

FC 5 AppleScript and FlightScript with FLIGHTCHECK® 5

FLIGHTCHECK® 5 supports two different 'dialects' of AppleScript. One is nicknamed 'FlightScript'. FlightScript finds its origins with FLIGHTCHECK® 4.5. The other dialect is FLIGHTCHECK® 5's native FC 5 AppleScript implementation, which ties strongly into the MarkzONE database engine that makes up the core of FLIGHTCHECK® 5. These dialects are complementary: some features are only available through FlightScript, some only only through FC 5 AppleScript

FLIGHTCHECK® 5 and FLIGHTCHECK® 4.5 are quite different, and as a result, FlightScript compatibility between the two versions is not 100%: most FlightScript statements are supported, but not all.

This document lists all the supported and unsupported FlightScript commands. For the unsupported commands, it shows how to get equivalent functionality through FLIGHTCHECK® 5's native AppleScript commands whenever possible.

The Default FlightPlan should be loaded into FLIGHTCHECK® 5 for proper FlightScript support. FlightScript is somewhat reliant on a particular user interface being present, and because the Default FlightPlan emulates the FLIGHTCHECK® 4.5 interface it needs to be loaded for FlightScript to work.

1 FlightScript Commands

This is an alphabetical list of the various FLIGHTCHECK® 4.5 FlightScript commands, and how they are supported by FLIGHTCHECK® 5.0.

1.1 add fonts folder "<fontfolderpath>"

Supported. FLIGHTCHECK® 5 uses a slightly different font database mechanism than FLIGHTCHECK® 4.5: the fonts database is managed automatically instead of manually, and FLIGHTCHECK adds and removes fonts from it on an as-needed basis.

FLIGHTCHECK® 5 also supports relative fonts folder paths, which are considered relative to the folder containing the main document – for example the folder “:Fonts” (with a leading semicolon) is considered to be a subfolder of the folder that also contains the main document.

This command add the fonts folder path to the list of font search paths.

See also `remove all fonts folders`, `remove fonts folder`.

1.2 add images folder "<imagesfolderpath>"

This command was not supported in FLIGHTCHECK® 4.5. It was introduced in FLIGHTCHECK® 5 to allow access to the image search folders.

FLIGHTCHECK® 5 also supports relative image folder paths, which are considered relative to the folder containing the main document – for example the folder “:Images” (with a leading semicolon) is considered to be a subfolder of the folder that also contains the main document.

This command add the images folder path to the list of image search paths.

See also `remove all images folders`, `remove images folder`.

1.3 close form

Not supported.

1.4 collect fonts

Not supported. FLIGHTCHECK® 5 has a different approach to collecting: a collected job consists of files loosely arranged into one or more collected ‘file classes’. The currently active FlightPlan determines what file classes are available – this is at the discretion of the FlightPlan developer.

Some FlightPlans will not necessarily offer a class called ‘Fonts’. Others might make a distinction between different kinds of fonts, and hence support ‘TrueType Fonts’, ‘Type 1 Fonts’, ‘OpenType Fonts’,...

FlightPlan file classes can be referred to by numerical index or by a name. The names are chosen by the FlightPlan developer. To list all the available FlightPlan classes for a particular FlightPlan, one can run the following AppleScript:

```
tell application "FLIGHTCHECK®.app"
    set theFileClassList to every file class
    set theFileNameList to {}
    repeat with theFileClass in theFileClassList
        set theFileNameList to ~
            theFileNameList & ~
                {the name of theFileClass}
    end repeat
end tell
```

This lists all the names of the file classes; for the Default FlightPlan these are:

CL_DOCUMENT_CLASS_NAME	top level documents
CL_IMAGE_CLASS_NAME	used images
CL_FONT_CLASS_NAME	loose font files (e.g. .TTF)
CL_SUITCASE_CLASS_NAME	font suitcases
CL_TYPE1_FONT_CLASS_NAME	Type 1 font files
CL_COLLECT_REPORT_CLASS_NAME	Collect report

By using these identifiers one can easily access individual file classes and set up the various properties.

Typically, you want to set these properties as early as possible in a read-collect cycle. File class properties like `collect` and `used only` serve only as defaults to use for individual file objects when the job is actually collected. Individual file objects can have their own individual `collect` setting which might or might not differ from the defaults defined for the file class.

To collect only the fonts, one could use the following script:

```
tell application "FLIGHTCHECK@.app"

--
-- First turn off the 'collect' property of all file
-- classes
--
set theFileClassList to every file class

set theFileClassNameList to {}
repeat with theFileClass in theFileClassList
    set the collect of theFileClass to false
end repeat

--
-- Then grab just the font-related file classes and set
-- their collect property back to true
--
set the collect of ↵
    file class "CL_FONT_CLASS_NAME" ↵
    to true
set the collect of ↵
    file class "CL_SUITCASE_CLASS_NAME" ↵
    to true
set the collect of ↵
    file class "CL_TYPE1_FONT_CLASS_NAME" ↵
    to true

--
-- Now, perform the collection. This assumes we have not
-- collected the job before. In that case, the 'collect'
-- property of the individual files is undefined, and
-- FLIGHTCHECK will use the settings of the
-- file classes to initialize each individual file's
-- 'collect' property
-- Hence: only font files will be collected
--

do collection to ↵
    "HeavyMetal:Users:kris:Desktop:CollectFolder" ↵
    name "MyJob.zip" ↵
    with compress

end tell
```

1.5 collect fonts suitcase

Not supported. See `collect font` for more information.

1.6 collect images

Not supported. See `collect font` for more information.

1.7 collect job **collect job "<pathtofolder>"**

Supported. Collects the job. <pathtofolder> is an optional path name of the folder to collect into. Example:

```

tell application "FLIGHTCHECK@.app"

    flightcheck "HeavyMetal:Users:kris:Desktop:Document1"
    collect job

end tell

```

See also the `do collection...` AppleScript command and the `file class` objects.

1.8 collect report

collect report "<pathtoreport>"

Supported. Saves a Collect Report. <pathtoreport> is an optional path name of the file to save the report into. Example:

```

tell application "FLIGHTCHECK@.app"

    flightcheck "HeavyMetal:Users:kris:Desktop:Document1"
    collect report "HeavyMetal:Users:kris:Desktop:Document1.txt"

end tell

```

1.9 count problems

Supported. Returns the number of flagged items.

```

tell application "FLIGHTCHECK@.app"

    count problems
    copy the result to theProblemCount

    display dialog (theProblemCount as string) & " problems found"
end tell

```

1.10 deselect compress job

Supported. Causes the 'Compress Job' option on the Collect Screen to be deselected.

See also the `with compress` option of the `do collection` AppleScript command

1.11 do full flightcheck

Supported. When one or more documents are currently open, they will be re-checked. Example:

```

tell application "FLIGHTCHECK@.app"

    -- Temporarily switch off sound alert
    set savedAlertSetting to play alert sound
    set play alert sound to false
    do full flightcheck
    set play alert sound to savedAlertSetting

end tell

```

This example rechecks the currently open documents. It first saves the current alert sound setting, then turns the alert sound off for the duration of the re-check, and finally restores the alert sound setting to its original state.

