

Qi Image Quality

Certain features planned for future releases are not supported in the current version. For completeness this manual describes some such features marked with a yellow background.

| | |
|---|-----------|
| Bug fix and Change History | 4 |
| <i>v1.0.0 (build 202211171831)</i> | 4 |
| Installation | 5 |
| Installing the main components | 5 |
| OECF installation and file system access | 5 |
| Installing the optional Excel Reporting app | 5 |
| System verification after installation | 6 |
| Overview | 8 |
| File organization inside the QiQ work folder | 8 |
| Example of an automated analysis | 9 |
| The user interface | 11 |
| <i>Context menus</i> | 11 |
| <i>Enabling or disabling projects</i> | 12 |
| <i>Project configuration</i> | 13 |
| Using excel reports | 15 |
| Raw and templated Excel reports | 16 |
| Creating report templates | 16 |
| Configuring projects with report templates..... | 18 |
| The compound reporting system | 20 |
| An end-to-end example of compound reports | 21 |
| Compound reports generally | 26 |
| Details of compound reporting | 27 |
| <i>Data selection and mapping (GAN)</i> | 27 |
| <i>Examples of selection criteria and mappings</i> | 28 |
| Generating the compound reports..... | 28 |
| <i>The list of compound reports (in the project report tab)</i> | 29 |
| <i>The detail view for a sample ID (right side of the project report tab)</i> | 30 |
| Test charts—digital charts, Chart Scripts and Fiducial Marks | 31 |
| What are chart scripts | 31 |
| The digital (and printed) chart name, version and identifier | 32 |
| Name and version of the chart script..... | 33 |

| | |
|---|-----------|
| Installation and use of chart scripts | 33 |
| Example of the life cycle of a chart and chart scripts | 34 |
| Fiducial marks | 36 |
| Working with distorted (or missing) fiducial marks | 37 |
| <i>Manually indicating the fiducial mark positions on a scanned image</i> | 37 |
| <i>Teaching QiQ to recognize distorted fiducial marks</i> | 39 |
| Chart example | 40 |
| Color swatches: QQChartFlatField | 44 |
| Lines: QQChartLine | 44 |
| Color-color registration: QQChartColorRegH and QQChartColorRegV | 44 |
| Scripts | 45 |
| Chart scripts | 46 |
| Analysis scripts | 46 |
| Measurements: Metric modules | 47 |
| Using metric modules, parameters and outputs in the analysis scripts | 47 |
| Image requirements | 48 |
| Generic metric module scripting support | 48 |
| Mottle metric module | 50 |
| Image requirements | 50 |
| Scripting support—parameters and outputs | 50 |
| Analysis method | 51 |
| Conformance to ISO 24790 | 54 |
| Extended analysis options | 55 |
| Influence of average density on the visual perception of mottle | 55 |
| Graininess metric module | 56 |
| Image requirements | 56 |
| Scripting support—parameters and outputs | 56 |
| Analysis method | 57 |
| Conformance to ISO 24790 | 58 |
| VBS (Visual Banding and Streaking) metric module | 59 |
| Image requirements | 59 |
| Scripting support—parameters and outputs | 59 |
| Analysis method | 61 |
| Practical use of the VBS measurement | 61 |
| Conformance to ISO 24790 | 61 |
| Lines metric module | 62 |

| | |
|---|-----------|
| Image requirements | 62 |
| Scripting support—parameters and outputs | 62 |
| Analysis method..... | 64 |
| Specifying the nominal line orientation (horizontal or vertical)..... | 65 |
| Coverage metric module | 66 |
| Image requirements | 66 |
| Scripting support—parameters and outputs | 66 |
| Analysis method..... | 67 |
| Voids metric module | 68 |
| Image requirements | 68 |
| Scripting support—parameters and outputs | 68 |
| Analysis method when plso24790Conformance = true..... | 70 |
| Analysis method when pAlgorithm = “VisualVoidsA1” | 71 |
| CIEColor metric module | 72 |
| Image requirements | 72 |
| Scripting support—parameters and outputs | 72 |
| Analysis method..... | 73 |
| Analysis jobs and data flow | 74 |
| Input to the analysis—analysis jobs..... | 74 |
| Intermediate analysis output—IAOB..... | 74 |
| Raw analysis output—JRAF..... | 75 |
| Analysis data base (QCD) | 75 |
| Intermediate report files—JRAF and compound JRAF..... | 75 |
| Data Flow | 75 |
| The QCD Database | 77 |
| Description of the data model entities | 78 |
| Troubleshooting | 81 |
| Program crashes | 81 |

Bug Fix and Change History

v1.0.0 (build 202211171831)

- Initial public release.
-

Installation

Installing the main components

There are 3 apps to consider:

- QiQ is the main app to perform analysis of TIFF image files.
- QiQ Scanner can be used to scan prints with a flatbed scanner.
- The optional QiQReporter app is used to generate Excel reports.

The QiQ app is self-contained and can be placed anywhere on the Mac's file system. It is recommended to follow convention and place QiQ either in the system Applications folder (where it can be accessed by all users), or in the user's Applications folder. The user's Applications folder should be located in the user's home folder; it can be created if not already there.

There should not be more than one copy of QiQ on the computer.

OECF installation and file system access

Scanner OECF installation. The OECF (Opto-Electric Conversion Function) is specific to each scanner device. Its purpose is to be able to convert the RGB scanner values to calibrated reflectance values. The OECF is created by scanning a specialized test chart (e.g. the ISO 24970 conformance chart), measuring densities on that chart with a densitometer, and then computing the OECF based on that data. A future version of QiQ will partly automate this process, but at this time the OECF must be created manually and installed manually. To install an OECF file select File > Locations > Show Scanner OECFs. The folder already exists and contains one demo OECF file.

Installing the optional Excel Reporting app

To generate Excel report you must first install a separate application *QiQReporter*. This application must be copied to the Applications folder on the Mac. This application should not

be run directly by the user, but will be run in the background by the main program. This app can be downloaded from QiQ > Preferences > Reports.

System verification after installation

After installation of an update it may be desirable to verify that measurements are consistent with previous installations. This is called system verification testing and comes in two flavors:

- System testing: Test data provided with the QiQ system. This allows a user to perform some of the same tests that are performed during quality assurance before a new version is released.
- Custom testing: Test data compiled by the user, in order to further verify calculations of special interest to the user.

The process is similar whether for system or custom verification.

- Initial setup
 - Download the test images. This can be done from [QiQ > Preferences > Verification](#).
 - The reference data for a project is defined by the project configuration file. This data is not editable from the user interface, and must be edited directly in the project configuration file. The QiQ Verification project's reference data are already included with the system installation.
- Verification
 - Perform analysis of all the test images.
 - Select [QiQ > Verification](#) to open the verification view.
 - From the project selector select the [QiQ Verification](#) project. The analyses performed for the project are listed (also showing the QiQ software version that produced the data).
 - Select one or more analyses and then click [Verify](#). The analysis data will be compared to the reference data and a summary will be reported.

