFlowchartExecuteStatus\]](#)

This request is not described in the [ServerScript] section. However after [ExecuteAll] is acknowledged the ArisFlow Commander sends GetFlowchartExecuteStatus requests to ArisFlow until ArisFlow answers that the execution has completed. This means that the user will not have to worry about including this command.

[\[SaveAs\(C:/Project/Soundwalls/City/Amsterdam/ams2000.afd\)\]](#)

This command is send after ArisFlow returned a "not executing" result for the last of the GetFlowchartExecuteStatus requests. The SaveAs command saves the project file under input filename. The SaveAs may not even be necessary as the required output data were already generated during the execution of the flowchart!

After the Amsterdam flowchart is generated and executed the Rotterdam and Utrecht flowcharts will be generated and executed using the same command sequence with different parameters. By simply changing the Year to 2001 new project files and data can easily be generated next year.

Example of a Multiple-part ArisFlow Commander File

The following is an example of a multi-part ArisFlow Commander File. This commander file consists of three parts, where:

- A DDE debug file is initiated, which facilitates the writing of the DDE commands received by ArisFlow to this debug file.
- A flowchart is executed for different months.
- Execution of the last flowchart combines the results before ArisFlow shuts down.

In this example the second part contains multiple output sections. Thus the commander file should have multiple parts because the first and third parts require only one output section.

```
//*****
[Commentary]
    This is a multi-part ArisFlow Commander File.
```

```

//*****
// First part: Set-up debug file and open the template
//      file

[Variables]
    ProjectDir = p:\project\rainfall\

[ServerScript]
    // Open and initialise the debug file
    SetDDEDebugFilename( c:\temp\ddedebbug.txt )
    SetDDEDebugOn

    // Open template file.
    Open(%ProjectDir%template.afd)

//*****
// Second part: Execution of flowchart for different months.

[Variables]
    Month
    ProjectPath = %ProjectDir%%Month%

[ServerScript]

    // Set correct directory for rainfall data.
    SetDirAliasLocalPath(RainfallDir, %ProjectPath%)

    // Execute the flowchart.
    ExecuteAll

[Output]
    Month = January

[Output]
    Month = February

[Output]
    Month = March

//*****
// Third part: Combine the different month results in the
//      flowchart rainfall.afd, which is executed to make
//      one graph for all months.

[Variables]
    ChartStatus

[ServerScript]

    Open( %ProjectDir%rainfall.afd )
    ExecuteAll
    Save
    ChartStatus = GetFlowchartStatus
    MessageBox( ArisFlow Commander,
                Flowchart status = %ChartStatus% )

    Quit

//*****

```

The ArisFlow Commander Program sends the following commands to ArisFlow:

[\[SetDDEDebugFilename\(c:\temp\ddedebbug.txt \)\]](#)

This command makes ArisFlow open the debug file c:\temp\ddedebbug.txt. ArisFlow writes every DDE command to this file.

[\[SetDDEDebugOn\]](#)

This command is required; otherwise ArisFlow does not write anything to the DDE debug

file.

[\[Open\(P:\project\rainfall\template.afd\)\]](#)

This command opens the template file.

[\[SetDirAliasLocalPath\(RainfallDir, P:\project\rainfall\january\)\]](#)

The first month calculated is January. Apparently the data for this month are on this directory. The ProjectPath was derived from variables ProjectDir and Month.

[\[ExecuteAll\]](#)

Performs the execution of the flowchart for January.

[\[GetFlowchartExecuteStatus\]](#)

This request is not described in the [ServerScript] section. However after acknowledgement of the [ExecuteAll] command the ArisFlow Commander sends GetFlowchartExecuteStatus requests to ArisFlow, until ArisFlow replies that the execution has completed. This means that the user does not have to worry about including this command.

[\[SetDirAliasLocalPath\(RainfallDir, P:\project\rainfall\february\)\]](#)

The second month calculated is February. The variable Month is now February, so ProjectPath becomes P:\project\rainfall\february.

[\[ExecuteAll\]](#)

Performs the execution of the flowchart for February.

[\[GetFlowchartExecuteStatus\]](#)

Again the ArisFlow Commander sends GetFlowchartExecuteStatus requests to ArisFlow until ArisFlow replies that the execution has completed.

[\[SetDirAliasLocalPath\(RainfallDir, P:\project\rainfall\march\)\]](#)

The last month calculated is March.

[\[ExecuteAll\]](#)

Performs the execution of the flowchart for March.

[\[GetFlowchartExecuteStatus\]](#)

Again the ArisFlow Commander sends GetFlowchartExecuteStatus requests to ArisFlow until ArisFlow replies that the execution has completed.

[\[Open\(P:\project\rainfall\rainfall.afd\)\]](#)

This rainfall.afd project is opened, which combines the data calculated for January, February and March and calculates one graph for these months. The changes in the project P:\project\rainfall\template.afd are not saved (no [Save](#) command was given), but this was probably not required anyway.

[\[ExecuteAll\]](#)

Creates the graphs for rainfall.afd.

[\[GetFlowchartExecuteStatus\]](#)

Also here the ArisFlow Commander sends GetFlowchartExecuteStatus requests to ArisFlow until ArisFlow answers that the execution has completed.

[\[Save\]](#)

Saves the flowchart rainfall.afd after the execution was terminated.

[\[GetFlowchartStatus\]](#)

Returns the status of the flowchart. This status is assigned to the ChartStatus variable, whose value is displayed later in the messagebox. If the flowchart is up-to-date ArisFlow returns the status string "UpToDate".

[MessageBox(ArisFlow Commander, FlowchartStatus = UpToDate)]

Displays message with header "ArisFlow Commander" and message "FlowchartStatus = UpToDate". Of course the message could also display flowchartstatus NotUpToDate or Undefined if that was the outcome of the [\[GetFlowchartStatus\]](#) request cast earlier. ArisFlow will acknowledge the MessageBox command after the the OK button is clicked, so that the commander will not continue while the messagebox is displayed.

[Quit]

Quits ArisFlow, so that the flowchart is closed. Make sure it is saved first (the Save command); otherwise the changes due to the DDE session are lost in the flowchart.

Finally when starting the ArisFlow Command File from the command prompt with the -c option the ArisFlow Commander Program quits after ArisFlow has shut down. This is described in the Starting the ArisFlow Commander section.

Example of an ArisFlow Commander File with Assignments, Control Structures and Arrays

The following example illustrates the use of Assignments, Control structures and Arrays in the Commander file:

```
[Commentary]
    This commander file checks which data have certain directory aliases.
    The results are written to an output file by ArisFlow.

[Variables]
    DirAlias[] = Data, Temp, Util

    ProjectFile = P:/Project/Meteo/RainFall.afd
    OutFile = C:/Temp/OutputFile.txt

    Project          // Index counter through projects
    DataCount        // Number of data in project
    DataIndex        // Index counter through data
    DataName         // Name of the data
    DataDir          // Directory alias assigned to data
    DirIndex         // Counts through directory aliases

[ServerScript]

    // Start new OutFile by opening it and write header
    // line. Then it is automatically overwritten.
    WriteToFile(%OutFile%, Data - Directory, 1)

    // Open this ArisFlow project file.
    Open(%ProjectFile%)

    // Retrieve number of data in project.
    DataCount = GetDataCount

    // Repeatedly go through the data and check their directory
    // aliases. Start with DataIndex = 1.
    DataIndex = 1
    While %DataIndex% <= %DataCount%
        // Retrieve the DataName from DataIndex from
        // ArisFlow.
        DataName = GetDataName(%DataIndex%)

        // Retrieve the DataDir for DataName from ArisFlow.
```

```
DataDir = GetDataDirAlias(%DataName%)

// Check whether any of the DirAlias contains DataDir.
// If so: write Project, DataDir and DataName.
DirIndex = %DirAlias[].firstindex%
While %DirIndex% <= %DirAlias[].lastindex%
  // Check whether DataDir matches DirIndex-th
  // DirAlias.
  If DataDir = %DirAlias[%DirIndex%]%
    // A match. Write to file.
    AppendToFile(%OutFile%, %DataName%: %DataDir%)

    // No more directories require checking. So make
    // sure next time the while section is not entered.
    DirIndex = %DirAlias[].lastindex%
  End

  // Increase DirIndex to next value
  DirIndex = %DirIndex% + 1
End
// Increase DataIndex to next value
DataIndex = %DataIndex% + 1
End

// Finished. Quit ArisFlow.
Quit
```

This example defines three directory aliases in the *DirAlias* array: *Data*, *Temp* and *Util*. It then retrieves the number of data using the [\[GetDataCount\]](#) request. The statements in the while section are executed for the data indices 1 to this number of data.

For each data the data name is retrieved using the [\[GetDataName\(%DataIndex%\)\]](#) request. Having retrieved the data's directory alias is retrieved using the [\[GetDataDirAlias\(%DataName%\)\]](#) request. A second while section is started, moving through the directory aliases in the *DirAlias* array, and checking whether any of these directory aliases match with the data directory alias. If so the data and directory alias are written to the output file by ArisFlow through the [\[AppendToFile\]](#) transaction.

ArisFlow Utilities

ADO table browser

Program: ADObrowse

Purpose: The program displays a list of tables present in a chosen database and enables selection of a table. This program can be used to access any ADO database.

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: ADObrowse {-a} {-s} {connectionstring} [username/password|#|?] [:

Where:

-a Option for showing all table types by default

-s Option for showing schema-name by default

connectionstring ADO connection string

username/password Username/password of the database

No username/password is needed

? Display a username/password dialog

resultfile Ascii-file to which the selection result is

Example of definition in ArisFlow "User defined datatype details"

```
«program» -a -s «location» # «browser file»
```

Dialog: Connection:

The connectionstring can be build by clicking the 'Build...'-button. In ArisFlow the connectionstring will be saved in the location.