1.12 flightcheck "<pathtofile>"

Supported. If a document or set of documents are currently open, they will be closed first before the document is preflighted. Example:

```
tell application "FLIGHTCHECK®.app"
    flightcheck "HeavyMetal:Users:kris:Desktop:scaled.qxd"
end tell
```

This example will close the current document(s), if any, and then preflight the document scaled.qxd

The `flightcheck` command merely emulates FLIGHTCHECK® 4.5 and it does not allow checking of multiple files as part of a single job. To accomplish that you need to use the `new job` and `add to job` AppleScript commands instead of the `flightcheck` command.

See also: `open`, `add to job`, `new job` in the “AppleScript Commands” section of this document.

1.13 get application info

Not supported. In FLIGHTCHECK® 5, the various information panels in the Overview Window are built dynamically by the FlightPlan. FlightPlans can have many forms and shapes – most of which might not have an ‘Application Info’ panel.

To get the same, and more, information out of the FLIGHTCHECK® 5, one can use an AppleScript modelled after one the following examples.

The first example retrieves the width and the height of all the top level documents currently displayed and puts them into a string ‘report’.

```
tell application "FLIGHTCHECK®.app"

    set report to ""
    set cr to ASCII character 13
    set docList to every top level document of job
    repeat with doc in docList
        set report to report & "Document name: " & the name of doc & cr
        set report to report & "Width: " & the width of doc & cr
        set report to report & "Height: " & the height of doc & cr
    end repeat

    return report
end tell
```

This ties in directly into MarkzONE – the variable `doc` refers directly to MarkzONE objects stored in the MarkzONE database. The script above is using direct access to the MarkzONE attributes of the DOCUMENT objects that are returned in a list by `every top level document of job`.

FLIGHTCHECK® 5 makes the MarkzONE attributes of any object available to AppleScript by their name. In this case, three attributes are retrieved: NAME, WIDTH, and HEIGHT – MarkzONE attribute names are not case sensitive, hence in the script above they have been written in lower case for consistency.

The list of available attributes for any particular object type is easiest to determine by using the Scout utility, or by getting hold of a copy of the ‘FLIGHTCHECK® Workflow Reference Manual’ which documents the internal data structures used by MarkzONE.

Another approach to extract information is to use an embedded MarkzONE script – e.g. if one has access to a MarkzONE script as used in FLIGHTCHECK® Workflow, it can be converted into an embedded MarkzONE script for use with FLIGHTCHECK® 5. This script can fetch the required data and pass the data back to AppleScript.

First we'll look at the MarkzONE. Careful: this script is *not* an AppleScript – it cannot be simply entered in the Script Editor, as it won't be understood by the AppleScript engine.

```
PROC ProcessFile
  cr = CHR(13)
  theReport = theReport + 'Document: ' + it.NAME + cr
  theReport = theReport + 'File Type: ' + it.CLASS_NAME + cr
  IF it.platform IS DEFINED THEN
    theReport = theReport + 'Platform: ' + it.PLATFORM + cr
  END IF
  IF it.creator IS DEFINED THEN
    theReport = theReport + 'Creator: ' + it.CREATOR + cr
  END IF
  IF it.version IS DEFINED THEN
    theReport = theReport + 'Version: ' + it.VERSION + cr
  END IF
  IF it.width IS DEFINED THEN
    theReport = theReport + 'Width (inch): ' + it.WIDTH + cr
  END IF
  IF it.height IS DEFINED THEN
    theReport = theReport + 'Height (inch): ' + it.HEIGHT + cr
  END IF
  IF it.width IS DEFINED THEN
    theReport = theReport + 'Width (cm): ' + it.WIDTH.CENTIMETER + cr
  END IF
  IF it.height IS DEFINED THEN
    theReport = theReport + 'Height (cm): ' + it.HEIGHT.CENTIMETER + cr
  END IF
END PROC

theReport = ''
theJob = it
SCAN RESET
ON DOCUMENT DO ProcessFile
SCAN theJob LINKS CONTAINS
RETURN theReport
```

This script will scan the job, and visit all top-level documents it can find. For each of them it will report some MarkzONE attributes if they are defined: the NAME, the CLASS_NAME (i.e. the file format as recognized by MarkzONE), the PLATFORM (Mac or Windows), the CREATOR (the name of the application used to create the file), the VERSION (the version of the file format), the WIDTH and HEIGHT (in inch and in cm).

To make this script usable from AppleScript we need to ‘embed’ – i.e. we need to create a long string variable that contains the above script. The end-result is not easy to read, but essentially, it is the same as the script above, with some syntactic sugar added.

```

tell application "FLIGHTCHECK®.app"

    set cr to ASCII character 13

    set theMarkzONEScript to ¬
"PROC ProcessFile" & cr & ¬
" cr = CHR(13)" & cr & ¬
" theReport = theReport + 'Document: ' + it.name + cr" & cr & ¬
" theReport = theReport + 'File Type: ' + it.class_name + cr" & cr & ¬
" IF it.platform IS DEFINED THEN" & cr & ¬
"     theReport = theReport + 'Platform : ' + it.platform + cr" & cr & ¬
" END IF" & cr & ¬
" IF it.creator IS DEFINED THEN" & cr & ¬
"     theReport = theReport + 'Creator: ' + it.creator + cr" & cr & ¬
" END IF" & cr & ¬
" IF it.version IS DEFINED THEN" & cr & ¬
"     theReport = theReport + 'Version: ' + it.version + cr" & cr & ¬
" END IF" & cr & ¬
" IF it.width IS DEFINED THEN" & cr & ¬
"     theReport = theReport + 'Width (inch)' + it.width + cr" & cr & ¬
" END IF" & cr & ¬
" IF it.height IS DEFINED THEN" & cr & ¬
"     theReport = theReport + 'Height (inch): ' + it.height + cr" & cr & ¬
" END IF" & cr & ¬
" IF it.width IS DEFINED THEN" & cr & ¬
"     theReport = theReport + 'Width (cm): ' + it.width.centimeter + cr" & cr & ¬
" END IF" & cr & ¬
" IF it.height IS DEFINED THEN" & cr & ¬
"     theReport = theReport + 'Height (cm): ' + it.height.centimeter + cr" & cr & ¬
" END IF" & cr & ¬
"END PROC" & cr & ¬
" & cr & ¬
"theReport = ''" & cr & ¬
"theJob = it" & cr & ¬
"SCAN RESET" & cr & ¬
"ON DOCUMENT DO ProcessFile" & cr & ¬
"SCAN theJob LINKS CONTAINS" & cr & ¬
"RETURN theReport"

    do script theMarkzONEScript with the job
    set report to the result

end tell

```

See also: FLIGHTCHECK® Workflow Reference Manual, the `do script AppleScript` command, the object `AppleScript` class.

1.14 get box clipping path images

Not supported. In FLIGHTCHECK® 5, the various information panels in the Overview Window are built dynamically by the FlightPlan. FlightPlans can have many forms and shapes – most of which might not have an ‘Image Info’ panel.

See also: the `get application info FlightScript` command, the FLIGHTCHECK® Workflow Reference Manual, the `do script AppleScript` command, the object `AppleScript` class and its subclasses for examples of alternate ways to access this info.

