

Astrometric Observatory Control Script Library

Copyright 2008-2010 Astrometric Instruments, Inc.

Last revision date: 28-Apr-2010

This document can be ordered as Astrometric Instruments' part number DOC-20

Table of Contents

1. Introduction	3
2. Scripting Interfaces.....	3
3. Running Scripts	3
3.1. Scripting hosts	3
3.2. Types of scripts.....	4
3.3. Astrometric's software approach.....	4
4. Companion Documentation.....	4
5. Error handling.....	5
5.1. Messages introduced	5
5.2. Message Levels	5
5.3. Error handling philosophy.....	6
5.4. Error handling details	6
6. Files	7

Global replace script library with Script Library

1. Introduction

This file documents the over-all function of Astrometric's observatory control script library. This script library contains Microsoft Windows scripts designed to run under Windows Scripting Host and provide a basic set of setup and operational functionality for Astrometric's observatory control software and hardware.

The latest version of Astrometric's observatory control components for which this library has been tested is summarized in this table:

<u>Component</u>	<u>Firmware version</u>	<u>Software version</u>
Telescope	S-Box 0.29.052	Maestro 0.29.052
Dome	0.20.000	DomePro.exe 0.20.000
Focus	0.30.000	FocusPro.exe 0.30.000
Filterwheel	0.30.000	FilterPro.exe 0.30.000

2. Scripting Interfaces

Astrometric software products expose COM objects which provide interface to control system functions, via script files (e.g. WSF, VBS or JScript) run under Microsoft's Windows Script Host. These COM objects are fully ASCOM compliant. The following software and associated objects are presently provided:

- ◆ Maestro: Maestro.Telescope
- ◆ DomePro: DomePro.Dome
- ◆ FocusPro: FocusPro.Focuser
- ◆ FilterPro: FilterPro.FilterWheel
- ◆ SwitchPro: SwitchPro.Switch

The Script Library is distributed as a self-extracting .exe file that includes a hierarchy of folders expanded out under a top-level ScriptLib\ folder. A complete file list is provided below.

3. Running Scripts

3.1. Scripting hosts

Assuming Astrometric's software has been properly installed then the example scripts provided in the script library can be run through one of two methods:

1. Open a Command Prompt window, change directory (cd) to the directory containing the script you want to run and type "cscript script-name", where "script-name" is the name of the script file you want to run (e.g. "StartUp.wsf"). This method will run the script using Window's CScript.exe scripting host.
2. Simply double-click on the script file from My Computer or Windows Explorer. This method will run the script with a dialog box interface using Window's WScript.exe scripting host.

Many scripts require that they be run from the command prompt (using CScript) and will present an error if run via method #2. These scripts require the running dialog and status-update that a Command Prompt window provides.

Special note for running scripts on 64-bit versions of Windows

The cscript.exe and wscript.exe scripting hosts must be run from the \Windows\SysWOW64\ folder as follows:

- ◆ For cscript: “\Windows\SysWOW64\cscript script-name”
- ◆ For wscript: You cannot double-click on the script files from My Computer or Windows Explore unless VBS files are associated with \Windows\SysWOW64\wscript.exe. However, you cannot change script file associates in Default Programs (in Control Panel) to \Windows\SysWOW64\wscript.exe! This insidiousness can be circumvented if you first copy \Windows\SysWOW64\wscript.exe to \Windows\SysWOW64\wscriptfoo.exe (or some other name of your choice) and associate VBS files with that.

3.2. Types of scripts

The scripts in this library are a mix of Visual Basic Scripts (VBS... files with .vbs extensions) and Windows Script Files (WSF... files with .wsf extensions). WSF format allows the inclusion of other script files within a top-level file. In general, basic low-level functions are authored in VBS files and higher level tasks are in WSF files. For example, the test for being in a Hard Limit condition, and code to exit Hard Limit, is contained in the HardLimitBackOut() subroutine in the Telescope\Common_HardLimitBackOut.vbs file however code that calls HardLimitBackOut() is contained in several WSF files in this library. “Common” files are indicated with a preceding underscore character in their filename.