Examples of a connectionstring:

```
Provider=SQLOLEDB.1;Persist Security Info=False;User ID=myName;Initial Catalog=myDatabase;Data Source=pc3
```

To access an ESRI Geodatabase Extended Properties are needed. They can be found on tab 'All'.

```
Provider=ESRI GeoDatabase OLE DB Provider;Extended Properties="WorkSpaceFactory.1";Data Source=myPersonalGeodatabase;User ID=admin;Password=""
```

Tabletypes:

By default the tabletypes table, view en synonym are checked and the tabletypes system and temporary can be displayed. The mentioned

most common tabletypes. To display all tabletypes present in the

Result: The result will be written to the resultfile:

1st line: NOK if the user clicked the 'Cancel'-button.

of

1st line: OK if the user clicked the 'OK'-button.

2nd line: location (=database)

3rd line: entry (=table)

ADO table checker

Program: ADOcheck

Purpose: The program checks whether an ADO database table is present in the database and returns depending on the chosen option:

1. the last mutation date/time of the table;
2. the number of rows in the table;
3. the number of bytes in the table;
4. the checksum;
5. the update information of the table in a lookup table.

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: adocheck [-e|-r|-b|-s|-l] [-d] [connectionstring] [username/password]

Where:	-e	Option to check existance and last
	-r	Option to check the number of rows
	-b	Option to check number of bytes in
	-s	Option to check the checksum of all
	-l	Option to check the update informa
	-d	Option to check the date/time of t
		This option only works for MS Access

MS Excel files.

connectionstring	ADO connection string
username/password	Username/password for the database
#	Passed if no username/password req
?	Creates a dialogbox for passing the
table	The table being checked.

controlfile	File to which result is written
inifile	The name of the inifile. This argu

Example of definition in ArisFlow "User defined datatype details"

```
«program» -e «data.location» # «data.entry» «checker file»
```

Result: The controlfile contains the checking result:

```
1st line: OK      Checking was performed all right. The database
           NOK     The table does not exist, but the database does
                   thus be created.
           Other   Error during checking, the line displays the message
                   is a non-existing database.
```

The second line is only written if the first line was OK and controlfile on option:

```
2nd line: Database table last mutation data/time in the following
           year-month-day hour:minute:second
           Number of rows in the table.
           Number of bytes in the table.
           Checksum of all rows and columns in the table.
           Update information from the lookup table
```

Note 1: When using an ODBC datasource to connect to a database, the existence option (-e) is used to obtain the last mutation data/time of the table. In this case, the existence option should be used to determine whether a table is changed.

Note 2: When connecting to an MS Access table or MS Excel Worksheet through the existence option (-e) always returns the last mutation data/time of the table, respectively the xls-file.

Note 3: Checking the checksum of the table is the most reliable method to determine if the table is updated. It is also the most time-consuming one. Checking the number of rows is less reliable. If a table is updated this method will give a different result if the number of characters/bytes has changed. This method also works for tables that are deleted or appended. An update of the table is not registered. When checking large tables. Checking the number of rows only works for small tables. The last mutation date/time of the table is the least reliable method. The database does not store the mutation date of the individual tables. In this case, the existence option is used to check the database. This is no guarantee that the table being checked is up-to-date.

Note 4: When choosing the option -l the update information in a lookup table is written to the controlfile. By default the name of the lookup table is 'Lookupchk' and the table is created in the same database as the table being checked. The lookup table should have one column listing the tablename and one column listing the update information. The default name of the first column is 'TableName' and the second column is 'UpdateInfo'. If no inifile is provided, the default inifile adocheck.ini will be present in the same directory as the adocheck executable. If no inifile is provided, the defaults described above will be used.

A connectionstring, username and password should be specified in lookup table is not in the same database as the table being checked.

The format of the inifile is specified below:

```
[TABLE]
Name=lookupchk

[CONNECTION]
String=
Username=#
Password=

[COLUMNS]
TableNameCol=TableName
UpdateInfoCol=UpdateInfo
```

ADO table viewer

Program: ADOview

Purpose: The program ADOview displays the contents of any table that can be accessed through an ADO connection.

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: adoview [connectionstring] [username/password|#|?] [table]

connectionstring	ADO connection string
username/password	Username/password for the database
#	Passed if no username/password required
?	Creates a dialogbox for passing the username/password
table	The table being viewed.

Example of definition in ArisFlow "User defined datatype details"
 «program» «data.location» # «data.entry»

Example connectionstrings for some commonly used databases:

MS Access:
 "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\Data\MyData\

```
Oracle, with Microsoft OLE DB provider
    "Provider=MSDAORA;Data Source=MyDatabase;User ID=<username>;
MySQL
    "Provider=MySQLProv;Data Source=MyDatabase;User ID=<username>;
DBase (through ODBC)
    "Provider=MSDASQL;Driver={Microsoft dBASE Driver (*.dbf)};Db
```

More examples for connectionstrings can be found at:
http://www.able-consulting.com/ADO_Conn.htm

Scripting language for ArisFlow

Scripting language for ArisFlow

Program: Afscript

Version: 1.0

Purpose: Executes VBScript commands.

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: afscript [scriptfile] [controlfile]

or

afscript

Remarks: This utility executes a script with VBScript code.
The script should contain at least one subroutine,
called "Main", like

```
Sub Main()
```

```
...
```

```
End Sub
```

The script may contain additional subroutines, which
are called from the "Main" subroutine, like

```
Sub Main()
```

```
...
```

```
c = Add(a,b)
```

```
...
```

```
End Sub
```

```
Function Add(a,b)
Add = a + b
End Function
```

For additional information on how to use VBSCRIPT the following book is recommended: VBSCRIPT in a Nutshell, Matt Childs, Paul Lomax & Ron Petruscha, O'Reilly, ISDN 1-56592-720-6

To include an AFScript action type to your ArisFlow document, you need to perform the following steps:

1. Add a directory alias pointing to the directory that contains afscript.e:
2. Add an action type describing afscript

1. Add a directory alias:

- Choose 'Directory aliases' from then menu 'Define'
- Click 'Add'
- Choose a name for the alias (i.e. 'AFScript dir')
- Supply the path where afscript.exe is located. In a standard installation this is 'c:\program files\arisflow2.0\util', or '\$ARISFLOWDIR\util'

2. Add an action type

- Choose 'Action types' from the menu 'Define'
- Click 'Add'
- Choose a name for the alias (i.e. 'AFScript')
- Make sure 'Base type' is set to 'System'
- Click 'Details'
- Fill in the following fields:

Program Details:

Program: «shellscrip»

Option: Wait on terminate

Shell Script:

User: «Dir: AFScript dir»\afscript.exe «actionscrip»

Execute as: Command

Action Script:

Execute as: Script

Extension: scr

At this moment the default VBScript commands are extended with the following subroutines and functions:

```
Sys.ArisFlowCancelAction
Sys.Sleep
Sys.ClipBoardSetText
Sys.ClipBoardGetText
Sys.ClipBoardClear
Sys.ConcatPathFile
Sys.FileExists
Sys.DeleteFile
Sys.TouchFile
Sys.ReadFile
Sys.WriteFile
Sys.ExtractLine
```

```

Sys.AppDoEvents
Sys.AppActivate
Sys.AppIsRunning
Sys.AppDdeExecute
Sys.AppDdeRequest
Sys.Shell
Sys.SendKeys
Sys.WinFindWindow
Sys.WinFindAppWindow
Sys.WinFindApp
Sys.VBIncludeFile
Sys.MsgBoxChoice
Sys.MsgBoxList
Sys.MsgBoxPassword
    
```

```

*****
* Sys.ArisFlowCancelAction
*****
    
```

Purpose:

Cancel execution of the current script. The ArisFlow action will be marked as not up-to-date.

Syntax:

```

Sys.ArisFlowCancelAction
    
```

Arguments:

-

```

*****
* Sys.Sleep
*****
    
```

Purpose:

Suspends the execution for n milliseconds

Syntax:

```

Sys.Sleep(numofmsec)
    
```

Arguments:

numofmsec number of milliseconds to sleep.

```

*****
* Sys.ClipBoardSetText
*****
    
```

Purpose:

Puts a text string on the Clipboard.

Syntax:

```
    Sys.ClipBoardSetText(text)
```

Arguments:

```
    text        text to be placed on the clipboard.
```

```
*****
* Sys.ClipBoardGetText
*****
```

Purpose:

Returns a text string from the Clipboard.

Syntax:

```
    Sys.ClipBoardGetText
```

Return value:

The text available on the clipboard.

```
*****
* Sys.ClipBoardClear
*****
```

Purpose:

Clears the contents of the system Clipboard.

Syntax:

```
    Sys.ClipBoardClear
```

```
*****
* Sys.ConcatPathFile
*****
```

Purpose:

Concatenate a path and filename.

Syntax:

```
    Sys.ConcatPathFile(path, filename)
```

Arguments:

```
    path        a path.
    filename    a filename.
```

Return value:

The concatenated path and filename.

```
*****  
* Sys.FileExists  
*****
```

Purpose:

Checks if a file exists.

Syntax:

Sys.FileExists(filename)

Arguments:

filename a filename.

Return value:

True if the file exists, otherwise False.

```
*****  
* Sys.DeleteFile  
*****
```

Purpose:

Deletes a file.