1.15 get clipping path images

Not supported. In FLIGHTCHECK® 5, the various information panels in the Overview Window are built dynamically by the FlightPlan. FlightPlans can have many forms and shapes – most of which might not have an ‘Image Info’ panel.

See also: the `get application info` FlightScript command, the FLIGHTCHECK® Workflow Reference Manual, the `do script` AppleScript command, the `object` AppleScript class and its subclasses for examples of alternate ways to access this info.

1.16 `get color list`

Supported. When no document is open, an empty list will be returned. The list of color names that is returned depends on the current 'filter' setting for colors in the overview window - the visual color selection is reflected in the returned list. Example:

```
tell application "FLIGHTCHECK®.app"

    flightcheck "HeavyMetal:Users:kris:Desktop:scaled.qxd"
    set color view to Flagged Colors
    get color list
    set theFlaggedColorList to the result

    display dialog "There are " ~
        & (the count of theFlaggedColorList) ~
        & " flagged colors"

end tell
```

This example will preflight `scaled.qxd`, set the color view to show only the flagged colors, and retrieve the list of color names. It then displays a dialog mentioning the number of flagged colors.

The example above can also be rewritten using FC 5 AppleScript, which does not rely on a particular view to be selected. It also does not necessitate the `Default.fpn.xml` FlightPlan to be active.

```
tell application "FLIGHTCHECK®.app"
    new job "HeavyMetal:Users:kris:Desktop:scaled.qxd"

    -- First fetch all the colors
    set theColorList to every color object of job

    -- Make a list of colors that are flagged with
    -- a warning or an error
    set theFlaggedColorList to {}
    repeat with theColor in theColorList
        if error level of theColor > 0 then
            set theFlaggedColorList to ~
                theFlaggedColorList & {theColor}
        end if
    end repeat

    display dialog "There are " ~
        & (the count of theFlaggedColorList) ~
        & " flagged colors"

end tell
```

Remark: currently, FLIGHTCHECK® 5 does not implement the 'whose' AppleScript selector, which would allow the repeat loop above to be expressed in a single line.

One of the main differences between the two scripts is that the FlightScript-based script will receive a list of strings, whereas the FC 5 AppleScript version will receive a list of MarkzONE objects.

In the FC 5 AppleScript version it is possible further process the individual color objects to get access to more specific color attributes – e.g. one can iterate the list and for example access the `color_model` of each individual color.

See also: the `set color view to FlightScript` command

1.17 get color view

Supported. Returns a string that represents the currently selected view for colors.

The visual naming of the selection in the filtering popup menu on the Overview Window will not necessarily correspond to the FLIGHTCHECK® 4.5 naming. However, for compatibility, the returned string will conform to the string that would have been returned by FLIGHTCHECK® 4.5.

See also: `set color view to ...` for more information about the mapping between FLIGHTCHECK® 4.5 view names and FLIGHTCHECK® 5 rule codes.

1.18 get document title

Supported. Returns a string with the document name, or alternatively, a list of strings in case FLIGHTCHECK® 5 has multiple documents open concurrently.

1.19 get file info

Not supported. In FLIGHTCHECK® 5, the various information panels in the Overview Window are built dynamically by the FlightPlan. FlightPlans can have many forms and shapes – most of which might not have a ‘File Info’ panel.

To get the same, and more, information out of the FLIGHTCHECK® 5, one can use an AppleScript modelled after the examples you can find in the paragraph about `get application info`.

See also: the `get application info FlightScript` command, the FLIGHTCHECK® Workflow Reference Manual, the `do script AppleScript` command, the `object AppleScript` class.

1.20 get flightcheck results

Supported. Returns a string with the current FlightCheck results. The ‘layout’ of the returned string depends on the open/close status of the various elements in the Results Window. Example:

```
tell application "FLIGHTCHECK®.app"
    show specific items
    get flightcheck results
    put the result into theExtendedResults
end tell
```

This example first opens up all the subdivisions in the results window, and then fetches the results of the window as a string into the `result`.

1.21 get flightcheck time

Supported. Returns the number of seconds that were spent by FLIGHTCHECK® 5 working on preflighting the job. Example:

```
tell application "FLIGHTCHECK®.app"

    flightcheck "HeavyMetal:Users:kris:Desktop:Document1"
    get flightcheck time
    if the result > 10 then
        display dialog "Time to get a G5"
    end if

end tell
```

1.22 get font list

Supported. When no document is open, an empty list will be returned. The list of font names that is returned depends on the current 'filter' setting for fonts in the overview window - the visual font selection is reflected in the returned list. Example:

```
tell application "FLIGHTCHECK®.app"

    flightcheck "HeavyMetal:Users:kris:Desktop:Document1"
    set font view to Missing Fonts
    get font list
    set theMissingFontList to the result

    return theMissingFontList

end tell
```

This example will preflight Document1, then set the font view to show only the missing fonts, then retrieve the list of font names. It then displays the font name list in the Script Editor "Result" zone.

The example above can also be rewritten using FC 5 AppleScript, which does not rely on a particular view to be selected. This example relies on the presence of the Default FlightPlan because it needs access to the rule with the code FONT_REQUEST_PRINTER_FONT which is provided by the Default.fpn.xml FlightPlan.

```

tell application "FLIGHTCHECK®.app"

    new job "HeavyMetal:Users:kris:Desktop:Document1"

    set theFlaggedFontRequestList to {}
    try
        --
        -- Fetch the MarkzONE flag object that corresponds
        -- with missing printer fonts.
        -- This will fail if the flag does not exist
        -- (meaning: no flagged font requests), so we put
        -- it inside a try/end try block.
        --
        set theMissingFontFlag to ↵
            flag object "FONT_REQUEST_PRINTER_FONT"
        set theFlaggedFontRequestList to ↵
            every flagged object of theMissingFontFlag
    end try

    --
    -- Build a list of names from the list of objects
    -- so we have something to display
    --
    set theMissingFontList to {}
    repeat with theFontRequest in theFlaggedFontRequestList
        set theMissingFontList to ↵
            theMissingFontList & ↵
                {the name of theFontRequest}
    end repeat

    return theMissingFontList

end tell

```

See also: the set font view to FlightScript command

1.23 get font view

Supported. Returns a string that represents the currently selected view for fonts.

The visual naming of the selection in the filtering popup menu on the Overview Window will not necessarily correspond to the FLIGHTCHECK® 4.5 naming. However, for compatibility, the returned string will conform to the string that would have been returned by FLIGHTCHECK® 4.5.

See also: set font view to ... for more information about the mapping between FLIGHTCHECK® 4.5 view names and FLIGHTCHECK® 5 rule codes.

1.24 get ground controls title

Supported. Example:

```

tell application "FLIGHTCHECK®.app"

    get ground controls title
    display dialog ↵
        "The currently selected set is " ↵
        & the result

end tell

```

This example will select the default Ground Control Set if it exists. Nothing happens if the set does not exist.

1.25 get halftone screen images

Not supported. In FLIGHTCHECK® 5, the various information panels in the Overview Window are built dynamically by the FlightPlan. FlightPlans can have many forms and shapes – most of which might not have an ‘Image Info’ panel.