The expected output from running the system verification is similar to this:

```
Chart: QiQ MacroUniformity A.K40_v1.0.0
Comparing to reference: <C3F5D5F6-FE0A-473E-B9AF-AFC0B9D77504>.
Reference analysis script: <Verify VBS_v1.0>
Compared analysis script : <Verify VBS_v1.0>
✅ CIEColor: 14 measurements; 196 reference values; all matching; max relative difference:0.0000% ☐;
✅ VBS: 1 measurements; 11 reference values; all matching; max relative difference:0.0000% ☐;
```

No issues were found.

Chart: QiQ Voids and Dots A_v1.0.0

Comparing to reference: <6BED0B31-3D92-4493-9936-CCF30361DC7A>.

Reference analysis script: <Verify Voids Dots_v1.0>

Compared analysis script : <Verify Voids Dots_v1.0>

✔ Dots: 2 measurements; 22 reference values; all matching; max relative difference:0.0000% ☐;

✔ Voids: 2 measurements; 30 reference values; all matching; max relative difference:0.0000% ☐;

No issues were found.

Chart: QiQ Lines A_v1.0.0

Comparing to reference: <5139FCC5-0559-49F8-9434-FE9531DB791E>.

Reference analysis script: <Verify Lines ColorReg_v1.0>

Compared analysis script : <Verify Lines ColorReg_v1.0>

✔ Lines: 400 measurements; 6400 reference values; all matching; max relative difference:0.0000% ☐;

No issues were found.

Chart: QiQ MicroUniformity A_v1.0.0

Comparing to reference: <E041EC9A-E2AD-4A9F-98D2-0D485437C72B>.

Reference analysis script: <Verify CIE Mottle Graininess_v1.0>

Compared analysis script : <Verify CIE Mottle Graininess_v1.0>

✔ CIEColor: 40 measurements; 560 reference values; all matching; max relative difference:0.0000% ☐;

✔ Graininess: 40 measurements; 400 reference values; all matching; max relative difference:0.0000% ☐;

✔ Mottle: 40 measurements; 400 reference values; all matching; max relative difference:0.0000% ☐;

No issues were found.

Overview

File organization inside the QiQ work folder

For the purpose of backing up, or moving or duplicating QiQ to another Mac, you should know the location of the sandbox. To open the sandbox folder, select [File > Show Locations > Sandbox](#). This will show the sandbox folder, named QiQ, which we also call the *QiQ sandbox*, or just the *sandbox*. The sandbox contains files that you normally do not interact with directly, for example database files and raw measurement results.

Normally you just interact with files in the QiQ work folder. When you first launch QiQ it will ask you for a location for the work folder, and create the folder. Normally the work folder is named QiQuality and is located directly in your home folder.

The work is organized like this:

Table: Organization of the QiQ work folder.

| | |
|---|--|
| QiQuality | Work folder |
| QiQuality / user | All user files |
| QiQuality / user / configuration | Files typically only used by a QiQ system administrator. Contains scanner OECF, user scripts, user report templates, and more. |
| QiQuality / user / projects | Should contain one or more user-created project folders, which in turn contain scanned images, reports, and more. |
| QiQuality / user / projects / completed | Scan files that have been fully analyzed can be placed here. |
| QiQuality / user / projects / images | Scan files to be analyzed should be placed here. |
| QiQuality / user / projects / reports | Contains final report files (Excel) for this project. |
| QiQuality / system | Contains files needed by the QiQ system, and should not be modified by the user. For example scripts and data for verification of the system, and for analysis of system provided test charts. |

The sandbox is organized like this:

Table: Organization of the sandbox folder.

| | |
|---|--|
| QiQ | The sandbox folder |
| QiQ / systemdata / database | Database files containing measurements results. |
| QiQ / systemdata / qiq_work_template | Folder that is a template for new user work folders. |
| QiQ / tmp | During analysis contains temporary data. |
| QiQ / userdata | <p>Various data generated during analyses. JRAF subdirectories contain raw analysis results which are used when generating final reports.</p> <p>For each work folder, there is a corresponding path of subdirectories to contain only data related to that work folder.</p> |

Example of an automated analysis

This tutorial starts with a script that defines a test chart; uses that to generate a PDF file that can be printed; scans the print with the QiQ Scanner app, and then performs a fully automated analysis of all the test elements, leading to a spreadsheet-style report.

1. Launch QiQ.
2. Select [Projects > View Projects](#) to make sure the projects window is open (see figure).
3. In the projects window sidebar, turn on the [Master Switch](#) power switch, then turn on the power switch next to [QiQ Tutorial Project](#). Select [QiQ Tutorial Project](#), so it highlights and shows project details on the right side of the window. The details contain a list of images, and a view of the configuration of the project, but since we have not yet scanned any prints, the list of images is empty. However, the QiQ app is now monitoring the project and will start analysis as soon as scanned images arrive. See the figure below for comparison.
4. Select [Chart > View Charts](#). The Chart window opens.
5. From the pop menu select the chart named [QiQ MicroUniformity A_1.0.0](#). A script that defines this chart has now been loaded, and a list of elements of the chart is shown.
6. Select [Make PDF](#). QiQ will create a PDF as defined by the script, and the PDF will be opened in the default PDF reader app (normally Apple's Preview app). The chart has 40 colored swatches, five fiducial marks and a QR code that holds the name and version of the chart.
7. From the Preview app, save the PDF file, print it, then close the Preview app.

8. Put the print on the scanner platen. Launch the QiQ Scanner app, and make the following selections:
 1. Make sure the check mark is selected to [Use QR Code for Chart Identifier](#). This way you don't have to type the chart name and version.
 2. Under [Sample Identifier](#) type "my first test" into the first component, and leave the remaining 3 components unchanged.
 3. Select to scan at 300 [DPI](#) resolution.
 4. For scan destination select [QiQ Tutorial Project](#).
 5. Select [Scan](#). The sample will be scanned to the folder [QiQ > user > projects > QiQ Tutorial Project](#).
9. Return to the QiQ app and make sure the projects window is still open, with the tutorial project selected. When the scanner has completed scanning, you should see the image file appear in the project image list, and shortly after a green check mark next to the image will indicate that analysis of the image has completed.
10. Control-click on [QiQ Tutorial Project](#) to view the context menu and select [Show Reports](#). This opens the project's reports folder in the Finder, and if you have installed the Excel components, the Excel report will be located here.

The report contains both input parameters and calculated results from the analysis. It also contains information on the CMYK coverages of each element, and several other attributes of the scanned image.