Syntax:

Sys.DeleteFile(filename)

Arguments:

filename a filename.

```
*****  
* Sys.TouchFile  
*****
```

Purpose:

Updates the timestamp of a file with the current datetime.

Syntax:

Sys.TouchFile(filename)

Arguments:

filename a filename.

```
*****  
* Sys.ReadFile  
*****
```

Purpose:

Reads a text string from a file. If the file contains several lines those lines will be concatenated with carriage returns and line feeds.
Use the function ExtractLine to extract separate lines.

Syntax:

```
Sys.ReadFile(filename)
```

Arguments:

filename a filename.

Return value:

A string with read text (one or more lines).

```
*****
* Sys.ReadFile
*****
```

Purpose:

Writes a text string to a file.

Syntax:

```
Sys.WriteFile(filename,string)
```

Arguments:

filename a filename.
string the text to be written.

```
*****
* Sys.ExtractLine
*****
```

Purpose:

Extracts a single line from a string with lines separated with carriage returns and line feeds.

Syntax:

```
Sys.ExtractLine(lines,index,byref line,byref success)
```

Argumentsts:

string the string with one or more lines.
index the n-th line to return; the first line has index 0.


```
*****  
* Sys.AppDdeExecute  
*****
```

Purpose:

Executes a DDE command.

Syntax:

Sys.AppDdeExecute (server, topic, command)

Arguments:

server the name of the DDE server program
topic the topic name
command the DDE command wich will be executed

Return value:

Returns 0 if ok, or not 0 if the command fails.

```
*****  
* Sys.AppDdeRequest  
*****
```

Purpose:

Requests an item from a DDE server.

Syntax:

Sys.AppDdeRequest (server, topic, item)

Arguments:

server the name of the DDE server program
topic the topic name
item the name of an item which value is requested

Return value:

Returns the value of the item as a string, or ERROR if the request fails.

```
*****  
* Sys.Shell  
*****
```

Purpose:

Runs an executable program and returns a Variant representing the program's task ID if successful, otherwise it returns -1.

Syntax:

`Sys.Shell(command,windowstyle)`

Arguments:

`command` the command to start the program.
`windowstyle` the way the program is started.

Return value:

Returns the program's task ID if successful, otherwise -1.

Remarks:

The `windowstyle` argument has these values:

<code>vbHide</code>	Window is hidden and focus is passed to the hidden window.
<code>vbNormalFocus</code>	Window has focus and is restored to its original size and position.
<code>vbMinimizedFocus</code>	Window is displayed as an icon with focus.
<code>vbMaximizedFocus</code>	Window is maximized with focus.
<code>vbNormalNoFocus</code>	Window is restored to its most recent size and position. The currently active window remains active.
<code>vbMinimizedNoFocus</code>	Window is displayed as an icon. The currently active window remains active.

```
*****
* Sys.SendKeys
*****
```

Purpose:

Sends one or more keystrokes to the active window as if typed at the keyboard.

Syntax:

`Sys.SendKeys(string,wait)`

Arguments:

`string` string expression specifying the keystrokes to send.
`wait` Boolean value specifying the wait mode. If False, control is returned to the procedure immediately after the keys are sent. If True, keystrokes must be processed before control is returned to the procedure.

Remarks:

Each key is represented by one or more characters. To specify a single keyboard character, use the character itself. For example, to represent the letter A, use "A" for string. To represent more than one character, append each additional character to the one preceding it. To represent the letters A, B, and C, use "ABC" for string.

The plus sign (+), caret (^), percent sign (%), tilde (~), and

parentheses () have special meanings to SendKeys. To specify one of these characters, enclose it within braces ({}). For example, to specify the plus sign, use {+}. Brackets ([]) have no special meaning to SendKeys, but you must enclose them in braces. To specify brace characters, use {{} and {}}.

To specify characters that aren't displayed when you press a key, such as ENTER or TAB, and keys that represent actions rather than characters, use the codes shown below:

```
{BACKSPACE} or {BS} or {BKSP},
{BREAK} {CAPSLOCK}
{DELETE} or {DEL}
{DOWN} {END}
{ENTER} or ~
{ESC} {HELP} {HOME}
{INSERT} or {INS}
{LEFT} {NUMLOCK} {PGDN} {PGUP} {PRTSC}
{RIGHT} {SCROLLLOCK} {TAB} {UP}
{F1} ... {F16}
```

To specify keys combined with any combination of the SHIFT, CTRL, and ALT keys, precede the key code with one or more of the following codes:

Key	Code
SHIFT	+
CTRL	^
ALT	%

To specify that any combination of SHIFT, CTRL, and ALT should be held down while several other keys are pressed, enclose the code for those keys in parentheses. For example, to specify to hold down SHIFT while E and C are pressed, use "+(EC)". To specify to hold down SHIFT while E is pressed, followed by C without SHIFT, use "+EC".

To specify repeating keys, use the form {key number}. You must put a space between key and number. For example, {LEFT 42} means press the LEFT ARROW key 42 times; {h 10} means press H 10 times.

```
*****
* Sys.WinFindWindow
*****
```

Purpose:

Returns the handle of a window by means of (the beginning part) of the windowcaption string.

Syntax:

```
Sys.WinFindWindow(caption)
```

Arguments:

caption the (beginning part) of the windowcaption string.

Return value:

A handle of the window if the window is found, otherwise 0.

```
*****
* Sys.WinFindAppWindow
*****
```

Purpose:

Returns the handle of a window by means of the program's task ID.

Syntax:

```
Sys.WinFindAppWindow(taskid)
```

Arguments:

taskid id returned by the Shell or the FindApp function.

Return value:

A handle of the window if the program is found, otherwise 0.

```
*****
* Sys.WinFindApp
*****
```

Purpose:

Returns the task ID of program by means of (the beginning part) of the windowcaption string.

Syntax:

```
Sys.WinFindApp(caption)
```

Arguments:

caption the (beginning part) of the windowcaption string.

Return value:

A program's task ID if successful, otherwise -1.

```
*****
* Sys.VBIncludeFile
*****
```

Purpose:

Includes a file with additional VBScript into the current script

Syntax:

Sys.VBIncludeFile(filename)

Arguments:

filename A file with valid VBScript code.

Return value:

This function returns False if 'Filename' could be found, otherwise it returns true. This function does not check for valid VBScript code.

* Sys.MsgBoxChoice

Purpose:

This function presents the user with a modal dialog. This dialog contains a combobox with choices, a 'Cancel' and an 'OK' button.

Syntax:

Sys.MsgBoxChoice(itemlist, caption, message)

Arguments:

itemlist An array of strings. These appear in a combobox
caption An optional string which appears as the caption.
If omitted, the text "Input" will be used.
message An optional string which appears above the combobox.
If omitted, the text "Choose an item:" will be used.

Return value:

This function returns the index of the chosen item, where the first item has an index of 0, or -1 if the user pressed Cancel

* Sys.MsgBoxList

Purpose:

This function presents the user with a modal dialog. This dialog contains a listbox with choices, a 'Cancel' and an 'OK' button

Syntax:

Sys.MsgBoxList(itemlist, caption, message)

Arguments:

itemlist An array of strings. These appear in a combobox
caption An optional string which appears as the caption
If omitted, the text "Input" will be used

message An optional string which appears above the combobox
 If omitted, the text "Choose one or more items:"
 will be used

Return value:

This function returns an array of indices of the chosen items, where the first item has an index of 0, or an empty string if the user pressed 'Cancel'. Use the IsArray function to determine whether the result is a string or an array.

```
*****
* Sys.MsgBoxPassword
*****
```

Purpose:

This function presents the user with a modal dialog. This dialog contains a combobox with choices, a 'Cancel' and an 'OK' button.

Syntax:

```
Sys.MsgBoxPassword(caption, message)
```

Arguments:

caption An optional string which appears as the caption.
 If omitted, the text "Password" will be used.
 message An optional string which appears above the combobox.
 If omitted, the text "Enter a Password:" will be used.

Return value:

This function returns the given password as a string

ArisFlow-ArcView Server Extension

Program: afwserver.avx

Purpose: Execution of Avenue scripts from ArisFlow in ArcView.

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Usage: In ArisFlow define "action type" with "base type" DDE:

```
Server commands:
    Pre:          Extension.Open("«Dir: Util»\AfwServer.avx".A:
    User:          av.run("afwserver", "«actionscript»")
    Execute as    Script
```

Where:

«Dir: Util» The directory which has AfwServer.avx.
 «actionscript» Pre-defined option which resembles the action

Or:

Server commands:

Pre: Extension.Open("«Dir: Util»\AfwServer.avx".AsFileName
 User: «actionscript»
 Execute as Command

Where:

«Dir: Util» The directory which has AfwServer.avx.
 «actionscript» Pre-defined option which resembles the action
 Script: av.run("afwserver",{"filename", attributes})

Example: See ArisFlow example file: Example/ArcView/av_scrpt.afd.
 Action type: ArcView Script.

See also ArisFlow example file: Example/ArcView/av_runave.afd.
 Action type: ArcView Command.

Send ArcInfo commands to a server

Program: airun

Purpose: The program sends ArcInfo commands as an aml to an
 ArcInfo RPC server.

Copyright: ARIS, Utrecht, the Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: airun {-k} {-ux} {-r} [connectfile] [scriptfile] [controlfile]

Where:

-k Option where the aml is not removed after execution.
 -ux Option where the aml is written in unix format.
 -r Option which makes the program display error messages
 ask for a return after an error was detected.
 [connectfile] The name of the file created by Arc:[iacopen] which
 contains the RPC information.
 [scriptfile] The script file with (ArcInfo) commands.
 [controlfile] Name of the file into which control information is
 written by the airun.