See also: the `get application info` FlightScript command, the FLIGHTCHECK® Workflow Reference Manual, the `do script` AppleScript command, the `object` AppleScript class and its subclasses for examples of alternate ways to access this info.

1.26 get image colors

Not supported. In FLIGHTCHECK® 5, the various information panels in the Overview Window are built dynamically by the FlightPlan. FlightPlans can have many forms and shapes – most of which might not have an ‘Image Info’ panel.

To get the same, and more, information out of the FLIGHTCHECK® 5, one can use an AppleScript modelled after one the following example.

```
tell application "FLIGHTCHECK®.app"

    set report to ""
    set cr to ASCII character 13
    set docList to every top level document of job
    repeat with doc in docList
        set report to report & "Image list for " & the name of doc & cr
        set imageList to every used image of doc
        repeat with img in imageList
            set report to report & ¬
                " Color list for image " & ¬
                the filename of img & cr
            set colorList to every color object of img
            if the length of colorList = 0 then
                set report to report & ¬
                    " -- no colors -- " & cr
            else
                repeat with clr in colorList
                    set report to report & ¬
                        " " & the name of clr & cr
                end repeat
            end if
        end repeat
    end repeat

    return report
end tell
```

This ties in directly into MarkzONE – the variables `doc`, `img`, `clr` refer directly to MarkzONE objects stored in the MarkzONE database. The script above is using direct access to the MarkzONE attributes of the objects.

FLIGHTCHECK® 5 makes the MarkzONE attributes of any object available to AppleScript by their name.

The list of available attributes for any particular object type is easiest to determine by using the Scout utility, or by getting hold of a copy of the 'FLIGHTCHECK® Workflow Reference Manual' which documents the internal data structures used by MarkzONE.

See also: the `get application info FlightScript` command, the FLIGHTCHECK® Workflow Reference Manual, the `do script AppleScript` command, the `object AppleScript` class and its subclasses.

1.27 get image fonts

Not supported. In FLIGHTCHECK® 5, the various information panels in the Overview Window are built dynamically by the FlightPlan. FlightPlans can have many forms and shapes – most of which might not have an 'Image Info' panel.

To get the same, and more, information out of the FLIGHTCHECK® 5, one can use an AppleScript modelled after one the following example.

```
tell application "FLIGHTCHECK®.app.debug"

    set report to ""
    set cr to ASCII character 13
    set docList to every top level document of job
    repeat with doc in docList
        set report to report ~
            & "Image list for " ~
            & the name of doc ~
            & cr
        set imageList to every used image of doc
        repeat with img in imageList
            set fontList to every font request object of img
            if the length of fontList > 0 then
                set report to report ~
                    & " Font list for image " ~
                    & the filename of img ~
                    & cr
                repeat with fnt in fontList
                    set report to report ~
                        & " " ~
                        & the name of fnt & cr
                end repeat
            end if
        end repeat
    end repeat

    return report
end tell
```

This ties in directly into MarkzONE – the variables `doc`, `img`, `clr` refer directly to MarkzONE objects stored in the MarkzONE database. The script above is using direct access to the MarkzONE attributes of the objects.

FLIGHTCHECK® 5 makes the MarkzONE attributes of any object available to AppleScript by their name.

The list of available attributes for any particular object type is easiest to determine by using the Scout utility, or by getting hold of a copy of the 'FLIGHTCHECK® Workflow Reference Manual' which documents the internal data structures used by MarkzONE.

See also: the `get application info FlightScript` command, the FLIGHTCHECK® Workflow Reference Manual, the `do script AppleScript` command, the object `AppleScript` class and its subclasses.

1.28 `get image list`

Supported. When no document is open, an empty list will be returned. The list of image path names that is returned depends on the current 'filter' setting for images in the overview window - the visual image selection is reflected in the returned list. Example:

```
tell application "FLIGHTCHECK®.app"

    flightcheck "HeavyMetal:Users:kris:Desktop:Document1"
    set image view to Missing
    get image list
    set theMissingImageList to the result

    return theMissingImageList

end tell
```

This example will preflight `Document1`, then set the image view to show only the missing images, retrieve the list of image path names. It then displays the image path name list in the Script Editor "Result" zone.

The example above can also be rewritten using FC 5 AppleScript, which does not rely on a particular view to be selected. This example relies on the presence of the Default FlightPlan because it needs access to the rule with the code `LOCATION_MISSING` which is provided by the `Default.fpn.xml` FlightPlan.

```

tell application "FLIGHTCHECK®.app"

    new job "HeavyMetal:Users:kris:Desktop:Document1"

    set theFlaggedImageContainerList to {}
    try
        --
        -- Fetch the MarkzONE flag object that corresponds
        -- with missing images.
        -- This will fail if the flag does not exist
        -- (meaning: no missing images), so we put it
        -- inside a try/end try block
        --
        set theMissingImageFlag to ~
            flag object "LOCATION_MISSING"
        --
        -- Given the flag, retrieve the flagged objects
        --
        set theFlaggedImageContainerList to ~
            every flagged object of theMissingImageFlag
    end try

    --
    -- Build a list of names from the list of objects
    -- so we have something to display.
    -- This is a list of image containers - so we need to
    -- fetch the image shown in the container
    --
    set theMissingImageList to {}
    repeat with theImageContainer in ~
        theFlaggedImageContainerList
        --
        -- Retrieve the image being displayed in the
        -- container
        --
        set theImage to the image of theImageContainer
        set theMissingImageList to ~
            theMissingImageList & ~
                {the fullpath of theImage}
    end repeat

    return theMissingImageList

end tell

```

In this example, some of the differences in approach between FLIGHTCHECK® 4.5 and FLIGHTCHECK® 5 become more apparent. In this case, the flag object (`LOCATION_MISSING`) is used to flag *image containers*, not images.

Hence `theFlaggedImageContainerList` will contain a list of MarkzONE objects which are all representing image containers (e.g. image boxes).

To get the same output as the FlightScript script, we need to iterate through the list of containers, and first use the construct the image of `<thecontainer>` to fetch the MarkzONE FILE object that is displayed in each container.

Finally, we need to create a list of the `fullpath` of `theImage` to mimic FLIGHTCHECK® 4.5's behavior - `theImage` contains a MarkzONE object, not a string. `fullpath` is a MarkzONE attribute available for FILE objects.

See also: the `set image view to FlightScript` command

1.29 get image names list

Supported. When no document is open, an empty list will be returned. The list of images that is returned depends on the current 'filter' setting for images – the visual image selection is reflected in the returned list.

This is very similar to the `get image list` command. The difference between `get image list` and `get image names list` is that the first returns a list of full file path names, and the second returns a list of just short file names, without mention of the folders the images are located in. Example:

```
tell application "FLIGHTCHECK@.app"

    flightcheck "HeavyMetal:Users:kris:Desktop:Document1"
    set image view to All Images
    get image names list
    set theCompleteImageNameList to the result

    return theCompleteImageNameList

end tell
```

This example will preflight `Document1`, then set the image view to show all images, then retrieves the list of image names. It then displays the image name list in the Script Editor "Result" zone.