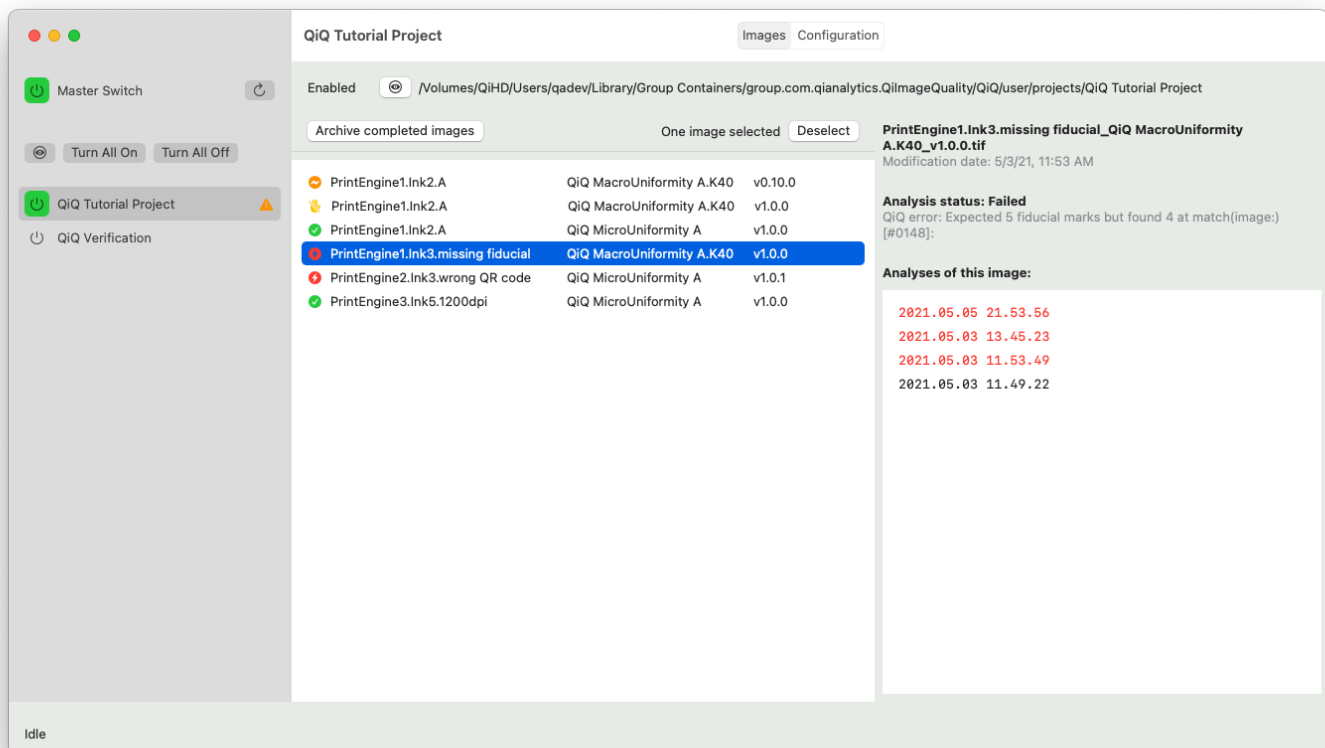


Figure: The projects window.

The User Interface

Context menus

QiQ uses *context menus* throughout its user interface. This means that commands are available by control-clicking (or right-mouse clicking) on an object on the user interface. The figure below shows an example: Control-clicking on an image entry brings forward a context menu that allows you to operate on the image.

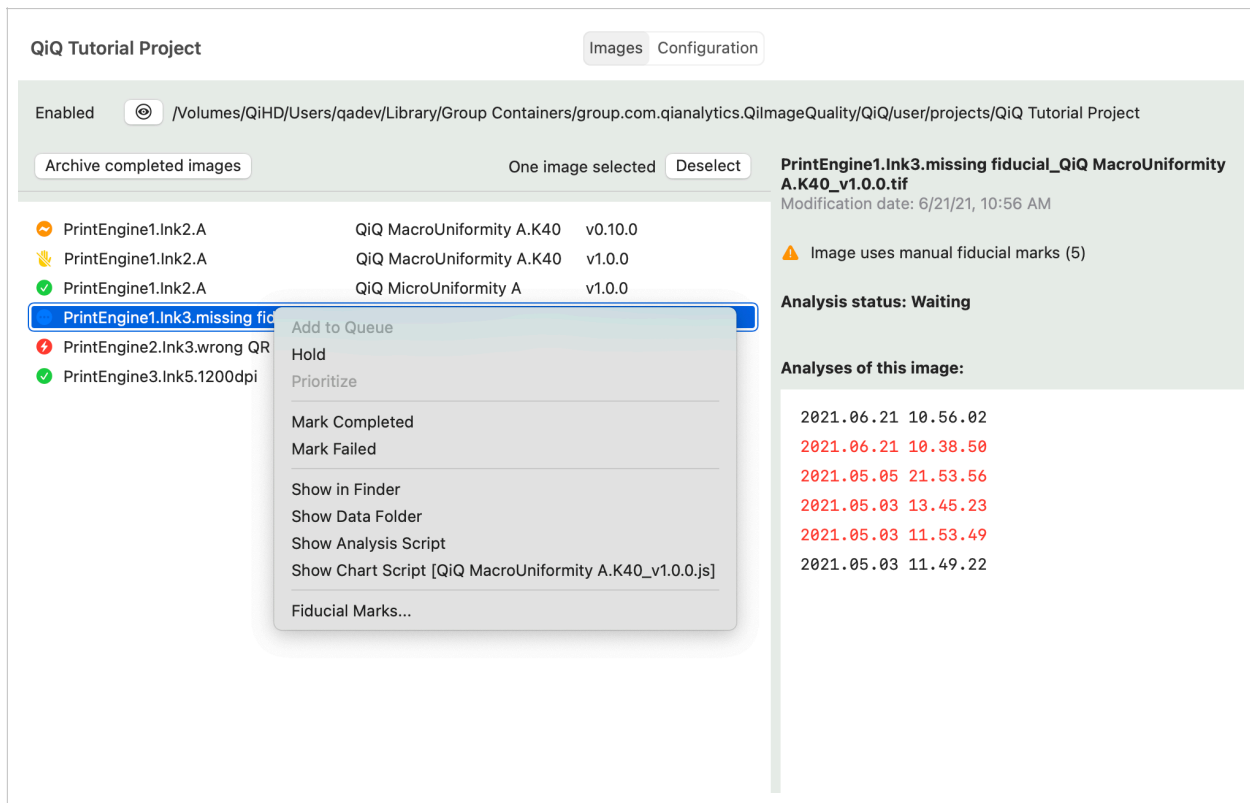


Figure: Context menu of an image entry. The menu allows you to perform operations for the image (processing queue, file system, and fiducial mark operations)

As you explore the user interface, remember the context menus since many essential commands are available there.