Controlfile contents, first (and only) line:

NOK Error during execution
 OK No error during execution

Remark:**Extra files:**

The program uses three extra files, namely `airun.shr`, `airun.pre` and `airun.pos`. These files should reside in the same directory as `airun.aml` and can be adapted by the user / administrator.

airun.shr

This file contains information about shared directories. Its first line describes the local directory, the second line the remote directory. The remote directory should be in windows or unix format depending on the operating system on which the ArcInfo RPC-server is run.

Example 1 (windows):

```
t:\tmp
d:\tmp
```

Example 2 (unix):

```
t:\tmp
/users/tmp
```

airun.pre (optional)

This file contains ArcInfo commands which are executed prior to user commands. These commands will be inserted at the beginning of the `aml`.

airun.pos (optional)

This file contains ArcInfo commands which are executed after the user commands. These commands are appended at the end of the `aml`. These post commands may contain for example an error routine.

ArcInfo as RPC-server

Description of actions required to start up ArcInfo as an RPC-server.

1. Load portmapper services (if not yet executing):

```
- inst_pm %ARCHOME%\bin\portmap.exe [return]
```

from `%ARCHOME%\bin` directory or another directory which contains `portmap`.

2. Start portmapper services:

```
- open Control Panel | Services
- select portmap
- push "Start" button
```

3. Start ArcInfo:

```
- open DOS box.
- arc [return]
```

or start Arc through menu.

4. Start ArcInfo RPC Server:

- Arc: &ty [iacopen u:\iac.dat]

Extra:

A. Stop portmapper services:

- inst_pm remove

B. Stop ArcInfo RPC Server:

- &ty [iacclose]

ON THE WORKSTATION

1. Start portmapper (if not yet running):

- portmap [return]

2. Create script file.

3. Execute command:

- airun t:\tmp\iac.dat tmp.aml tmp.ctr

Execute an ArcMap 9.x VBA macro from the command line

Program: ArcMapExec

Version: 1.0

Purpose: Execute an ArcMap 9.x VBA macro from the command line.
VBA is deprecated in ArcGIS 10.
It is advised to use Python scripts instead of VBA.
See the ArisFlow ArcMapPython example.

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: ArcMapExec <-f filename> {-c filename} {-s} {-a} {-v} {-h|-?}'#1'
-f VBA script
-c Control file
-s Show ArcMap during execution
-a Do not close ArcMap after execution
-v Show version information
-h, -? Show usage information

Remarks: This utility executes a script with VBA code from within ArcMap. The script should contain at least one subroutine called "Main", like

```
Sub Main()  
    ...  
End Sub
```

The script can contain additional subroutines which are called from the "Main" subroutine, like

```
Sub Main()  
    ...  
    c = Add(a,b)  
    ...  
End Sub  
  
Function Add(a,b)  
    Add = a + b  
End Function
```

To include an ArcMapExec action type to your ArisFlow document, you need to perform the following steps:

1. Add a directory alias pointing to the directory that contains ArcMapExec
2. Add an action type describing ArcMapExec

1. Add a directory alias:

- Choose 'Directory aliases' from the menu 'Define'
- Click 'Add'
- Choose a name for the alias (i.e. 'ArcMapExec dir')
- Supply the path where afscript.exe is located. In a standard installation this is 'c:\program files\arisflow2.0\util', or '\$ARISFLOWDIR\util'

2. Add an action type

- Choose 'Action types' from the menu 'Define'
- Click 'Add'
- Choose a name for the alias (i.e. 'ArcMapExec')
- Make sure 'Base type' is set to 'System'
- Click 'Details'
- Fill in the following fields:
 - Program Details:
 - Program: <Dir: ArcMapExec>\arcmapexec.exe -f <shellscrip> -c <c
 - Option: Wait on terminate
 - Shell Script:
 - User: <actionscrip>
 - Execute as: Command
 - Action Script:
 - Execute as: Script
 - Extension: bas
- Optionally, you can add the -a or -s switches to 'Program', these might be debugging the VBA script.

ArcMap 9 data viewer

ArcMap 9 data viewer

Program: ArcMap9View

Purpose: View GIS-data by ArcMap 9.x (or ArcMap 8.x).

Supported geodatasets are:

ArcInfo Coverage or Featureclass

ArcInfo Grid

ArcMap layer file (lyr)

CAD datasets (dxf, dwg, dgn)

Images (jpeg, bmp, tiff, gif, MrSid, bip, bil, gis, img, ras,]

Personal Geodatabase Featureclass

Shapefile

TIN

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: ArcMapView.exe [location] [entryname] {separator}

Where:

location Full location of the dataset to view

entryname Geodataset (Shapefile, coverage, grid, image) :
The entryname can consist of coverage and subtype
personal geodatabase featuredataset and featureclass
In case of coverage the subtype can be: point,
polygon, region.{class}, route.{class} or annotation
If the subtype is omitted the program will search for
polygons, then lines and at last points.

separator Separator between featureclass and subtype in coverage
and between featuredataset and featureclass in personal geodatabase
Default is space.

Remarks: This version of ArcMapView is only to support the old ArcMap version
therefore renamed to ArcMap9View.

Use the regular ArcMapView for the current ArcMap version.

Example: Viewer in ArisFlow File Definition:

Command: <program> <data.location> <data.entry> \

Program: Util ArcMapView.exe

ArcMap data viewer

Program: ArcMapView

Purpose: View GIS-data by ArcMap 10. Supported geodatasets are:
 ArcInfo Coverage or Featureclass
 ArcInfo Grid
 ArcMap layer file (lyr)
 CAD datasets (dxf, dwg, dgn)
 File Geodatabase Featureclass
 Images (jpeg, bmp, tiff, gif, MrSid, bip, bil, gis, img, ras, j
 Personal Geodatabase Featureclass
 Shapefile
 TIN

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: ArcMapView.exe [location] [entryname] {separator}

Where:

location	Full location of the dataset to view
entryname	Geodataset (Shapefile, coverage, grid, image) : The entryname can consist of coverage and subtype file/personal geodatabase featuredataset and fe In case of coverage the subtype can be: point, polygon, region.{class}, route.{class} or anno If the subtype is omitted the program will see polygons, then lines and at last points.
separator	Separator between featureclass and subtype in and between featuredataset and featureclass in Default is space.

Remarks: -

Example: Viewer in ArisFlow File Definition:

```
Command: <program> <data.location> <data.entry> \  
Program: Util ArcMapView.exe
```

ArcView data viewer

Program: AVview

Purpose: View GIS-data by ArcView 3.x

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: AVview [ArcView bin directory] [dataset] {featureclass}

Where:

ArcView bin directory Directory containing arcview.exe

dataset Geodataset (Shapefile, coverage, image) to

featureclass In case of coverage: point, line, polygon

Remarks: This viewer uses SrcName.Make in Avenue. See the ArcView help on and featureclasses.

For a ESRI Grid it is advised to have color file in the same directory as the grid (as required by ArcView to display a grid in proper colors). AVview integrates viewers you would expect to be called GridView according to the ArisFlow utilities GridBrws, GridChk, CovBrws.

Example: Viewer in ArisFlow File Definition:

ESRI Shapefile, GRID or image:

```
Command: «program» «Dir: ArcView GIS bin» «data.fullpath»
Program: Util AVview.exe
```

ArcInfo polygon coverage:

```
Command: «program» «Dir: ArcView bin» «data.fullpath» polygon
Program: Util AVview.exe
```

ArcInfo coverage browser

Obsolete: This browser has been replaced by the generic databrowser and ch GeoDataBrowser. The ArcInfo coverage browser is only included for ArisFlow projects still using this browser.

Program: covbrws

Purpose: ArcInfo coverage browser. The program displays a directory tree of ArcInfo coverages and enables the selection of a coverage.

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: covbrws {-w(sec)} [directory] [controlfilename]

Where:

-w(sec) Number of seconds waited before a directory is

directory Browser start directory.
controfilename Ascii-file to which the browsing result is written.

Result:

The controlfile contains browsing result:

1st line: OK Browsing was performed all right.
 NOK Browsing was cancelled.
 Other Error during browsing, the line displays the message.

The 2nd and 3rd lines only have a result if first line was OK:

2nd line: Location of selected coverage.
3rd line: Name of selected coverage.

ArcInfo coverage checker

Obsolete: This checker has been replaced by the generic databrowser and checker GeoDataBrowser. The ArcInfo coverage checker is only included for ArisFlow projects still using this checker.

Program: covchk

Purpose: The program covchk checks whether an ArcInfo coverage exists and determines the last date of all files in the coverage directory (excl. the logfile).
To existence of a coverage is determined by checking the existence of the coverage directory and the bnd.adf file in this directory.

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: covchk {-w(sec)} [directory] [coverage] [controfilename]

Where:

-w(sec) Number of seconds waited before the coverage directory is read.
directory Location of the coverage.
coverage Name of the coverage being checked.
controfilename Ascii-file to which the checking result is written.

Result:

The controlfile contains checking result:

1st line: OK Checking was performed all right. The coverage directory
 NOK The coverage does not exist, but the directory
 can thus be created.
 Other Error during checking, the line displays the message.
 is a non-existing directory.

The second line is only written if the first line was OK and con:
2nd line: Coverage date in the following format:
 year-month-day hour:minute:second

Application that converts a space delimited file to a comma delimited file

Program: dat2txt

Purpose: Converts a space delimited file to a comma delimited file.