The example above can also be rewritten using FC 5 AppleScript, which does not rely on a particular view to be selected. It also does not necessitate the `Default.fpn.xml` FlightPlan to be active.

```
tell application "FLIGHTCHECK@.app"

    new job "HeavyMetal:Users:kris:Desktop:Document1"

    set theImageList to every used image of job
    --
    -- Build a list of names from the list of objects
    -- so we have something to display.
    --
    set theCompleteImageNameList to {}
    repeat with theImage in theImageList
        set theCompleteImageNameList to ~
            theCompleteImageNameList & ~
                {the filename of theImage}
    end repeat

    return theCompleteImageNameList

end tell
```

To get the same output as the FlightScript script, we need to iterate through the list of images, and use the construct `the filename of theImage` to fetch the name of the MarkzONE FILE object.

See also: the `set image view to ... FlightScript` command

1.30 get ICC Profile images

Not supported. In FLIGHTCHECK® 5, the various information panels in the Overview Window are built dynamically by the FlightPlan. FlightPlans can have many forms and shapes – most of which might not have an ‘Image Info’ panel.

See also: the `get application info` FlightScript command, the FLIGHTCHECK® Workflow Reference Manual, the `do script AppleScript` command, the `object AppleScript` class and its subclasses for examples of alternate ways to access this info.

1.31 get image view

Supported. Returns a string that represents the currently selected view for images.

The visual naming of the selection in the filtering popup menu on the Overview Window will not necessarily correspond to the FLIGHTCHECK® 4.5 naming. However, for compatibility, the returned string will conform to the string that would have been returned by FLIGHTCHECK® 4.5.

See also: `set image view to ...` for more information about the mapping between FLIGHTCHECK® 4.5 view names and FLIGHTCHECK® 5 rule codes.

1.32 get linear transfer function images

Not supported. In FLIGHTCHECK® 5, the various information panels in the Overview Window are built dynamically by the FlightPlan. FlightPlans can have many forms and shapes – most of which might not have an ‘Image Info’ panel.

See also: the `get application info` FlightScript command, the FLIGHTCHECK® Workflow Reference Manual, the `do script AppleScript` command, the `object AppleScript` class and its subclasses for examples of alternate ways to access this info.

1.33 get nonlinear transfer function images

Not supported. In FLIGHTCHECK® 5, the various information panels in the Overview Window are built dynamically by the FlightPlan. FlightPlans can have many forms and shapes – most of which might not have an ‘Image Info’ panel.

See also: the `get application info` FlightScript command, the FLIGHTCHECK® Workflow Reference Manual, the `do script AppleScript` command, the `object AppleScript` class and its subclasses for examples of alternate ways to access this info.

1.34 get OPI images

Not supported. In FLIGHTCHECK® 5, the various information panels in the Overview Window are built dynamically by the FlightPlan. FlightPlans can have many forms and shapes – most of which might not have an ‘Image Info’ panel.

See also: the `get application info` FlightScript command, the FLIGHTCHECK® Workflow Reference Manual, the `do script AppleScript` command, the `object AppleScript` class and its subclasses for examples of alternate ways to access this info.

1.35 get page info

Not supported. In FLIGHTCHECK® 5, the various information panels in the Overview Window are built dynamically by the FlightPlan. FlightPlans can have many forms and shapes – most of which might not have a ‘Page Info’ panel.

To get the same, and more, information out of the FLIGHTCHECK® 5, one can use an AppleScript modelled after the examples you can find in the paragraph about `get application info`.

See also: the `get application info` FlightScript command, the FLIGHTCHECK® Workflow Reference Manual, the `do script` AppleScript command, the `object` AppleScript class.

1.36 get page setup info

Not supported. In FLIGHTCHECK® 5, the various information panels in the Overview Window are built dynamically by the FlightPlan. FlightPlans can have many forms and shapes – most of which might not have an ‘Page Info’ panel.

To get the same, and more, information out of the FLIGHTCHECK® 5, one can use an AppleScript modelled after the examples you can find in the paragraph about `get application info`.

See also: the `get application info` FlightScript command, the FLIGHTCHECK® Workflow Reference Manual, the `do script` AppleScript command, the `object` AppleScript class.

1.37 get printer info

Not supported. In FLIGHTCHECK® 5, the various information panels in the Overview Window are built dynamically by the FlightPlan. FlightPlans can have many forms and shapes – most of which might not have a ‘Printer Info’ panel.

To get the same, and more, information out of the FLIGHTCHECK® 5, one can use an AppleScript modelled after the examples you can find in the paragraph about `get application info`.

See also: the `get application info` FlightScript command, the FLIGHTCHECK® Workflow Reference Manual, the `do script` AppleScript command, the `object` AppleScript class.

1.38 get PS Color images

Not supported. In FLIGHTCHECK® 5, the various information panels in the Overview Window are built dynamically by the FlightPlan. FlightPlans can have many forms and shapes – most of which might not have an ‘Image Info’ panel.

See also: the `get application info` FlightScript command, the FLIGHTCHECK® Workflow Reference Manual, the `do script` AppleScript command, the `object` AppleScript class and its subclasses for examples of alternate ways to access this info.

1.39 get RGB images

Supported. When no document is open, an empty list will be returned. Example:

```
tell application "FLIGHTCHECK@.app"

    flightcheck "HeavyMetal:Users:kris:Desktop:Document1"
    get RGB images
    set theRGBImageList to the result

    return theRGBImageList

end tell
```

This example will preflight Document1, then retrieves a list of RGB image names. It then displays the image name list in the Script Editor "Result" zone.

The example above can also be rewritten using FC 5 AppleScript, which does not rely on a particular view to be selected. This example relies on the presence of a FlightPlan that provides an implementation for the rule FILE_MODE_RGB_COLOR (e.g. Default.fpn.xml)

```
tell application "FLIGHTCHECK@.app"

    new job "HeavyMetal:Users:kris:Desktop:Document1"

    set theRGBImageList to {}
    try
        --
        -- Fetch the MarkzONE flag object that corresponds
        -- with RGB images.
        -- This will fail if the flag does not exist
        -- (meaning: no RGB images), so we put it
        -- inside a try/end try block
        --
        set theRGBImageFlag to ~
            flag object "FILE_MODE_RGB_COLOR"
        --
        -- Given the flag, retrieve the flagged objects
        --
        set theRGBImageList to ~
            every flagged object of theRGBImageFlag
    end try

    --
    -- Build a list of names from the list of objects
    -- so we have something to display.
    -- This is a list of image containers - so we need to
    -- fetch the image shown in the container
    --
    set theRGBImageNameList to {}
    repeat with theImage in theRGBImageList
        --
        -- Retrieve the image being displayed in the
        -- container
        --
        set theRGBImageNameList to ~
            theRGBImageNameList & ~
                {the filename of theImage}
    end repeat

    return theRGBImageNameList

end tell
```

To get the same output as the FlightScript script, we need to iterate through the list of images, and use the construct the filename of the `Image` to fetch the file name of the MarkzONE file object.