Enabling or disabling projects

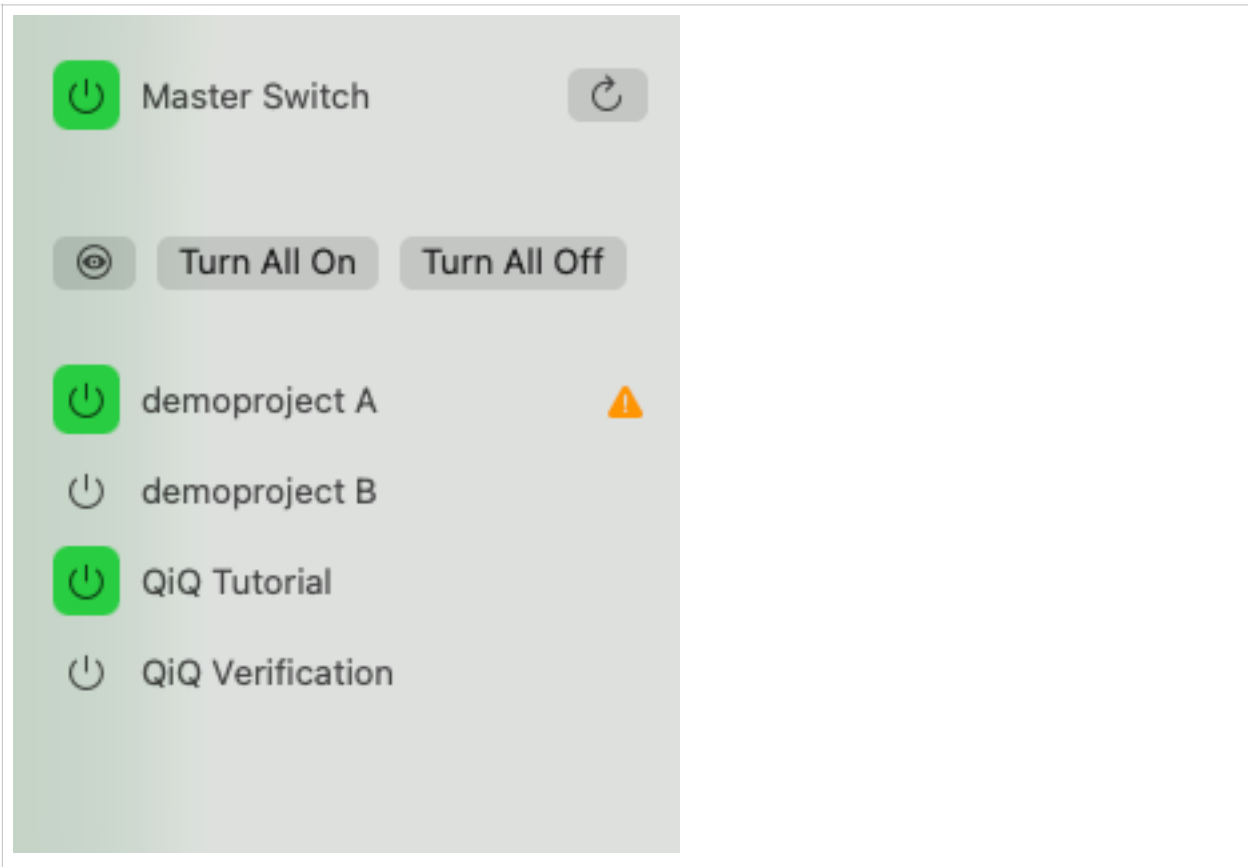


Figure: Enabling and disabling projects from the Projects view.

QiQ monitors images inside the projects' images folder, and will, in some cases, automatically start processing the images. To have project images be processed:

- First, the project must be enabled, that is, the on/off button next to the project name must be turned on. In the figure, two projects are enabled: "demoproject A" and "QiQ Tutorial".
- Secondly, the [Master Switch](#) must be turned on.

Project configuration

QiQ Tutorial

Images Configuration

Disabled /Volumes/QiHD/Users/qadev/Library/Group Containers/group.com.qianalytics.QiImageQuality/QiQ/user/projects/QiQ Tutorial

OECF: Linear OECF QR Code: Require match if present

Select Charts for Project Add Chart

| | | | |
|----------------------------------|---|--------------|---|
| * QiQ Lines A_v1.0 | → | Generic Auto | Template: template_QiQ Lines A.xlsx |
| * QiQ MacroUniformity A.K40_v1.0 | → | Generic Auto | Template: template_QiQ MacroUniformity... |
| * QiQ MicroUniformity A_v1.0 | → | Generic Auto | Template: template_QiQ MicroUniformity A... |

☒ Compound report

Report name: Tutorial

Template for compound report: compoundtemplate_QiQ Micro+Macro A_v1.0.xlsx

Number of analyses used by the compound report: 2. Add generic analysis

| | |
|-------|-----------------------------|
| MICRO | QiQ MicroUniformity A.* |
| MACRO | QiQ MacroUniformity A.K40.* |

Select to exclude any modules from the analysis

☐ QQCIEColor
 ☐ QQCoverage
 ☐ QQDots
 ☐ QQGraininess
 ☐ QQLines
 ☐ QQMottle
 ☐ QQStatistics
 ☐ QQVBS
 ☐ QQVoids
 ☐ QQWaveletFilter

Figure: The project configuration view.

You can configure a project from the project configuration view.

Project folder. The path to the project folder is shown at the top, and a button allows you easily to open the folder.

Protection from accidental changes. When the project is first viewed, it is *locked*, meaning that the configuration cannot be changed. Tapping on the lock button will toggle between locked and unlocked. The lock can be password protected: This must be done by editing the project_config.json file, where the password is stored as clear text (so anybody who wants, can easily find the password and unlock the project). Furthermore, while the project is enabled, the configuration cannot be changed.

OECF. The OECF must match the scanner that is the source of the images being analyzed. If multiple scanners are used, it is recommended that separate projects be used, each with the corresponding OECF (so that the operator does not have to remember to switch OECF).

QR Code. This selects the behavior with respect to QR codes on the image, informing QiQ about the chart name and version. The recommended option is [Require match if present](#), meaning that if a QR code is present in the image, specifying a chart name and version, then it must match the chart name and version given by the image file name—otherwise an error is reported.

Select Charts for Project. The popup menu allows you to select charts and then add them to the project. The project will only support analysis for the charts added here (the patch version is ignored—see the section on test charts for more on this subject). For each chart, you must also select a corresponding analysis script, and optionally select a report template. In many cases, the system analysis script [Generic Auto](#) can be used.

Compound Report. You can optionally add a compound report to the project. With a compound report you can get a single report that contains data from multiple image analyses. Here you must give a name for the report (which will be part of the file name of the compound report) and select a template. Then you must define one or more *generic analyses*—see the section on the compound reporting system for more information on how to use compound reports. In the case shown in the figure, the generic analyses are setup as follows:

- The report template must include the *generic analysis name* (GAN) “MICRO,” and rows labelled MICRO will receive data from charts that match the name “QiQ MicroUniformity A” (followed by any additional text).
- The report template must also include the GAN “MACRO,” and rows labelled MACRO will receive data from charts that match the name “QiQ MacroUniformity A.K40” (followed by any additional text).

Excluded modules. The analysis script specifies which measurement modules will be used. However, it can sometimes be desirable to exclude some of the measurements (e.g. to speed up the measurement by excluding unnecessary measurements).