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: dat2txt [textfile] [controlfile]

Remarks: None

Errorchecker

Program: errchk

Purpose: The program errchk checks for a certain string in an ascii-file.
 string is located errchk writes, depending upon the options pass:
 starting the program, status information to a file.

Call: errchk {-i -ok} string infile outfile

Where: -i Check case-insensitive
 -ok Check for ok-string instead of error-string
 string The string checked for. When this string contains blank
 should be enclosed by either single or double quotes.
 infile The ascii-file checked.
 outfile The ascii-file which contains the search result.

Result: 1st line: NOK error-string located, or ok-string not located (o)
 OK error-string not located or ok-string located (op

FileGeodatabase browser and checker

Program: fgdbbrowser

Purpose: Browser and checker for FileGeodatabase datasets. The program displays a list of datasets.

can be selected. Supported FileGeodatabase datasets are:
 Feature Class
 Rasters
 Tables

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: For browsing geodata:

```
fgdbbrowser -b {-w[sec]} {location} [controlfilename]
```

For checking geodata:

```
fgdbbrowser -c {-a} {-w[sec]} [location] [entryname] [controlfilename]
```

Where:

-b -c	Browse or check.
-a	Also checks FileGeodatabase internal index files. By default these files are not included in the listing.
-w[sec]	Wait the given number of seconds before starting.
location	Browser start directory or the full location of the geodatabase.
entryname	The entryname of the dataset. When the entryname is featuredataset and featureclass they should be separated by a space.
controlfilename	Ascii-file to which the browsing or checking results are written.

Example: Browser in ArisFlow "User defined datatype details":

```
«program» -b «location» «browser file»
```

Checker in ArisFlow "User defined datatype details":

```
«program» -c «data.location» «data.entry» «checker file»
```

Result:

The controlfile contains browsing result:

```
1st line: OK      Browsing was performed all right.
          NOK      Browsing was cancelled.
          Other    Error during browsing, the line may have an error
```

The 2nd and 3rd lines only have a result if first line was OK.

```
2nd line: Location of selected dataset.
```

3rd line: Name of selected dataset. When a dataset in a feature class is selected a combination of featurename and featureclass is returned by a "\" is returned.

The controlfile contains checker result:

```
1st line: OK      Checking was performed all right.
          NOK     Dataset not found.
```

The 2nd line has a result if first line was OK:

```
2nd line: Date and time when dataset was last modified.
```

The 2nd line may have a result if first line was NOK:

```
2nd line: An error message.
```

Remarks:

- The fgdbbrowser requires .NET Framework 3.5 is installed on your computer.
- Changes in the metadata of a dataset are not detected by the checker.
- By clicking one of the columnheaders in the browser the list of datasets can be resorted.

Geodata browser and checker

Purpose: Browser and checker for geodatasets. The program displays a dialog box where a dataset can be selected. Supported geodatasets are:

```
ArcInfo Coverage or Featureclass
ArcInfo Grid
ArcMap layer file (lyr)
CAD datasets (dxf, dwg, dgn)
Images (jpeg, bmp, tiff, gif, MrSid, bip, bil, gis, img, ras, jpe)
Personal Geodatabase Featureclass
PC Arc/Info Coverage or Featureclass
Shapefile
TIN
MapInfo File (tab, map)
```

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: For browsing geodata:
geodatabrowser -b {-w[sec]} [location] [datatype] {separator} [entryname]

For checking geodata:
geodatabrowser -c {-w[sec]} {-d} [location] [entryname] [datatype]

Where:

```
-b|-c      Browse or check
-w[sec]    Wait the given number of seconds before starting
```

-d	Check the date/time of the database in case of Featureclass. When omitted, the date/time of the database is checked.
location	Browser start directory or the full location of the dataset.
entryname	The entryname of the the dataset. When the entryname is not a file it should be quoted.
datatype	Type of geodata to browse for or check. Valid datatypes are: all (or #), coverage, coveragefc, shapefile, coveragefc, pgdbfc, layer, image, pccoverage, pccoveragefc
separator	Separator between featureclass and subtype in the entryname and between featuredataset and featureclass in the entryname. Default is space.
controlfilename	Ascii-file to which the browsing or checking results are written.

Note: Using 'all' or '#' for checking geodatasets can take a long time only for browsing.

Example: Browser in ArisFlow "User defined datatype details":

```
«program» -b «location» # «browser file»
«program» -b «location» shapefile «browser file»
«program» -b «location» coveragefc \ «browser file»
```

Checker in ArisFlow "User defined datatype details":

```
«program» -c «data.location» «data.entry» coverage «checker file»
«program» -c «data.location» «data.entry» coveragefc «checker file»
«program» -c «data.location» "france polygon" coveragefc «checker file»
```

Result:

The controlfile contains browsing result:

```
1st line: OK      Browsing or checking was performed all right.
          NOK     Browsing or checking was cancelled.
          Other   Error during browsing, the line displays the message
```

The 2nd and 3rd lines only have a result if first line was OK:

```
2nd line: Location of selected coverage.
3rd line: Name of selected coverage.
```

Arclinfo grid browser

Obsolete: This browser has been replaced by the generic databrowser and checked

GeoDataBrowser. The ArcInfo grid browser is only included for old ArisFlow projects still using this browser.

Program: gridbrws

Purpose: The program gridbrws displays a directory tree with ArcInfo grids and enables the selection of a grid.

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: gridbrws {-w(sec)} [directory] [controlfilename]

Where:

-w(sec)	Number of seconds waited before a directory is
directory	Browser start directory.
controlfilename	Ascii-file to which the browsing result is written.

Result:

The controlfile contains browsing result:

1st line: OK	Browsing was performed all right.
NOK	Browsing was cancelled.
Other	Error during browsing, the line displays the message.

The 2nd and 3rd lines only have a result if first line was OK:

2nd line: Location of selected grid.

3rd line: Name of selected grid.

ArcInfo grid checker

Obsolete: This checker has been replaced by the generic databrowser and checker GeoDataBrowser. The ArcInfo grid checker is only included for old ArisFlow projects still using this checker.

Program: gridchk

Purpose: The program gridchk checks whether an ArcInfo grid exists and determines the last date of all files in the grid directory (excl. the logfiles). To existence of a grid is determined by checking the existence of the directory and the dblbnd.adf file in this directory.

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: gridchk {-w(sec)} [directory] [coverage] [controlfilename]

Where:

-w(sec) Number of seconds waited before the grid direc
 directory Location of the grid.
 coverage Name of the grid being checked.
 controfilename Ascii-file to which the checking result is writ

Result:

The controlfile contains checking result:

1st line: OK Checking was performed all right. The grid exist
 NOK The grid does not exist, but the directory does
 thus be created.
 Other Error during checking, the line displays the me
 is a non-existing directory.

The second line is only written if the first line was OK and con

2nd line: Grid date in the following format:
 year-month-day hour:minute:second

Arclnfo INFO table browser

Program: infobrws

Purpose: The program infobrws displays a directory tree with INFO tables
 and enables the selection of a table.

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: infobrws {-w(sec)} [startdirectory] [controfilename]

Where:

-w(sec) Number of seconds waited before a directory is
 startdirectory Browser start directory.
 controfilename Ascii-file to which the browsing result is writ

Result:

The controlfile contains browsing result:

1st line: OK Browsing was performed all right.
 NOK Browsing was cancelled.
 Other Error during browsing, the line displays the me

The 2nd and 3rd lines only have a result if first line was OK:

2nd line: Location of selected INFO table.

3rd line: Name of selected INFO table.

Arclnfo INFO table checker

Program: infochk

Purpose: The program infochk determines date, size and possibly checksum of an INFO table.

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: infochk {-s} {-c} {-w(sec)} [workspace] [INFO table] [controlfile]

Where:

-s	No date, only file size is determined.
-c	A checksum is determined as well.
-w(sec)	Number of seconds waited before the INFO directory is read.
directory	Location of the INFO table.
coverage	Name of the INFO table being checked.
controlfilename	Ascii-file to which the result is written.

Result: The controlfile contains checking result:

1st line: OK	Checking was performed all right. The INFO table exists.
NOK	The INFO table does not exist, but the directory exists, so the INFO table can thus be created.
Other	Error during checking, the line displays the message. If the message is a non-existing directory.

The second, third and fourth lines are only written if the first line contains result:

2nd line: INFO table date in the following format:
 year month day hour minute second
 or 0 if no date was determined.

3rd line: Size in bytes

4th line: 0 if no checksum was determined, otherwise the value of the checksum:
 an unsigned hexadecimal integer of 4 characters wide (e.g. 0000)

INFO table viewer

Program: infoview

the cancel button.

If not all the records are displayed in the table shown it is possible to display other records by pressing the following buttons or by following keystrokes :

Button	Keystroke	Action
First	Ctrl+Home	Move to the first series of records in the
Last	Ctrl+End	Move to the last series of records in the
Previous	Ctrl+PgUp	Move to the previous series of records in
Next	Ctrl+PgDown	Move to the next series of records in the

Example: Viewer in ArisFlow File Definition to retrieve all records and write in decimal notation and three digits behind the decimal point:

```
Command: «program» «data.fullpath» -1 F10.3
Program: Util  Infoview.exe
```

For a usage example see the ArcInfo examples, datatype InfoTable data Tmp.aat).

Conversion of Access to dBase

Program: MDB2DBF

Purpose: Converts an MS-Access table to a dBase table.

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: mdb2dbf [database] [username/password|#|?] [table] [dbf-file] [c

Where:

database	name of the ODBC database
username/password	username/password required to access the da
#	used when no username/password required
?	used when username/password should be enter
	a dialog box
table	the MS-Access table
dbf-file	the dBase table

Remarks: None

MDB table browser

Program: MdbBrowse

Purpose: Dialog for selection of a table from an MS Access database.