1.40 get stylized images

Not supported. In FLIGHTCHECK® 5, the various information panels in the Overview Window are built dynamically by the FlightPlan. FlightPlans can have many forms and shapes – most of which might not have an ‘Image Info’ panel.

See also: the `get application info` FlightScript command, the FLIGHTCHECK® Workflow Reference Manual, the `do script` AppleScript command, the `object` AppleScript class and its subclasses for examples of alternate ways to access this info.

1.41 get TIFF clipping path images

Not supported. In FLIGHTCHECK® 5, the various information panels in the Overview Window are built dynamically by the FlightPlan. FlightPlans can have many forms and shapes – most of which might not have an ‘Image Info’ panel.

See also: the `get application info` FlightScript command, the FLIGHTCHECK® Workflow Reference Manual, the `do script` AppleScript command, the `object` AppleScript class and its subclasses for examples of alternate ways to access this info.

1.42 get trap info

Not supported. In FLIGHTCHECK® 5, the various information panels in the Overview Window are built dynamically by the FlightPlan. FlightPlans can have many forms and shapes – most of which might not have a ‘Trap Info’ panel.

To get the same, and more, information out of the FLIGHTCHECK® 5, one can use an AppleScript modelled after the examples you can find in the paragraph about `get application info`.

See also: the `get application info` FlightScript command, the FLIGHTCHECK® Workflow Reference Manual, the `do script` AppleScript command, the `object` AppleScript class.

1.43 get transfer function images

Not supported. In FLIGHTCHECK® 5, the various information panels in the Overview Window are built dynamically by the FlightPlan. FlightPlans can have many forms and shapes – most of which might not have an ‘Image Info’ panel.

See also: the `get application info` FlightScript command, the FLIGHTCHECK® Workflow Reference Manual, the `do script` AppleScript command, the `object` AppleScript class and its subclasses for examples of alternate ways to access this info.

1.44 load "<controlsfilepath>"

Supported, but not compatible. This statement is interpreted and executed by FLIGHTCHECK® 5, but the effect is different compared to FLIGHTCHECK® 4.5.

In FLIGHTCHECK® 5 Ground Controls (i.e. the currently active collection of Ground Control Sets) can be exported to a file, and imported back into FLIGHTCHECK® 5.

It is not possible to “load” a Ground Controls file into FLIGHTCHECK® 5 as it is in FLIGHTCHECK® 4.5. Loading a Ground Controls file creates an active ‘association’ between FLIGHTCHECK® 4.5 and that file, and changes in the Ground Controls are written back to the associated Ground Controls file.

In FLIGHTCHECK® 5 the file is merely read ‘into’ the program, and there is no further association with the Ground Controls file after importing it: changes in the Ground Controls will not affect the imported file.

In FLIGHTCHECK® 5, the above statement will execute an import of the file referred to by <controlsfilepath>. This import is additive: the imported file’s Ground Control Sets are appended to the list of already available ground control sets.

If the same statement is executed again, a second copy of the same ground controls will be imported (FLIGHTCHECK® 5 will rename the clashing ground control sets on-the-fly).

Contrasting to that: re-executing the same statement has no effect in FLIGHTCHECK® 4.5 – it merely sets up the association with the file.

To mimic the FLIGHTCHECK® 4.5 behavior more closely, one needs to delete the existing ground control sets prior to loading the ground controls file:

```
tell application "FLIGHTCHECK®.app"

    set theSetList to every ground control set
    repeat with theSet in theSetList
        delete theSet
    end repeat
    load -
    "Heavymetal:Users:kris:Desktop:GroundControls.gcs.xml"

end tell
```

Note that this script will delete the Default ground control set. But FLIGHTCHECK® 5 will automatically re-create a new Default set – so the Default set will not seem to be deleted. Instead it will be reset to the FlightPlan defaults.

Important: Make sure you have exported and safeguarded any Ground Control Sets you want to keep before running the above script.

1.45 open form "<formname>"

Not supported.

1.46 print form

Not supported.

1.47 print report

Supported. Prints a FlightCheck report. Example:

```
tell application "FLIGHTCHECK®.app"

    flightcheck "HeavyMetal:Users:kris:Desktop:Document1"
    print report

end tell
```

1.48 print results

Supported. Prints the results. Example:

```
tell application "FLIGHTCHECK®.app"

    flightcheck "HeavyMetal:Users:kris:Desktop:Document1"
    print results

end tell
```

1.49 remove all fonts folders

Supported. FLIGHTCHECK® 5 uses a slightly different font database mechanism than FLIGHTCHECK® 4.5: the fonts database is managed automatically instead of manually, and FLIGHTCHECK adds and removes fonts from it on an as-needed basis.

This command removes all font search paths.

See also `remove fonts folder`, `add fonts folder`.

1.50 remove all images folders

This command was not supported in FLIGHTCHECK® 4.5. It was introduced in FLIGHTCHECK® 5 to allow access to the image search folders.

This command removes all image search paths.

See also `remove images folder`, `add images folder`.

1.51 remove fonts folder "<fontfolderpath>"

Supported. FLIGHTCHECK® 5 uses a slightly different font database mechanism than FLIGHTCHECK® 4.5: the fonts database is managed automatically instead of manually, and FLIGHTCHECK adds and removes fonts from it on an as-needed basis.

FLIGHTCHECK® 5 also supports relative font folder paths, which are considered relative to the folder containing the main document – for example the folder “:Fonts” (with a leading semicolon) is considered to be a subfolder of the folder that also contains the main document.

This command removes the font folder path from the list of font search paths.

See also `remove all fonts folders`, `add fonts folder`.

1.52 remove images folder "<imagefolderpath>"

This command was not supported in FLIGHTCHECK® 4.5. It was introduced in FLIGHTCHECK® 5 to allow access to the image search folders.

FLIGHTCHECK® 5 also supports relative image folder paths, which are considered relative to the folder containing the main document – for example the folder “:Images” (with a leading semicolon) is considered to be a subfolder of the folder that also contains the main document.

This command removes the images folder path from the list of image search paths.

See also `remove all images folders`, `add images folder`.

1.53 save form **save form "<formname>"**

Not supported.

1.54 save report **save report "<pathtoreport>"**

Supported. Saves a FlightCheck report. <pathtoreport> is an optional path name of the file to save the report into. Example:

```
tell application "FLIGHTCHECK®.app"

    flightcheck "HeavyMetal:Users:kris:Desktop:Document1"
    save report -
    "HeavyMetal:Users:kris:Desktop:Document1.txt"

end tell
```

1.55 select binhex

Not supported.

1.56 select compress job

Supported. Causes the ‘Compress Job’ option on the Collect screen to be selected.

See also the `with compress` option of the `do collection` AppleScript command

1.57 select include collect report

Not supported. Instead, in FLIGHTCHECK® 5 one needs to use the file classes provided by the FlightPlan. In case of the Default FlightPlan, the relevant file class is called `CL_COLLECT_REPORT_CLASS_NAME`.

By manipulating the `collect` property of the `CL_DOCUMENT_CLASS_NAME` file class one can determine the default handling of document files for collection – i.e. whether to include or not. For more information: see the paragraph about the `collect fonts` FlightScript command.