Using Excel Reports

When you run an analysis of an image, all the results are stored in an internal database. Afterwards, the results can be pulled from the database and used to create reports in various manners that suit the end purpose. This can happen automatically after each analysis is completed, or you can access previous analyses in the database, and generate new reports at a later time. This chapter describes how to setup and work with Excel reports.

If wish to use other means than Excel—for example a custom external database—see [the section](#) that which describes the raw analysis output file. The raw analysis output is in a JSON format, well suited to import into other systems.

Before proceeding, make sure the optional QiQ Excel components have been installed (see the section on QiQ installation).



Raw and templated Excel reports

A report template is an Excel document that has been prepared in advance to customize which parameters and results will be reported, and to define the layout of that data. The template may also contain summary calculations, graphs of the data, etc.

A report can be generated either with a full template (*templated report*) or with a blank (raw) template (*raw report*). To use a blank template, select [template_QiQ_blank.xlsx](#) when configuring the project. In both cases, the report will have sheets for each analysis module that contributed to the analysis, for example one sheet for Mottle, another sheet for Graininess, and so forth. In addition, a sheet ("AnalysisMetaData") contains information about the sample that was analyzed, the date, the chart, the scripts, etc.

On each module sheet, the data are organized by columns:

- Column A, [GAN](#): This is reserved for compound reports (described later).
- Column B, [SampleID](#): The sample ID of the image analyzed.
- Column C, [Group](#): This indicates that optional group identifier, which is useful if multiple otherwise identical measurements have been performed on the same chart element (e.g. with differing parameters). The group identifier can be assigned in the analysis script.
- Column D, [Chart Elements](#): This column holds the chart element names, and determines the order in which the results are listed.
- Column E, [Content](#): This is a description of the content of the element, e.g. color percentages.
- The subsequent columns are input parameters and results from the measurements.

An example of a report sheet, for the Graininess module, is given in the figure. Columns F through K are results, while column L and higher are input parameters.

Creating report templates

The easiest way to create a report template is to start with a blank report.

1. Configure the project to use the blank system template. Then perform an analysis.
2. Open the report (located in the projects/<projectName>/reports folder).
3. Save the report to the QiQ/user/configuration/reporttemplates folder, naming it appropriately.
4. Using the data already in the report, modify the layout and formatting, and add any calculations or graphs that you would like. There is a summary sheet which is intended for

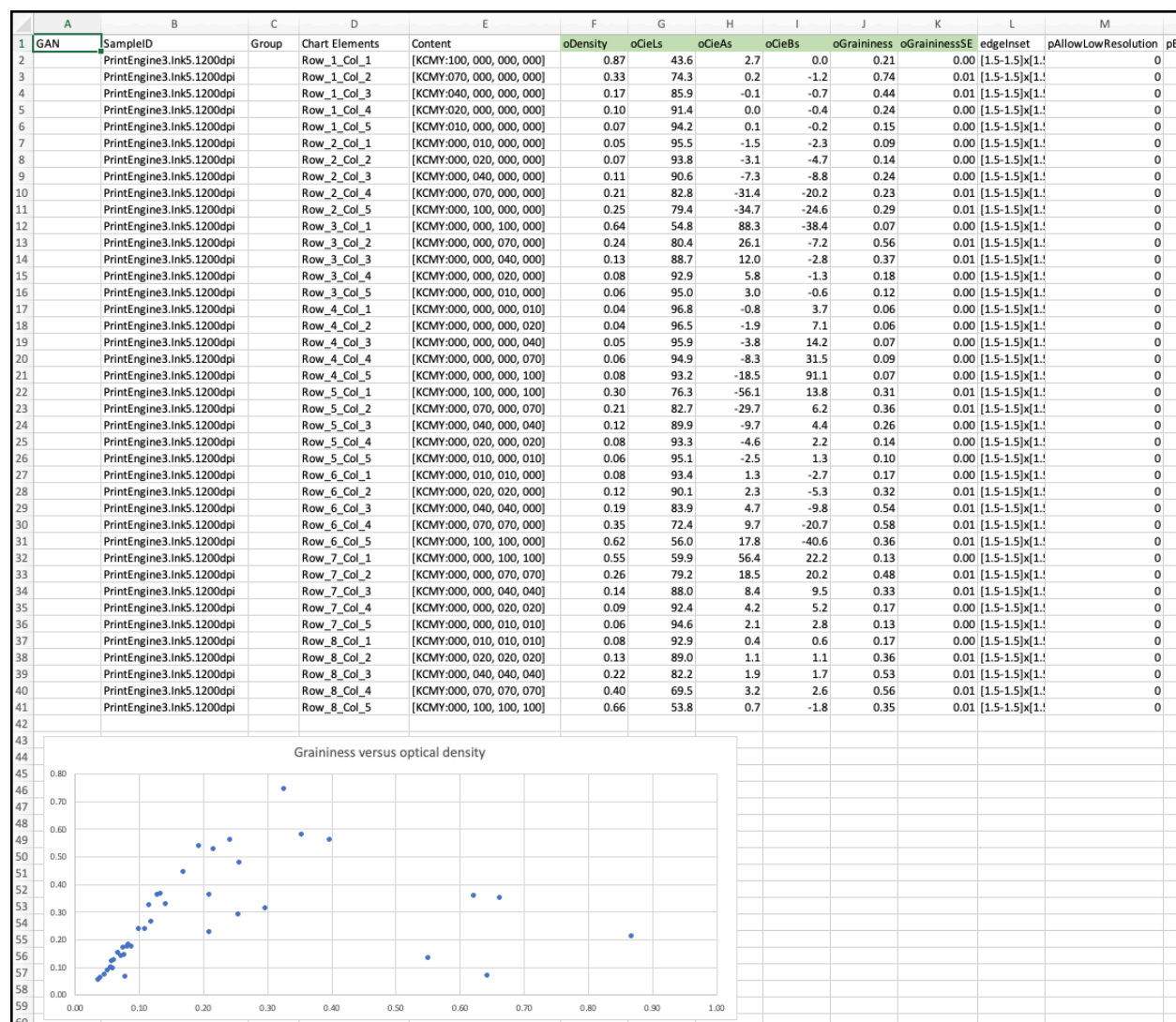


Figure: Example of report sheet for the Graininess module. Columns F through K are results, while column L and higher are input parameters.

you to create summary information pulled from the various other sheets, but you can also create additional sheets as needed. However, there are certain limitations to what you are allowed to do:

1. Data must be organized by the module sheets. Sheets can be reordered, but should not be renamed or deleted, and data should not be mixed between the different module sheets.
2. You can add non-module sheets where you can perform any calculations or aggregations of data from the module sheets.
3. You should not delete row in the module sheets, but you can re-order the rows as long as they are contiguous, without any blank rows.