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: MdbBrowse [mdbfilename] [username/password|#|?] [resultfile]

Where:

MdbFilename	name of the MS Access database when empty a dialog for selecting an mdb-file
username/password	username/password required to access the database
#	used when no username/password required
?	used when username/password should be entered in a dialog box
resultfile	name of the file to which the result should be written

Results: The result is written to the resultfile:

1st line: NOK when the user pressed 'Cancel',

or

1st line: OK when the user pressed 'OK'

2nd line: location (=database)

3rd line: entry (=table)

Remarks: - This utility can be used with all MS Access database versions
- Queries are not displayed, only tables.

MDB table checker

Program: MDBcheck

Purpose: The program checks whether an MDB database table is present in the database and returns the last mutation date/time of the table.

Due to the fact that MS Access doesn't always update the date/time of an individual table, the date/time stamp of the MDB-file itself is used.

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: MDBcheck [MDBFileName] [username/password|#|?] [table] [resultfile]

Where: MDBFileName MDB database name.
 username/password Username/password for the database
 # Passed if no username/password req
 ? Creates a dialogbox for passing the
 username/password.
 table The MS-Access table checked.
 controlfile File to which result is written.

Result: The controlfile contains checking result:

1st line: OK Checking was performed all right. The database
 NOK The table does not exist, but the database doe
 thus be created.
 Other Error during checking, the line displays the m
 is a non-existing database.

The second line is only written if the first line was OK and con
 2nd line: Database table last mutation data/time in the following
 year-month-day hour:minute:second

MSAccess table viewer

Program: MdbView

Purpose: View MS-Access table

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: MdbView [MS-Access exe directory] [MDB-file] [Tablename]
 Where:

MS-Access exe directory Directory containing arcview.exe

MDB-file MS-Access database

Tablename Table to view

Remarks: This viewer opens the MDB-file using MS-Access and sends an
 Opentable command to the MS-Access process.

Example: Viewer in ArisFlow User Defined Data Details:

Command: <program> <Dir: MS Office> <data.location> <data.entri
 Program: Util MdbView.exe

Force not-up-to-date checker

Program: NOK

Purpose: Return NOK in order to set the data involved not-up-to-date on e

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: NOK [resultfile] {optional parameters}

Where:

resultfile name of the file to which the result should
optional parameters all remaining parameters are optional, e.g.
 to the data.

Results: The result is written to the resultfile:

1st line: NOK

Remarks: Use this utility with caution because this checker does not real
This checker set the data not-up-to-date independent of actual cl

Force up-to-date checker

Program: OK

Purpose: Return OK in order to set the data involved up-to-date on every

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: OK [resultfile] {optional parameters}

Where:

resultfile name of the file to which the result should
optional parameters all remaining parameters are optional, e.g.
 to the data.

Results: The result is written to the resultfile:

1st line: OK

Remarks: Use this utility with caution because this checker does not real
the data!!!
This checker sets the data up-to-date independent of actual chang

Parameterfile generator

Program: parfile

Purpose: Generates a parameter file based upon a script.

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: parfile [scriptfile] {controlfile}

Remark: Syntax layout of the scriptfile:

```
1st regel:          PARAMFILE <name of the parameterfile>
2nd regel:          COMMAND <command to be executed>
3rd en next lines: The contents of the parameter file to be generated
```

Example:

```
PARAMFILE c:\temp\tmp.par
COMMAND c:\arisflow\test\bin\ascmin c:\temp\tmp.par
c:\data\test\gridin1.asc
c:\data\test\gridin2.asc
c:\data\test\gridout.asc
```

Display a dialog window with text for a user action

Program: useract

Purpose: Displays a dialog window with text for a user action.

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: useract [textfile] [answerfile] [controlfile]

Remark: The text file contains the text displayed in the dialog window. Text which can be skipped should be preceded by /*, a ' or REM.

Close a windows application

Program: wkill

Purpose: Closes a windows application. The program can be used for example to close a program which is managed by ArisFlow through DDE commands. wkill closes all applications that have a matching title.

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: wkill [title]

Examples: wkill notepad : Kills 'Untitled - Notepad', 'Another document - 1'
wkill winfile
wkill excel

Start an application

Program: wlaunch

Purpose: Starts an application if not already running. Useful for example starting the PORTMAPPER which is managed by Arc/Info through RPC

Copyright: ARIS, Utrecht, The Netherlands.

License: Usage allowed only in combination with ArisFlow license.

Call: wlaunch [command]

Example: wlaunch c:\tools\portmap.exe

ArisFlow Examples

Calling another ArisFlow project through the ArisFlow commander

Demonstration of the usage of ArisFlow Commander from within an ArisFlow flowchart.

The ListDir flowchart example is opened. The output file in the ListDir flowchart is set to the value of the output in this flowchart. The flowchart is executed twice by calling the ArisFlow Commander. The Output generated by the ListDir flowchart can be viewed in this flowchart.

Command Lines and Script Files

This example shows how command lines and script files are created for four situations.

Situation 1. Both shell script and action script written to script files. Error handling is done in the shell script "post action" of the action type.

Situation 2. Shell script written to a script file, action script executed as command. Error handling is done in the shell script "post action" of the action type.

Situation 3. Both shell script and action script executed as command. Only error handling possible on "Status cheking on output data".

Situation 4. Action script written to a script file, shell script executed as command. Error handling is done inside the action script.

The example is described in the ArisFlow Help in the Execution Parsing chapter (Working with ArisFlow section).

The example contains MS-DOS actions.

Execution Quoting

This example shows how quotes are placed around script elements for three situations.

Situation 1. Quote Script Elements and Quote Clever options set.

Situation 2. Only Quote Scripts Elements set.

Situation 3. No Quoting set, all quotes inserted manually.

The example is described in the ArisFlow Help in the Execution Parsing chapter (Working with ArisFlow section).

The example contains MS-DOS actions.

List ArisFlow directories - AFCommander

Generates a directory listing of the directory in directory alias DirToList.

This alias is being set by the ArisFlow Commander script ListAfDirs.acf for every subdirectory in the ArisFlow program directory.

Open ListAfDirs.acf in the ArisFlow Commander to run this example.

Topic Demonstration

Demonstration of the usage of different topics in ArisFlow. ArisFlow will call ArisFlow and execute the flowchart it opened. Some topics are defined using variables and directory aliases.

The SendKeys example is executed. To execute another example the variables settings should be adjusted.

Remark: This example is an illustration of usage of variable topics. However, it is pointed out that for driving ArisFlow the usage of the ArisFlow Commander is more straightforward.

ArcMap Geoprocessing - ModelBuilder

In this example a predefined ModelBuilder model is executed, using ArcMap Geoprocessing with Python. ArcMap is not started for executing this example.

In the action "Run Model AFDissolve" the ModelBuilder model AFDissolve is run. This model is defined in the toolbox AFTools.tbx.

This example requires the following programs to be installed: - ArcGIS 9.3 or ArcGIS 10 - Python 2.4 or higher

Before running this example, set the ESRI-product to be used in the action "Run Model AFDissolve" bychanging the value of the variable ESRIProduct.

Notes:

1. The action "Run Model AFDissolve" is executed as a python-script. Python requires double ackslashes (\\) being used in path strings. In the actiontype Python, this is accomplished in the hellscript Pre action.

2. Under Windows NT it sometimes can occur that a directory is not recognized by the Geoprocessor. his problem can be solved by explicitly connecting to the directory in ArcMap.

3. When changing the version of Python being used, it could be necessary to delete the fileafcontrol.pyc in the examples directory.

ArcMap Geoprocessing - Python

In this example a dissolve is executed, using ArcMap Geoprocessing with Python. ArcMap is not being started for executing this example.

This example requires the following programs to be installed: - ArcGIS 9.3 or ArcGIS 10 - Python 2.4 or higher

Before running this example, set the ESRI-product to be used in the action "Dissolve" by changing the value of the variable ESRIProduct.

Notes:

1. The action Dissolve is executed as a python-script. Python requires double backslashes (\\) being used in path strings. In the actiontype Python, this is accomplished in the Shellscript Pre action.
2. When changing the version of Python being used, it could be necessary to delete the fileafcontrol.pyc in the examples directory.

GeodataTypes

This example shows how to use the GeodataBrowser and FGDBBrowser for different user-defined datatypes. The FGDBBrowser utility is used to browse and check only for geodatasets in File Geodatabases (FGDB). The GeodataBrowser utility is used to browse and check all other geodatasets in this example.

This example makes use of the utility ArcMapView to be able to view some of the geodatasets in ArcMap.

ArcInfo on local machine

This example generates a line coverage and a polygon coverage based on coordinates in the input text files. The lines in the line coverage and the square in the polygon coverage are intersected to get a line coverage with lines within the square. A copy of the info table is sorted on item 'ID'. Based on the line coverage, a grid is created.

This example assumes ArisFlow and ArcInfo are running on the same machine. If not, you should use the Arcinfo_remote example.

Before starting this example, follow the next steps to start ArcInfo as a server on this computer.

In the Control Panel - Services, make sure Portmap is started. If not, portmapper services must be loaded on the server. To do this, type: `inst_pm %ARCHOME%\bin\portmap.exe` in a DOS-box. Make sure you type the right path for portmap.exe. After that, start portmapper services by opening the Control Panel - Services, select portmap and press Start.

Start ArcInfo: Start ArcInfo RPC Server by typing: `arc: &type [iacopen c:\temp\iac.dat]` Make sure you type the right path for iac.dat, i.e. the path defined by the directory alias 'Temp', usually `c:\temp`.