The following example shows how to include the report using the file classes:

```
tell application "FLIGHTCHECK®.app"

    set the collect of -
        file class "CL_COLLECT_REPORT_CLASS_NAME" -
        to true

end tell
```

1.58 `select include dictionary`

Not supported.

1.59 `select include document`

Not supported. Instead, in FLIGHTCHECK® 5 one needs to use the file classes provided by the FlightPlan. In case of the Default FlightPlan, the relevant file class is called `CL_DOCUMENT_CLASS_NAME`.

By manipulating the `collect` property of the `CL_DOCUMENT_CLASS_NAME` file class one can determine the default handling of document files for collection – i.e. whether to include or not. For more information: see the paragraph about the `collect fonts FlightScript` command.

The following example shows how to exclude the document using the file classes:

```
tell application "FLIGHTCHECK®.app"
  set the collect of ↵
    file class "CL_DOCUMENT_CLASS_NAME" ↵
    to false
end tell
```

In addition to that, one can also set the `collect` property of individual files. The following example will mark all document files to be collected except if they have a `.jpg` or `.jpeg` file name extension:

```

tell application "FLIGHTCHECK@.app"

    set docList to every document object of the job
    repeat with doc in docList
        set docName to the filename of doc
        set docNameLen to the length of docName
        if docNameLen ≥ 5 then
            set nameEnd4 to ↵
                (characters (docNameLen - 3) thru↵
                docNameLen of docName) ↵
                as string
            set nameEnd5 to ↵
                (characters (docNameLen - 4) thru↵
                docNameLen of docName) ↵
                as string
            ignoring case
                if nameEnd4 = ".JPG" or ↵
                    nameEnd5 = ".JPEG" then
                    --
                    -- Override default collect setting
                    -- (which is determined via file
                    -- classes)
                    --
                    set the collect of doc to false
                end if
                --
                -- Do not change the collect property for
                -- the other file objects to 'true'
                -- otherwise their collect setting is
                -- not undefined any more and cannot be
                -- overwritten by the settings in the
                -- file classes.
                -- By _not_ doing 'set the collect of doc
                -- to true' we leave the collect of doc
                -- at an undefined value - which will
                -- then be overwritten by the file class
                -- collect setting when the
                -- collect function is called
                --
            end ignoring
        end if
    end repeat

    collect job

end tell

```

1.60 select include ground controls

Not supported.

1.61 select include req xts

Not supported.

1.62 select include xpress prefs

Not supported.

1.63 **select retain job name**

Not supported.

See also the `do collection AppleScript` command.

1.64 **select self extracting**

Not supported.

1.65 **select set "<setname>"**

Supported. Selects named Ground Control Set. Example:

```
tell application "FLIGHTCHECK®.app"
    select set "Default"
end tell
```

This example will select the default Ground Control Set if it exists. Nothing happens if the set does not exist.

1.66 **select retain job name**

Not supported.

See also the `do collection AppleScript` command.

1.67 **set color view to ...**

Supported. Example:

```
tell application "FLIGHTCHECK®.app"
    flightcheck "HeavyMetal:Users:kris:Desktop:Document1"
    set color view to Spot Colors
end tell
```

This example will preflight `Document1`, then select the *Spot Colors* filter on the overview window.

FLIGHTCHECK® 4.5 uses hard coded filter entries for the popup menu above the color list in the overview window. FLIGHTCHECK® 5 on the other hand uses all available color-related ground controls to dynamically build the contents of the popup menu. This means that if a FlightPlan has more or less color-related ground controls and rules, the filter will automatically show more or less options.

In order to emulate FLIGHTCHECK® 4.5's behavior, FLIGHTCHECK® 5's `Default.fpn.xml` FlightPlan contains a mapping table to map FLIGHTCHECK® 4.5's view names onto FLIGHTCHECK® 5 rule codes.

As a result, the visual naming of the selection in the filtering popup menu will not necessarily correspond to the FLIGHTCHECK® 4.5 naming, but the actual list of colors shown by FLIGHTCHECK® 5 should correspond to the list of colors shown in FLIGHTCHECK® 4.5.

See also: the section “List of Default.fpn.xml rule codes”.

The mapping table in Default.fpn.xml is:

FC 4.5 FlightScript	FC 5 Default.fpn.xml Rule Code
All Colors	*ALL
Flagged Colors	*FLAGGED
Used Colors	COLOR_USED
Spot Colors	COLOR_SPOT
Process Colors	COLOR_PROCESS
Trapped Colors	COLOR_TRAPPED
Unused Colors	COLOR_UNUSED
Mismatched Colors	COLOR_MISMATCHED
NonCMYK	COLOR_NON_CMYK_PAN
Unused Spot Colors	COLOR_UNUSED_SPOT
Unnamed Colors	COLOR_NOT_NAMED
NonDefault Trap	COLOR_NON_DEFAULT_TRAP
Bitmap Frames	COLOR_BITMAP_FRAME
Blends	COLOR_BLEND
Patterns	COLOR_PATTERN
Hairlines	COLOR_HAIRLINE
CMYK >	COLOR_SUM
CMYK <	COLOR_SUM_BELOW
Mismatched Spot/Process	COLOR_MISMATCHED_SPOT_PROCESS
Used Spot Colors	COLOR_USED_SPOT
Transparencies	TRANSPARENCIES

All Colors and Flagged Colors are special cases: they are not mapped onto rule codes, but instead their selection mechanism is hard coded into FLIGHTCHECK® 5.

1.68 set font view to ...

Supported. Example:

```
tell application "FLIGHTCHECK®.app"

    flightcheck "HeavyMetal:Users:kris:Desktop:Document1"
    set font view to NonAdobe

end tell
```

This example will preflight Document1, then select the *Non-Adobe Fonts* filter on the overview window.

FLIGHTCHECK® 4.5 uses hard coded filter entries for the popup menu above the font list in the overview window. FLIGHTCHECK® 5 on the other hand uses all available font-related ground controls to dynamically build the contents of the popup menu. This means that if a FlightPlan has more or less font-related ground controls and rules, the filter will automatically show more or less options.

In order to emulate FLIGHTCHECK® 4.5’s behavior, FLIGHTCHECK® 5’s Default.fpn.xml FlightPlan contains a mapping table which maps FLIGHTCHECK® 4.5’s view names onto FLIGHTCHECK® 5 rule codes.

As a result, the visual naming of the selection in the filtering popup menu will not necessarily correspond to the FLIGHTCHECK® 4.5 naming, but the actual list of fonts shown by FLIGHTCHECK® 5 should correspond to the list of images shown in FLIGHTCHECK® 4.5.

Some MarkzONE terminology: the ‘Font view’ in FLIGHTCHECK® 5 is not really a list of fonts. Instead, it is a list of MarkzONE `FONT_REQUEST` objects. If a corresponding font can be found, MarkzONE will link the font request to the physical, installed font. If no corresponding font is installed, the font request is marked as ‘missing’. Font requests are like ‘intents to display the font’ but there are no guarantees that the requested font will be available and installed.

See also: the section “List of `Default.fpn.xml` rule codes”.