3.3. Configuration

Not all observatories will have all of Astrometric’s hardware installed. For example, the TFT at Palomar does not use Astrometric’s DomePro for dome control. To provide for absent hardware, the file ScriptLib\Common\Constants.vbs contains global flags that specify the presence of each hardware component. Set these flags to configure the Script Library for a particular observatory’s hardware configuration.

3.4. Astrometric’s software approach

There are a few things to note about the approach used with Astrometric’s ASCOM-compliant software (e.g. Maestro, DomePro, FocusPro, FilterPro and SwichPro). First these software are stand-alone applications that allow the user to directly control the hardware. They are not simply drivers that have no user interface. Rather, they are complete programs including a user interface and include an ASCOM-compliant “server” interface as an aside. Therefore, they operate a bit differently than classic ASCOM-compliant drivers in the following regards:

1. The software will continually attempt to connect to hardware. Therefore, there is no need for the ASCOM client/script to explicitly write the Connected or Link property. In fact, writing these properties does nothing.
2. A user interface which allows direct control of the hardware is always present on the desktop. Of course this user interface can be minimized if it is not needed.

4. Companion Documentation

The following documents, available in the MiscDocs\ folder, document the operation of Astrometric components for scripting within the ASCOM environment.

- ◆ script56.chm: Windows help file for Microsoft Windows Script Technologies.
- ◆ MaestroASCOM.pdf: describes Maestro's support for the ASCOM Telescope interface standard
- ◆ DomeProASCOM.pdf: describes DomePro's support for the ASCOM Dome interface standard <-- Not presently available
- ◆ FocusProASCOM.pdf: describes FocusPro's support for the ASCOM Focuser interface standard

- ◆ FilterProASCOM.pdf: describes FilterPro's support for the ASCOM FilterWheel interface standard <-- Not presently available
- ◆ SwitchProASCOM.pdf: describes SwitchPro's support for the ASCOM Switch interface standard <-- Not presently available
- ◆ SW_Client.pdf and SW_Client.txt: detail SkyWalker's Astrometric Telescope Control Language (ATCL) commands. ATCL commands are often useful in accessing functions not provided by ASCOM.

5. Error handling

5.1. Messages introduced

Maestro's Telescope ASCOM object provides a means to “raise” messages, generated by SkyWalker, to the ASCOM client as trappable errors. SkyWalker messages are categorized into four groups:

1. Status (e.g. “Passed through Celestial Pole”)
2. Warnings (e.g. “Approaching Meridian Limit”)
3. Alerts (e.g. “Dec/Alt HardLimit condition entered”)
4. Fatal errors (e.g. raised from internal code error checking)

Note: DomePro, FocusPro, FilterPro and SwitchPro do not, in their present versions, raise errors to the ASCOM client.

5.2. Message Levels

To enable Maestro to handle messages as trappable errors the message “level” that Maestro should raise needs to be specified. Use the CommandBool() method to specify to Maestro what message level to raise. The command string to use with CommandBool(), associated message levels, and the definition of the message levels are contained in the table below.

<u>Command string</u>	<u>Message Level</u>	<u>Definition of Message Level</u>	<u>CommandBool() return value</u>
“L1”	Level 1	Causes Maestro to raise ALL messages as trappable errors.	True
“L2”	Level 2	Causes Maestro to raise only warnings, alerts and fatal errors.	True
“L3”	Level 3	Causes Maestro to raise only alerts and fatal errors.	True
“L4”	Level 4	Causes Maestro to raise only fatal errors.	True

At start-up, Maestro’s message level is set to L4 and only fatal errors are reported to the ASCOM client. Any control script using Maestro/SkyWalker should carefully consider what message level to use.