Set the terminal type with `&terminal`.

This example uses a shared directory, this is the place where the file iac.dat is placed. This can be local or on the server. In this example, iacopen places iac.dat in the local temp-directory.

The utility airun is used for communication between ArisFlow and the ArcInfo server. Be sure to change the directories in airun.shr to reflect your drivemappings.

This example works with ArcInfo Workstation.

ArcInfo where coverages and grids are treated as directories

This example generates a line coverage and a polygon coverage based on coordinates in the input text files. The lines in the line coverage and the square in the polygon coverage are intersected to get a line coverage with lines within the square. A copy of the info table is sorted on item 'ID'. Based on the line coverage, a grid is created.

This example assumes ArisFlow and ArcInfo are running on the same machine. If not, you should use the Arcinfo_remote example.

ArcInfo is being started in a simple way by sending "`arc &run`" to the operating system. This method is likely to work in most circumstances, but lacks good error handling. If you want better error handling you should use the ArcInfo_local or ArcInfo_remote example.

In this example the datatypes Arcinfo coverage and Arcinfo grid are defined as directories. If you want more dedicated datatypes you should use the ArcInfo_local or ArcInfo_remote example.

This example works with ArcInfo Workstation.

ArcInfo on remote machine

This example generates a line coverage and a polygon coverage based on coordinates in the input text files. The lines in the line coverage and the square in the polygon coverage are intersected to get a line coverage with lines within the square. A copy of the info table is

sorted on item 'ID'. Based on the line coverage, a grid is created.

This example assumes ArisFlow and ARC/INFO are running on different machines. If not, you can also use the Arcinfo_local example.

Before starting this example, follow the next steps to start ArcInfo as a server.

In the Control Panel - Services, make sure Portmap is started. If not, portmapper services must be loaded on the server. To do this, type: `inst_pm %ARCHOME%\bin\portmap.exe` in a DOS-box. Make sure you type the right path for portmap.exe. After that, start portmapper services by opening the Control Panel - Services, select portmap and press Start.

Start ArcInfo. Start ArcInfo RPC Server by typing: `arc: &type [iacopen I:\windows\temp\iac.dat]` Make sure you type the right path for iac.dat, i.e. the path defined by the directory alias 'Temp', This can be local or on the server. In this example, iacopen places iac.dat in the local temp-directory.

If ArcInfo is not running on the same machine as ArisFlow, map a network drive (on the server) to the machine where ArisFlow runs and the example data is located. In ArisFlow, use this path as Server Path for the "data" Directory Alias and make sure the checkbox 'Use Server Directories' in the "ArcInfo" Action Type is checked.

Be sure to change the server path for various directory aliases to let this flowchart work.

The utility airun is used for communication between ArisFlow and the ArcInfo server. Be sure to change the directories in airun.shr to reflect your drivemappings.

This example works with ArcInfo Workstation for Unix or Windows. The ArisFlow process can run on any Windows version.

AFScript - MsgBox

This example shows the use of custom user input.

AFScript - ADO - MS Access

This example shows the use of ADO with VBScript. It selects countries in Europe from a DBASE file and writes the selection to an MS Access database.

To run this example you need to have ADO installed on your computer.

In case of problems edit the actions script to change the ADO provider.

AFScript - ADO - Oracle

Example using the ArisFlow utility ADObrowse for browsing for an Oracle table. ADOcheck with lookup table option is used for determining changes in the content of the Oracle tables. This examples uses SQL to query a default demo table in Oracle (EMP). A variable is used to set the maximum number of rows to view through the action script.

Before this example can be used:

1. Use an Oracle database with the EMP table or create an Oracle database "arisflow" with the default tables.
2. Change in this example the variable Database and set the definition to the database (data source) in which the EMP table exists. Database ("Data Source=") must be set to the appropriate Oracle*Net name which is known to the (Oracle) naming method in use. For example, for Local Naming, it is the alias in the tnsnames.ora file; for Oracle Names, it is the Oracle*Net Service Name.
2. Make the lookup table and triggers:

```
cd %ARISFLOWDIR%\example\oracle
sqlplus scott/tiger@arisflow
@ora_run
```

The data can be viewed using the ADO Viewer.

For other RDBMS (DB2, Ingres, Sybase, SQLServer, MySQL, ..):

1. Copy and change the ora_... scripts for creating lookup table, procedure and triggers (rules).
2. Change the variables for username and password (usr/pwd)
3. Change the connectionstring according to the requirements of ADO and RDMS in use (see http://www.able-consulting.com/ADO_Conn.htm)
4. Change the data and action according to your example data.

AFScript - ADO - SQLServer

Example using the ArisFlow utility ADObrowse for browsing for an SQLServer table. ADOcheck with "number of bytes" option is used for determining changes in the content of the SQLServer table. This examples uses SQL to query a default system table in SQLServer (SYSDATABASES). A variable is used to set the maximum number of rows to view through the action script.

Before this example can be used:

1. Use a SQLServer database ("master") with the SYSDATABASES table. A MSDE installation instead of the full SQLServer installation is also possible.
2. Change in this example the variable Server and set the definition to the server (data source) where SQLServer or MSDE is running.

The data can be viewed using the ADO Viewer.

AFScript - Sendkeys

Starts Notepad and the Windows Calculator and uses the sendkey function of AFScript to calculate the sum of all numbers between a start and end value (e.g. $1 + 2 + 3 + \dots + 100$). Start value and end value for the sum is being set by ArisFlow variables.

The result is copied and inserted in Notepad and saved to a file.

This example works for Windows English Edition. If you have the dutch edition of Windows, see the script for necessary changes.

AFScript - Idrisi

This example shows the use of Idrisi with ArisFlow through COM objects.

To run this example you need atleast Idrisi (service release v32-11) to be installed on your computer.

MapInfo command

Make a spatial overlay with 2 MapInfo files. One contains countries from the world, the other the polygon of europe. Select all countries which fall within Europe. The result is a map file too.

This example works with MapInfo Professional 4.5.

To disable the MapInfo QuickStart dialog, open MapInfo, go to Options -> Preferences, and uncheck 'Display QuickStart Dialog'

MS-DOS

Generates a directory listing of the TEMP directory.

MS-DOS program

Starts a program that can be called in MS-DOS.

Windows

Asks user for input, for example when non-automated processes are finished.

Windows program

Starts a program that can be called in Windows. The program asks user for input, for example when non-automated processes are finished.

Excel - DDE

Select countries in continent Europe and Asia from a table with all countries of the world. The first action uses only DDE commands, the second uses a macro. In this case a temporary sheet is being used to let the macro know which arguments must be used.

Excel - AFScript

Select countries from a specific continent from a table with all countries of the world.

Word - DDE

This example selects, inserts and replaces text and makes text bold. The text file is converted to a MS Word doc file.

This example works with the current versions of MS Word. To use the example with MS Word 7.0 for Windows 95 you need the English Edition.

Your default text editor is used as data viewer for text files.

Word - AFScript

This example selects, inserts and replaces text and makes text bold. The text file is converted to a Word doc file.

This example works with the current versions of MS Word. To use the example with MS Word 7.0 for Windows 95 you need the English Edition.

Your default text editor is used as data viewer for text files.

ArcMap VBA Macro

Make a spatial overlay with 2 shapefiles. One contains countries from the world, the other the polygon of europe. Select all countries which fall within Europe.

This example works with ArcGIS 9.2 or ArcGIS 9.3 (ArcMap / ArcView) and makes use of the utilities ArcMapExec to execute a VBA macro in ArcMap and ArcMap9View to be able to view the ShapeFiles in ArcMap 9.

ArcView command

Make a spatial overlay with 2 shapefiles. One contains countries from the world, the other the polygon of europe. Select all countries which fall within Europe.

In this example every single Avenue command is sent separately to ArcView.

This example works with ArcView 2.x and ArcView 3.x.

ArcView script

Make a spatial overlay with 2 shapefiles. One contains countries from the world, the other the polygon of europe. Select all countries which fall within Europe.

In this example all Avenue commands of one action are grouped into a script and send to ArcView. The ArisFlow utility AfwServer.avx is used to process the Avenue script in ArcView.

This example works with ArcView 3.x.

ArcView run Avenue

Make a spatial overlay with 2 shapefiles. One contains countries from the world, the other the polygon of Europe. Select all countries which fall within Europe.

In this example all Avenue commands of one action are grouped into an Avenue script file (.ave). The ArisFlow utility AfwServer.avx is used to process the Avenue script in ArcView.

This example works with ArcView 3.x.

ArcView Spatial Analyst

Converts a shapefile to a grid, using ArcView Spatial Analyst extension.

In this example all Avenue commands of one action are grouped into an Avenue script file (.ave). The ArisFlow utility AfwServer.avx is used to process the Avenue script in ArcView.

This example works only one time when ArcView is open. If you run the example again when ArcView is open, ArcView cannot delete the grid properly and cannot overwrite it with the new grid. To run this example again, first close ArcView and then try again.

This example works with ArcView 3.x.

Kill ArcView3

Kills all processes with "ArcView GIS" as caption in the window

License Agreement

ARIS Software License Agreement for ArisFlow

IMPORTANT: Read carefully before opening the sealed media package !!

ARIS is willing to license the enclosed software and related materials to you only on the condition that you accept all of the terms and conditions contained in this license agreement. Please read the terms and conditions carefully before opening the sealed media package. By opening the sealed media package you are indicating your acceptance of the ARIS Software License Agreement. If you do not agree to the terms and conditions as stated ARIS will not be prepared to license the software and related materials to you. In this event you should return the media package with the seal unbroken and all other materials (e.g. CD(s) and/or diskette(s) and documentation) to ARIS or its authorised reseller for a refund. No refund will be given if the media package seal has been broken or any materials are missing.