4. Columns A through E must not be re-ordered or deleted. The subsequent columns (results and parameters) can be re-ordered, or deleted, but the columns should be contiguous without blank columns. Think twice before you delete a column that contains parameters, since the parameters serve to document the details of the measurement.
5. You can freely change the formatting such as column widths, colors, numeric precision, etc.
6. When you are done with the modifications of the template, go through each of the module sheets and delete all values except for the headers (row 1) and column D. The element names in column D must remain in order to specify the order of the results. Then save the template.

| | A | B | C | D | E | F | G | H | I | J | K | L | M |
|----|-----|----------|-------|----------------|---------|----------|--------|--------|--------|-------------|---------------|-----------|---------------------|
| 1 | GAN | SampleID | Group | Chart Elements | Content | oDensity | oCieLs | oCieAs | oCieBs | oGraininess | oGraininessSE | edgeInset | pAllowLowResolution |
| 2 | | | | Row_1_Col_1 | | | | | | | | | |
| 3 | | | | Row_1_Col_2 | | | | | | | | | |
| 4 | | | | Row_1_Col_3 | | | | | | | | | |
| 5 | | | | Row_1_Col_4 | | | | | | | | | |
| 6 | | | | Row_1_Col_5 | | | | | | | | | |
| 7 | | | | Row_2_Col_1 | | | | | | | | | |
| 8 | | | | Row_2_Col_2 | | | | | | | | | |
| 9 | | | | Row_2_Col_3 | | | | | | | | | |
| 10 | | | | Row_2_Col_4 | | | | | | | | | |
| 11 | | | | Row_2_Col_5 | | | | | | | | | |
| 12 | | | | Row_3_Col_1 | | | | | | | | | |
| 13 | | | | Row_3_Col_2 | | | | | | | | | |
| 14 | | | | Row_3_Col_3 | | | | | | | | | |
| 15 | | | | Row_3_Col_4 | | | | | | | | | |
| 16 | | | | Row_3_Col_5 | | | | | | | | | |
| 17 | | | | Row_4_Col_1 | | | | | | | | | |
| 18 | | | | Row_4_Col_2 | | | | | | | | | |
| 19 | | | | Row_4_Col_3 | | | | | | | | | |
| 20 | | | | Row_4_Col_4 | | | | | | | | | |

Figure: Module sheet in an Excel report template.

The module sheets of the finished template should look similar to that in the figure below, with only the first row and column D non-empty.

The next figure shows an example of a templated report where a column of reference CIE L* values were added in order to produce a graph of measured versus expected values. **Notice the blank column (P):** this separates the results and parameter columns from the comparison data. You should always leave one blank column after the results and parameters columns.

Configuring projects with report templates

The project configuration controls how Excel templates are used. In the projects window select the project, then select the [Configuration](#) tab. In the chart-analysis-template section of the view (see figure) you can configure which report template to use for each chart that the project uses:

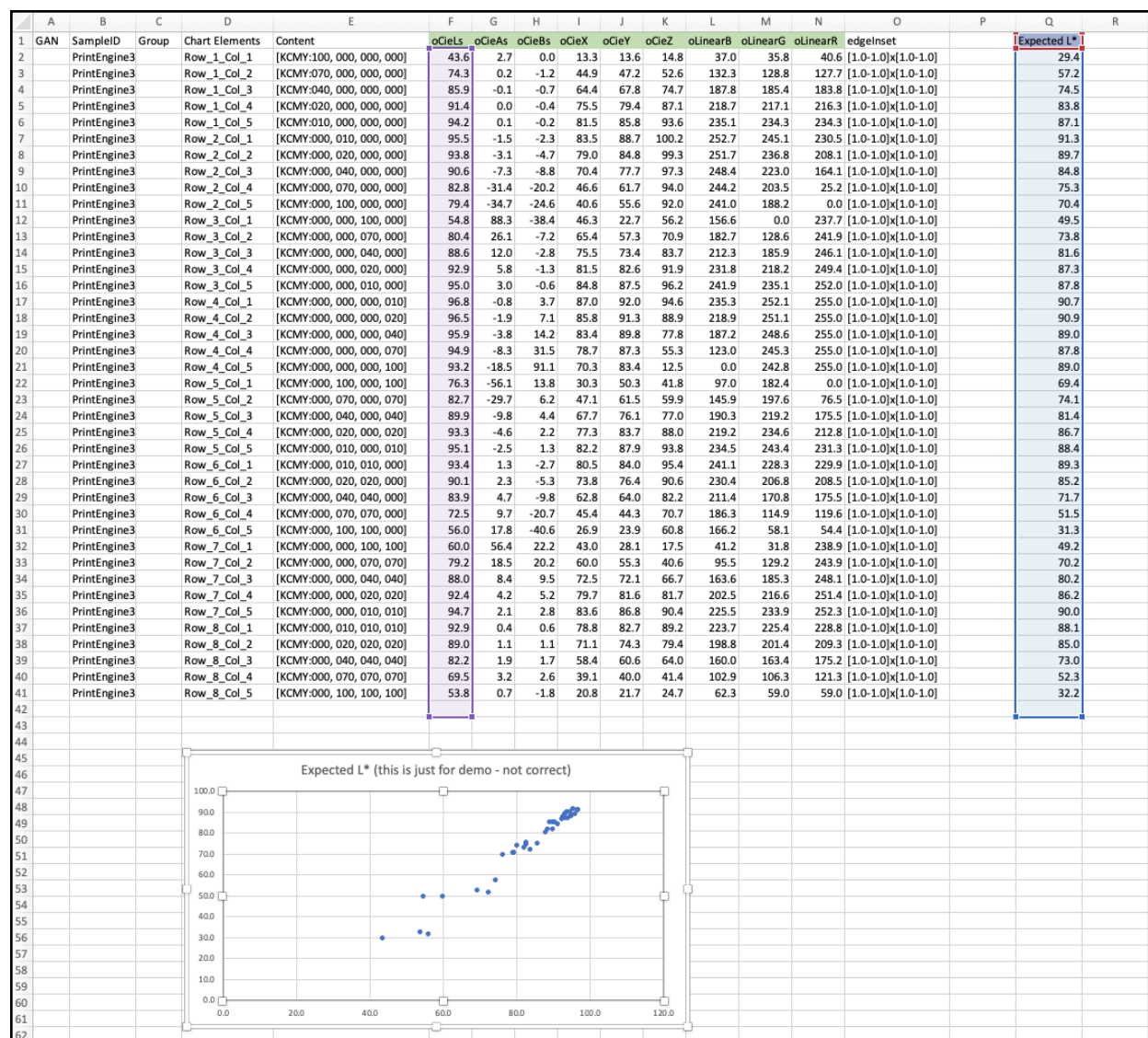


Figure: Report created with a template that has reference data in column Q.

- (No report): With this selection no Excel report is generated. The analysis results are written to a JRAF file (as always) and are then imported into the data base, where it may be used later to generate a plain or compound report (described later).
- A predefined template, either installed by the user, or provided by the QiQ system.
- template_QiQ_blank.xlsx: As described above, this will create a raw Excel report, which can be useful when setting up a new measurement and creating a new report template.