The trappable error code that Maestro raises for messages is 0x80040411. The message associated with this error is simply the text of the message directly from SkyWalker. Maestro raises SkyWalker’s messages by “throwing” a (trappable) run-time error back to the client application (with an error code of 0x80040411).

If no messages want to be received from SkyWalker then keep the message level at its default value of L4 and only fatal errors will terminate the script. This is a reasonable approach if the scripts have been

extensively tested and no messages are expected. It is also a reasonable approach if Maestro is running on the user's screen interactively because then any messages will be visible on Maestro's screen.

If all messages that could indicate problems want to be received then set the message level to L2. It is not necessary to set the message level to L1 since simple status message can remain hidden for most applications. When the message level is set to anything other than L4 then the script should be written to properly handle errors. For setup scripts this might mean that there is no error handling in the script since the user might simply want it to terminate on error to assure that illegal setup parameters do not go undetected. The Setup scripts in this library use this approach.

5.3. Error handling philosophy

The philosophy used in authoring the Script Library is that Warnings, Alerts and Fatal Errors (collectively called "exceptions") from the hardware are, in general, not expected. If an exception is received then there is a problem that needs to be dealt with. As a general rule, to assure that exceptions **are** raised to the script the message level should be set to L2. Additionally, operational scripts (as opposed to setup or status scripts for example) should be written so that they properly handle any "expected" exceptions so that they do not lead to script termination. The premise is that all Warnings, Alerts and Fatal Errors represent exceptions to normal operation and hence should terminate script execution **but** where expectations are expected they are checked for. This recommended approach to scripting assures that expectations will terminate automated operation unless they are trapped, interpreted and properly dealt with.

5.4. Error handling details

The scripts in this library use the following error processing and messaging policy:

- ◆ The message level is set to L2 so that, by default, exceptions will terminate scripts. The default message level 4 (or message level 3) are not used other than in simple example scripts... never in an operational script. The philosophy is that exceptions should be heeded. They should never be fully suppressed. Fully suppressing messages opens the system up to potentially damaging behavior.
- ◆ Where exceptions are a higher possibility, like with movement commands, On Error Resume Next is used to prevent the exceptions from terminating the script and the message text is checked for the presence of "Warning:". If "Warning:" is found the message is logged and operation continues. If "Warning:" is not found then the message is an Alert or Fatal Error and the script is terminated. Note: even in scripts that have sections with higher possibility of exceptions the On Error GoTo 0 statement is used to re-enabled exceptions (to terminate the script) over sections of code that are not expect to experience exceptions. Note: prior to On Error GoTo 0 any Maestro messages should be cleared. This is accomplished by using the CheckExceptions() routine.
- ◆ In certain rare instances, On Error Resume Next is used to prevent exceptions from terminating the script and messages are polled for (using the WScript Err object) and their description used to make a decision. This method is discouraged since a) message text is not guaranteed to remain consistent as SkyWalker firmware versions advance and b) Maestro will only raise the first message it receives for any property or method access and would loose subsequent messages in a multiple-message burst.
- ◆ Finally, another rarely used method is to use On Error Resume Next, to prevent exceptions from terminating the script, and then poll SkyWalker status to determine the state of operation. The HardLimitBackOut() routine uses this approach. This approach is not recommended except when SkyWalker's GetTopActiveFault ATCL command is continuously polled (using Scope.CommandString("HGtf")) to assure that no serious faults are active.

Note: The Maestro ASCOM Telescope Interface driver will only raise a trappable error when one of the ASCOM telescope properties or methods (for which Maestro provides support) is accessed. This prevents the ASCOM client from being

Observatory Control Script Library

potentially flooded with trappable errors. Therefore, to assure that messages are not missed the ASCOM telescope properties and methods should be polled and/or called on a regular basis (say once every second). An active ASCOM client is likely to do this anyhow to receive, for example, up-to-date celestial coordinates.

Note: any message that Maestro receives from SkyWalker during Maestro's processing of the property or method is raised to the client however any message arriving from SkyWalker after Maestro has returned from the property/method (for example: a SlewTo hits a limit after it has started) will not be raised to the client until the next property/method access.