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 ArisFlow, 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 unauthorised 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 use the PRODUCT acting as a server by means of DDE, RPC, Internet or Intranet, provided the LICENSEE has been allowed to so in writing by ARIS and the appropriate license fees have been paid. In this case the LICENSEE shall provide the following copyright notice in applications using ArisFlow as a server: "This application uses ArisFlow for dataflow management tasks. ArisFlow is proprietary software of ARIS, the Netherlands" (or similarly stated in the natural language used in the application).

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 using 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 and ArisFlow are registered trademarks of ARIS, the Netherlands.

[Index](#)

Index

- A -

- access 102
- action 81, 68
- Action Script Buttons 70
- Action Script Edit Field 68
- Action script file and Shell script as command ,
- Action Types 81, 34
- Actions 65
- ADO table browser 189
- ADO table checker 191
- ADO table viewer 192
- advanced
- AFScript - ADO - MS Access 231
- AFScript - ADO - Oracle 232
- AFScript - ADO - SQLServer 232
- AFScript - Idrisi 233
- AFScript - MsgBox 231
- AFScript - Sendkeys 233
- Alignment and Spacing Functions 58
- Application that converts a space delimited file to a comma delimited file 213
- ArcInfo coverage browser 212
- ArcInfo coverage checker 213
- ArcInfo grid browser 217
- ArcInfo grid checker 218
- ArcInfo INFO table browser 219
- ArcInfo INFO table checker 219
- ArcInfo on local machine 229
- ArcInfo on remote machine 231
- ArcInfo where coverages and grids are treated as directories 230
- ArcMap 9 data viewer 209
- ArcMap data viewer 210
- ArcMap Geoprocessing - ModelBuilder 228
- ArcMap Geoprocessing - Python 229
- ArcMap VBA Macro 235
- ArcView command 235
- ArcView data viewer 211
- ArcView run Avenue 236
- ArcView script 235
- ArcView Spatial Analyst 236
- ARIS Software License Agreement for ArisFlow 237
- arisflow 164, 141
- ArisFlow as DDE Server 139
- ArisFlow Commander 163
- ArisFlow Commander File 166
- ArisFlow Commander Topic Handling 180
- ArisFlow Definitions 33
- ArisFlow Executable Types 40
- ArisFlow Execution 36
- ArisFlow Flowchart 31
- ArisFlow-ArcView Server Extension 205
- Arrays 178
- Assignment of values to Commander Variables 172
- Assignments 173

- B -

- Body Text 108
- Browsing a File 121

- C -

Calling another ArisFlow project through the ArisFlow commander 227
Changing flowchart parameters 155
check 128
checker 99
Close a windows application 226
command 141
Command Lines and Script Files 135, 227
Command Lines and Script Files 135, 227
commander 164
Commander Errors 181
Commentary 171
Connected Input Data and Connected Output Data Listboxes 69
Control Structures 175
Conversion of Access to dBase 221
Copy and Paste 60
Cyclic Integrity Checking 38

- D -

data 102, 71, 63, 91
Data Types 35, 91
dde 89, 85, 141
defined 102
Delete and Undo 60
details 85
Directory Alias 74
Directory Alias List 74
Directory Alias Repair 77
Directory Aliases 34, 74
Directory Selection 123
Display a dialog window with text for a user action 226
Drawing the Flowchart 57

- E -

edit 89, 90
element 71
Environment List 78
Environment Variables 78
Environments 78
Errorchecker 213
Example of a Multi-run ArisFlow Commander File 183
Example of a Single-part ArisFlow Commander File 182
Example of an ArisFlow Commander File with Assignments, Control Structures and Arrays 186
Examples of ArisFlow commander files 182
Excel - AFScript 234
Excel - DDE 234
Execute an ArcMap 9.x VBA macro from the command line 208
Executing or checking the status of (parts of) the flowchart 147
execution 128
Execution Commands 173
Execution Log File 130
Execution of Actions 37
Execution of the Action Script 70
Execution Parsing 135
Execution Preferences 117
Execution Progress 129
Execution Quoting 227

- F -

fields 89, 90
File and Directory Data Types 92
File Preferences 119
FileGeodatabase browser and checker 215

Find Dialogs 126
Floating License 43
Flowchart 51
Flowchart Tab 111
Force not-up-to-date checker 224
Force up-to-date checker 225

- G -

Geodata browser and checker 216
GeodataTypes 229

- H -

Handling errors occurring during the DDE conversation 144
Header/Footer 110
How to interact with ArisFlow through DDE commands 139
How to set up an ArisFlow Project 54

- I -

Import / Export of Definitions 105
Import / Export Through Definitions Files 105
Importing From Definitions Windows 106
In Case of Failure 121
INFO table viewer 221
Input=output Data 127
Introduction 29

- K -

Kill ArcView3 236

- L -

list 141
List ArisFlow directories - AFCommander 228

- M -

Managing the appearance of ArisFlow 146
Managing the ArisFlow project 145
MapInfo command 233
Margins 109
MDB table browser 222
MDB table checker 223
Menu 45
Mouse Cursor 51
MS-DOS 233
MS-DOS program 234
MSAccess table viewer 223
Multi-part Files 172

- N -

No quoting set, all quotes inserted manually

- O -

Obsolete commands 160
Opening an existing ArisFlow Project 55

- P -

Page Grid Lines 111
Page Set-up 108
Parameterfile generator 225
Pre-defined Option Selection 122
preferences 117, 120
Print 113
Print Preview 114
Print To File 115

program 102, 164, 99, 101
Project Management 56
properties 56

- Q -

Quote Script Elements and Quote Clever
Quote Script Elements Only
Quick Start of an ArisFlow Project 54
Quote Script Elements and Quote Clever 124
Quoting 138

- R -

registration 42
Requesting the results of the last or the current execution run 148
Retrieving information about the flowchart and its components 149

- S -

script 71, 68
Scripting language for ArisFlow 204
Selection of Flowchart Elements 59
Send ArcInfo commands to a server 207
Setting path definitions 154
Shell Script and Action Script as Script Files
Shell script file and Action script as command
Single Use License 42
specifications 99, 101
Specifications of the Browser Program 98
Start an application 226
Starting ArisFlow 40
Starting the ArisFlow Commander 163
State of Actions and Data 36
status 128
system 89, 90
System Details 82

- T -

The ArisFlow Concept 31
Toolbar 49
Tooltips, Context Menus and Special Key Options in Dialogs 125
Topic Demonstration 228
types 81, 71, 91

- U -

Usage hints 161
user 102
User-defined Data 97
User-defined Data Types 95
Utilities 161

- V -

Variable Selection 122
Variables 72, 33
Variables List 72
Version 2.0.5 27
Version 2.0.6 27
Version 2.0.7 26
Version 2.0.8 24
Version 2.1 21
Version 2.2 18
Version 2.2.1 18
Version 2.3 16
Version 2.4 14
Version 2.4.1 14
Version 2.5.1 12

Version 2.5.2 11
Version 2.5.2.1 11
Version 2.6.1 10
Version 2.7.1 9
Version 2.8.0 9
Version 2.8.1 8
viewer 101

- W -

What is New in this Version 8
Windows 234
Windows program 234
Word - AFScript 235
Word - DDE 234
Working with ArisFlow

- Z -

zoom 63

- [-

[AppendActionDescription(ActionName,Description)] 155
[AppendDataDescription(DataName,Description)] 155
[AppendDirAliasDescription(AliasName,Description)] 155
[AppendFlowchartDescription(Description)] 155
[AppendToFile(FilePath,Text)] 145
[AppendVariableDescription(VariableN 155
[CancelExecute] 147
[CheckStatus] 147
[ExecuteAction(ActionName)] 147
[ExecuteAll] 147
[ExecuteUntilAction(ActionName)] 147
[ExecuteUntilData(DataName)] 147
[GetActionCount] 149
[GetActionDescription(ActionName)] 149
[GetActionExecuteEndTime(ActionName)] 148
[GetActionExecuteStartTime(ActionName)] 148
[GetActionExecuteStatus(ActionName)] 148
[GetActionName(Index)] 149
[GetActionStatus(ActionName)] 149
[GetActualPath(PathDefinition)] 149
[GetArisFlowPath] 149
[GetDataCount] 149
[GetDataDescription(D 149
[GetErrorMessage] 144
[GetFlowchartExe 148
[GetFlowChartExecuteEndTime] 148
[GetFlowchartExecuteStartTime] 148
[MessageBox(Title,Text)] 145
[Open(ProjectName)] 145
[OutputList] 170
[Output] 169
[Quit] 145
[SaveAs(ProjectName)] 145
[Save] 145
[ServerScript] 168
[SetAppMaximise] 146
[SetAppMinimise] 146
[SetAppRestore] 146
[SetAppShowOff] 146
[SetAppShowOn] 146
[SetAppTopOff] 146
[SetAppTopOn] 146
[SetDDEDebugFilename(FileName)] 144
[SetDDEDebugOff] 144

[SetDDEDebugOn] 144
[SetWindowPosition(X, Y)] 146
[SetWindowSize(Width, Height)] 146
[ShowErrorMessagesOff] 144
[ShowErrorMessagesOn] 144
[ShowProgressInfoOff] 147
[ShowProgressInfoOn] 147
[Variables] 167
[WriteToFile(FilePath,Text)] 145
[ZoomAll] 146