The screenshot shows a web-based configuration interface. At the top, it says 'Enabled' with a status icon and a path: '/Volumes/QiHD/Users/qadev/Library/Group Containers/group.com.qianalytics.QiImageQuality/QiQ/user/projects/QiQ Tutorial Project'. Below this are three dropdown menus: 'OECF: Linear OECF', 'Process: * QiQ System', and 'QR Code: Require match if present'. Underneath is an 'Add Chart' button. The main section contains a table with three rows of chart configurations:

| | | | | |
|----------------------------------|---|----------------|-----------|---------------------------------|
| * QiQ Lines A_v1.0 | → | * Generic Auto | Template: | template_QiQ Lines A.xlsx |
| * QiQ MacroUniformity A.K40_v1.0 | → | * Generic Auto | Template: | template_QiQ MacroUniformity... |
| * QiQ MicroUniformity A_v1.0 | → | * Generic Auto | Template: | template_QiQ_blank.xlsx |

Figure: The chart-analysis-template section of the project configuration view.

The Compound Reporting System

There are situations where a full characterization requires analysis of multiple print samples, and it is desirable to present all the measurement results within a single Excel report. Such a report, that contains data from multiple sample analyses, we call a *compound report*.

To explain compound reports we first go through an end-to-end example of using compound reports in one specific case. After that, we will have a more general discussion, including an explanation of why QiQ compound reports work the way they do.

An end-to-end example of compound reports

The QiQ system charts QiQ MicroUniformity A and QiQ MacroUniformity A are used to evaluate color uniformity, on small scales and large scale respectively. When you analyze prints of the charts, QiQ generates one Excel report for each chart. However, we'd like to have a single spreadsheet that includes both forms of uniformity measurements. Of course, we can create that manually by copying data from one spreadsheet document into the other, but QiQ compound reports can do it automatically. There are 3 steps to settings this up.

The first step is to create an Excel template for the compound report, and that is done by combining the basic templates for each chart. The figure below shows the key module sheets of the two basic templates.

| | Tempalte for QiQ MicroUniformity A | Tempalte for QiQ MacroUniformity A |
|-----------------------------|---------------------------------------|---------------------------------------|
| CIEColor module sheet | | |
| Mottle module sheet | | |

VBS
module
sheet

| | A | B | C | D | E | F | G | H |
|----|-----|----------|-------|----------------|---------|----------------|------------------|------------------|
| 1 | GAN | SampleID | Group | Chart Elements | Content | oVbsHorizontal | oVbsHorizontalHS | oVbsHorizontalHS |
| 2 | | | | Main | | | | |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |
| 6 | | | | | | | | |
| 7 | | | | | | | | |
| 8 | | | | | | | | |
| 9 | | | | | | | | |
| 10 | | | | | | | | |
| 11 | | | | | | | | |
| 12 | | | | | | | | |
| 13 | | | | | | | | |
| 14 | | | | | | | | |

◀ ▶
M.AnalysisMetaData
M.CIEColor
M.VBS
+

We will just look at three measurements performed on the charts: CIEColor (color measurement performed on both charts), Mottle (performed only on the micro-uniformity chart), and VBS (visual bands and streaks, performed only on the macro-uniformity chart).

We want the compound report to show the data in this way:

- One sheet, M.CIEColor, which contains color measurements from both charts.
- One sheet, M.Mottle, which contains the mottle measurements from the micro-uniformity chart.
- One sheet, M.VBS, which contains the VBS measurements from the macro-uniformity chart.

| | | Compound template | | | | | | | |
|-----------------------|-------|-------------------|-------|----------------|---------|--------|--------|--------|--------|
| CIEColor module sheet | | A | B | C | D | E | F | G | H |
| 1 | GAN | SampleID | Group | Chart Elements | Content | oCieLs | oCieAs | oCieBs | oCieCs |
| 35 | MICRO | | | Row_7_Col_4 | | | | | |
| 36 | MICRO | | | Row_7_Col_5 | | | | | |
| 37 | MICRO | | | Row_8_Col_1 | | | | | |
| 38 | MICRO | | | Row_8_Col_2 | | | | | |
| 39 | MICRO | | | Row_8_Col_3 | | | | | |
| 40 | MICRO | | | Row_8_Col_4 | | | | | |
| 41 | MICRO | | | Row_8_Col_5 | | | | | |
| 42 | MACRO | | | K000 | | | | | |
| 43 | MACRO | | | K005 | | | | | |
| 44 | MACRO | | | K010 | | | | | |
| 45 | MACRO | | | K020 | | | | | |
| 46 | MACRO | | | K030 | | | | | |
| 47 | MACRO | | | K040 | | | | | |

| | | | | | | | | | |
|---------------------|----|-------|----------|-------|----------------|---------|----------------|------------------|----------|
| Mottle module sheet | | A | B | C | D | E | F | G | H |
| | 1 | GAN | SampleID | Group | Chart Elements | Content | oDensity | oMottle | oMottleS |
| | 2 | MICRO | | | Row_1_Col_1 | | | | |
| | 3 | MICRO | | | Row_1_Col_2 | | | | |
| | 4 | MICRO | | | Row_1_Col_3 | | | | |
| | 5 | MICRO | | | Row_1_Col_4 | | | | |
| | 6 | MICRO | | | Row_1_Col_5 | | | | |
| | 7 | MICRO | | | Row_2_Col_1 | | | | |
| | 8 | MICRO | | | Row_2_Col_2 | | | | |
| | 9 | MICRO | | | Row_2_Col_3 | | | | |
| | 10 | MICRO | | | Row_2_Col_4 | | | | |
| | 11 | MICRO | | | Row_2_Col_5 | | | | |
| | 12 | MICRO | | | Row_3_Col_1 | | | | |
| | 13 | MICRO | | | Row_3_Col_2 | | | | |
| | 14 | MICRO | | | Row_3_Col_3 | | | | |
| VBS module sheet | | A | B | C | D | E | F | G | H |
| | 1 | GAN | SampleID | Group | Chart Elements | Content | oVbsHorizontal | oVbsHorizontalHS | oVbs |
| | 2 | MACRO | | | Main | | | | |
| | 3 | | | | | | | | |
| | 4 | | | | | | | | |
| | 5 | | | | | | | | |
| | 6 | | | | | | | | |
| | 7 | | | | | | | | |
| | 8 | | | | | | | | |
| | 9 | | | | | | | | |
| | 10 | | | | | | | | |
| | 11 | | | | | | | | |
| | 12 | | | | | | | | |
| | 13 | | | | | | | | |
| | 14 | | | | | | | | |