6. Files

Presently, the files in this script library are tailored for use with Palomar's TFT telescope. The TFT includes Maestro/SkyWalker telescope control, FocusPro focuser control, FilterPro filter wheel control and SwitchPro control of fan and OTA shutters. The TFT does not include DomePro dome control. The dome control scripts in this library are tailored for the Tenagra 32" telescope detailed at <http://www.tenagraobservatories.com/> (which includes a 14' Ash dome controlled by DomePro).

File list with descriptions...

```
.\
- Setup.wsf: Observatory setup script. Establishes all necessary observatory equipment settings.
  This script (or portions of it) need only be run once unless new hardware is acquired and
  fresh settings need to be established.
- Startup.wsf: Observatory startup script. Prepares all observatory equipment for
  use/observations.
- StartupLoop.wsf: An iterative version of Startup.wsf using random SlewToAltAz and random
  SyncToAltAz to simulate arbitrary/bogus startup positions. Used by Astrometric
  Instruments for testing.
- Operate.wsf: Example observatory operational script.
- Shutdown.wsf: Observatory shutdown script. Shuts-down telescope and accessories.
.\Common
- Util.vbs: contains utility routines for observatory operation. In particular, the Report
  routine is used through-out all ScriptLib code to simultaneously report script messages to
  the console and a log file.
- Constants.vbs: contains definitions of useful constants used throughout other scripts.
.\Telescope
- Setup.vbs: Telescope setup script called from .\Setup.wsf (not run stand-alone). Establishes
  all necessary telescope (SkyWalker) settings.
- Startup.vbs: Telescope startup script. Prepares telescope (SkyWalker) for
  use/observations.
.\Telescope\Common
- TUtil.vbs: telescope-related utility routines
- AlignFromHomeSeek.vbs: contains the AlignFromHomeSeek function used in .\Telescope\Startup.vbs
.\Telescope\Misc
- AcqAndTrack.vbs: a script exemplifying how to acquire and track a target when using
  high resolution axial encoders.
- Tweak.vbs: a script showing how to refine a GoTo using axial encoders.
- DynaCorr.vbs: script that sets SkyWalker's DynaCorr correction coefficients.
.\Telescope\ASCOM_Test: contains many small/simple scripts for testing various ASCOM properties
  and methods.
.\Telescope\Tours: scripts that successively GoTo interesting double stars.
.\Dome
- Setup.vbs: Dome setup script called from .\Setup.wsf (not run stand-alone). Establishes
  all necessary dome (DomePro) settings.
- Startup.vbs: Dome startup script. Prepares dome (DomePro) for use/observations.
.\Dome\ASCOM_Test: contains many small/simple scripts for testing various ASCOM properties
  and methods.
.\Focus
- Setup.vbs: Focuser setup script called from .\Setup.wsf (not run stand-alone). Establishes
  all necessary focuser (FocusPro) settings.
```

Observatory Control Script Library

- StartUp.vbs: Focus startup script. Prepares focuser (FocusPro) for use/observations.
- .\Focus\ASCOM_Test: contains many small/simple scripts for testing various ASCOM properties and methods.
- .\Filter
 - Setup.vbs: Filterwheel setup script called from .\Setup.wsf (not run stand-alone). Establishes all necessary filterwheel (FilterPro) settings.
 - Startup.vbs: Filterwheel startup script. Prepares filterwheel (FilterPro) for use/observations.
- .\Filter\ASCOM_Test: contains many small/simple scripts for testing various ASCOM properties and methods.
- .\Switches
 - Setup.vbs: switches setup script called from .\Setup.wsf (not run stand-alone). Establishes all necessary switches (SwitchPro) settings.
 - Startup.vbs: Switches startup script. Prepares switches (SwitchPro) for accessory control.
- .\MiscDocs
 - <see list in "Companion Documentation" section above>