The mapping table in `Default.fpn.xml` is:

<u>FC 4.5 FlightScript</u>	<u>FC 5 Default.fpn.xml Rule Code</u>
All Fonts	*ALL
Flagged Fonts	*FLAGGED
Active Fonts	FONT_REQUEST_ACTIVE
Database Fonts	FONT_REQUEST_DATABASE
Missing Fonts	FONT_REQUEST_MISSING
Missing PS Fonts	FONT_REQUEST_MISSING_PS_FONT
Menu Styled Fonts	FONT_REQUEST_MENU_STYLED
Inactive Fonts	FONT_REQUEST_INACTIVE
Screen Font	FONT_REQUEST_SCREEN_FONT
Printer Font	FONT_REQUEST_PRINTER_FONT
TrueType	FONT_REQUEST_TRUETYPE
City Font	FONT_REQUEST_CITY
Multiple Masters	FONT_REQUEST_MULTIPLE_MASTERS
NonAdobe	FONT_REQUEST_NON_ADOBE
Menu Styled	FONT_REQUEST_MENU_STYLED
Encoding	FONT_REQUEST_ENCODING
Unused Style Sheets	FONT_REQUEST_UNUSED
Corporate Fonts	FONT_REQUEST_CORPORATE

All Fonts and Flagged Fonts are special cases: they are not mapped onto rule codes, but instead their selection mechanism is hard coded into FLIGHTCHECK® 5.

1.69 set image view to ...

Supported. Example:

```
tell application "FLIGHTCHECK®.app"

    flightcheck "HeavyMetal:Users:kris:Desktop:Document1"
    set image view to RGB Images

end tell
```

This example will preflight `Document1`, then select the *RGB Images* filter on the overview window.

FLIGHTCHECK® 4.5 uses hard coded filter entries for the popup menu above the image list in the overview window. FLIGHTCHECK® 5 on the other hand uses all available image-related ground controls to dynamically build the contents of the popup menu. This means that if a FlightPlan has more or less image-related ground controls and rules, the filter will automatically show more or less options.

In order to emulate FLIGHTCHECK® 4.5’s behavior, FLIGHTCHECK® 5’s `Default.fpn.xml` FlightPlan contains a mapping table which maps FLIGHTCHECK® 4.5’s view names onto FLIGHTCHECK® 5 rule codes.

As a result, the visual naming of the selection in the filtering popup menu will not necessarily correspond to the FLIGHTCHECK® 4.5 naming, but the actual list of images shown by FLIGHTCHECK® 5 should correspond to the list of images shown in FLIGHTCHECK® 4.5.

The ‘Image view’ in FLIGHTCHECK® 5 is not really a list of images. Instead, it is a list of image containers. These containers then in turn refer to a placed or linked image. Hence, all the rule codes listed here are related to LOCATION or DOCUMENT_BOX MarkzONE objects, not to FILE or DOCUMENT objects.

See also: the section “List of Default.fpn.xml rule codes”.

The mapping table in Default.fpn.xml is:

<u>FC 4.5 FlightScript</u>	<u>FC 5 Default.fpn.xml Rule Code</u>
All Images	*ALL
Flagged Images	*FLAGGED
Missing Images	LOCATION_MISSING
Modified Images	LOCATION_MODIFIED
Unused Images	DOCUMENT_BOX_OFF_PAGE
Stored Images	LOCATION_STORED
Wrong Resolution	LOCATION_RESOLUTION_COMBINED
OPI Images	LOCATION_TYPE_OPI
RGB Images	LOCATION_MODE_RGB_COLOR
Compressed Images	LOCATION_COMPRESSED
Fill None	LOCATION_CLIPPING_PATH
Stylized Images	DOCUMENT_BOX_STYLES_CONTRAST
Bitmap Frames	BOX_BITMAP_FRAME
PICT	LOCATION_TYPE_PICT
TIFF	LOCATION_TYPE_TIFF
EPSF	LOCATION_TYPE_EPSF
EPS	LOCATION_TYPE_EPS
JPEG	LOCATION_TYPE_JPEG
CT	LOCATION_TYPE_CT
DCS	LOCATION_TYPE_DCS1
DCS2	LOCATION_TYPE_DCS2
GIF	LOCATION_TYPE_GIF
Other	LOCATION_TYPE_OTHER
PC	LOCATION_TYPE_PC
OPI	LOCATION_TYPE_OPI
Bitmap	LOCATION_MODE_BITMAP
Monotone	LOCATION_MODE_MONOTONE
Duotone	LOCATION_MODE_DUOTONE
Tritone	LOCATION_MODE_TRITONE
Quadtone	LOCATION_MODE_QUADTONE
Grayscale	LOCATION_MODE_GRAYSCALE
Indexed Color	LOCATION_MODE_INDEXED_COLOR
RGB Color	LOCATION_MODE_RGB_COLOR
CYMK Color	LOCATION_MODE_CMYK_COLOR
Lab Color	LOCATION_MODE_LAB_COLOR
Missing	LOCATION_MISSING
Modified	LOCATION_MODIFIED
Stored	LOCATION_STORED
NotIncluded	LOCATION_SUBLEVEL_NOT_INCLUDED
Nested	LOCATION_SUBLEVEL_NESTED
ASCII Data	LOCATION_ENCODING_ASCII
Binary Data	LOCATION_ENCODING_BINARY
LZW Encoding	LOCATION_ENCODING_LZW
JPEG Encoding	LOCATION_ENCODING_JPEG
Suppressed	LOCATION_SUPPRESSED
Off the Page	DOCUMENT_BOX_OFF_PAGE
Clipping Path	LOCATION_CLIPPING_PATH

Clipping	LOCATION_CLIPPING_PATH
Colored Fill	DOCUMENT_BOX_COLORED_FILL
Bitmap Frame	BOX_BITMAP_FRAME
Box Rotation	DOCUMENT_BOX_ROTATION
Box Skew	DOCUMENT_BOX_SKEW
Image Scale	LOCATION_CONTENTS_SCALE
Image Rotation	DOCUMENT_BOX_CONTENTS_ROTATION
Image Skew	DOCUMENT_BOX_CONTENTS_SKEW
HV Flip	DOCUMENT_BOX_CONTENTS_FLIP
Styles	DOCUMENT_BOX_STYLES_CONTRAST
PictureTrap	DOCUMENT_BOX_PICTURE_TRAP
Halftone Screen	DOCUMENT_BOX_HALFTONE_SCREEN
Transfer Function	LOCATION_TRANSFER_FUNCTION
Channels	LOCATION_CHANNELS
Layers	LOCATION_LAYERS
ICC Profile	LOCATION_ICC_PROFILE
PS Color Management	LOCATION_PS_COLORMANAGEMENT
Ink Density >	LOCATION_INK_DENSITY
Resolution	LOCATION_RESOLUTION_COMBINED
Bitmaps	LOCATION_MODE_BITMAP
Flatness	LOCATION_FLATNESS

All Images and Flagged Images are special cases: they are not mapped onto rule codes, but instead their selection mechanism is hard coded into FLIGHTCHECK® 5.

1.70 show general items

Supported. Closes all the individual results in the results window to show only the list of sections and problems per section.

1.71 show specific items

Supported. Opens all the results in the results window to show the individual items.