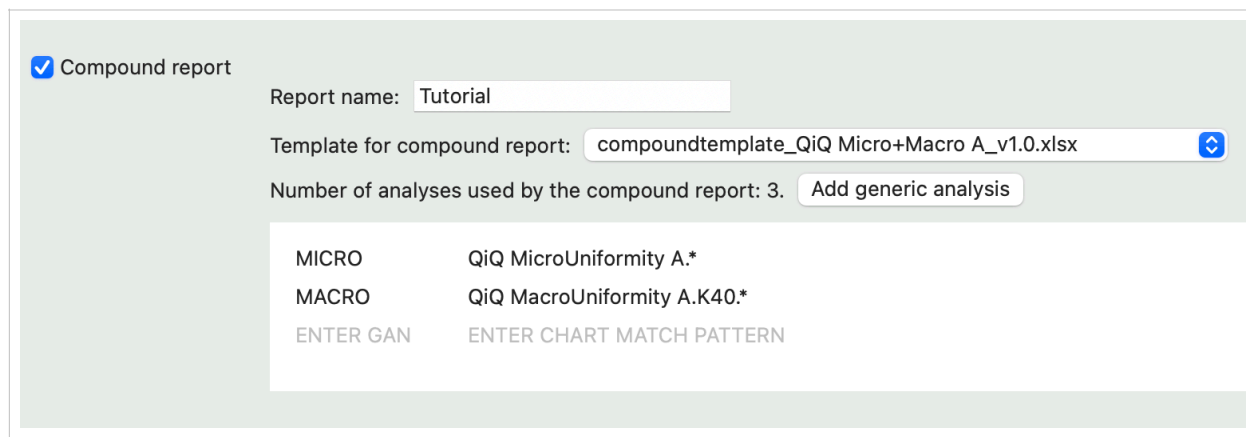
Figure: Compound report template

The figure of the compound report template shows the three module sheets. Notice this:

- On each sheet column A (“GAN”), has been filled in with either “MICRO” or “MACRO”: these are names that will be setup later to refer to the charts “QiQ MicroUniformity A” and “QiQ MacroUniformity A”, respectively. Results from the micro-uniformity chart will be placed in rows marked by GAN=MICRO, while results from the macro-uniformity chart will be placed in rows marks by GAN=MACRO.
- On the CIEColor module sheet, notice that column D contains chart elements from both charts.
- The other module sheets (Mottle and VBS) are identical to those from the basic templates, except that GAN values have been added in column A.


That covers creation of a template for the compound report.

The second step is to configure the project to use the compound report, including making a connection between the two different GAN values (MICRO and MACRO) and the corresponding charts.



☒ Compound report

Report name:

Template for compound report: 

Number of analyses used by the compound report: 3.

| | |
|-----------|-----------------------------|
| MICRO | QiQ MicroUniformity A.* |
| MACRO | QiQ MacroUniformity A.K40.* |
| ENTER GAN | ENTER CHART MATCH PATTERN |

Figure: Project configuration of compound report.

Go to the Configuration tab of the QiQ Tutorial Project (see the figure).

- The [Compound report checkbox](#) is selected, indicating that the project can generate compound reports.
- The [Report name](#) “Tutorial” will be included as part of the file name of the compound report.
- The [Template for compound report](#) selects the Excel template as explained earlier.
- The button [Add generic analysis](#) is used to add entries for each analysis that is part of the compound report.
- In the figure, there are two analysis entries. The first entry is: “MICRO” and “QiQ MicroUniformity A.*”; MICRO is called the GAN (generic analysis name) and will be matched to the GAN column of the Excel template. “QiQ MicroUniformity A.*” is the *chart match pattern*. This specifies that the compound report requires data from a chart with a name that matches “QiQ MicroUniformity A” (“.*” will match any version of the chart). Data from that chart will be placed into report rows with the matching GAN.

The third and final step is to generate the compound report with the QiQ user interface. The analyses of each chart must already have been completed. Then select [Results > Create Compound Report...](#), which open a new window:

Near the top, select the project, in this case QiQ Tutorial Project. The window shows three parts:

- [Generic analyses](#): This lists the charts that are required for the compound report, including the corresponding GAN, as was defined by the project configuration in the previous step. It also shows the number of analyses in the data base that match the required chart name—in the case 59 (MICRO) and 9 (MACRO), respectively.

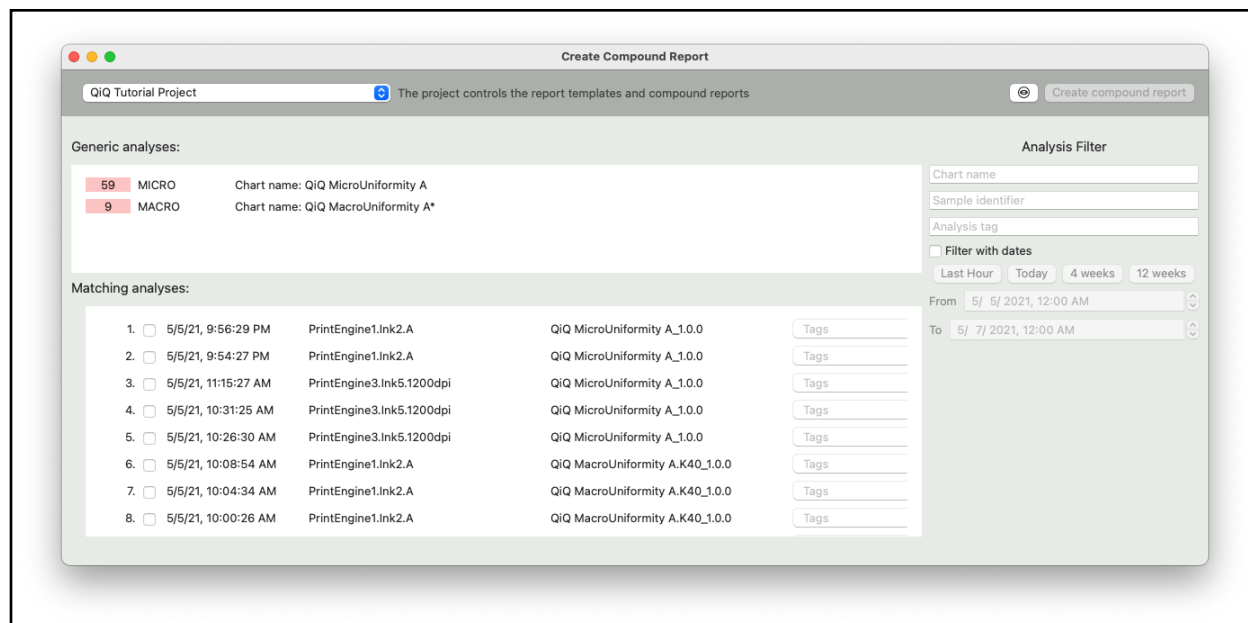


Figure: The Create Compound Report window.

- **Analysis filter:** You have to find and select the specific analyses that you want to contribute to the compound report, and to help find it more quickly, use the filter options. This will limit the analyses listed under **Matching analyses**.
- **Matching analyses:** List of all the analyses that match the filter criteria.

For now, we will not use the analysis filter. Instead, select the first entry (MICRO) under generic analyses. This will cause the matching analysis to highlight only those that can be used for GAN=MICRO:

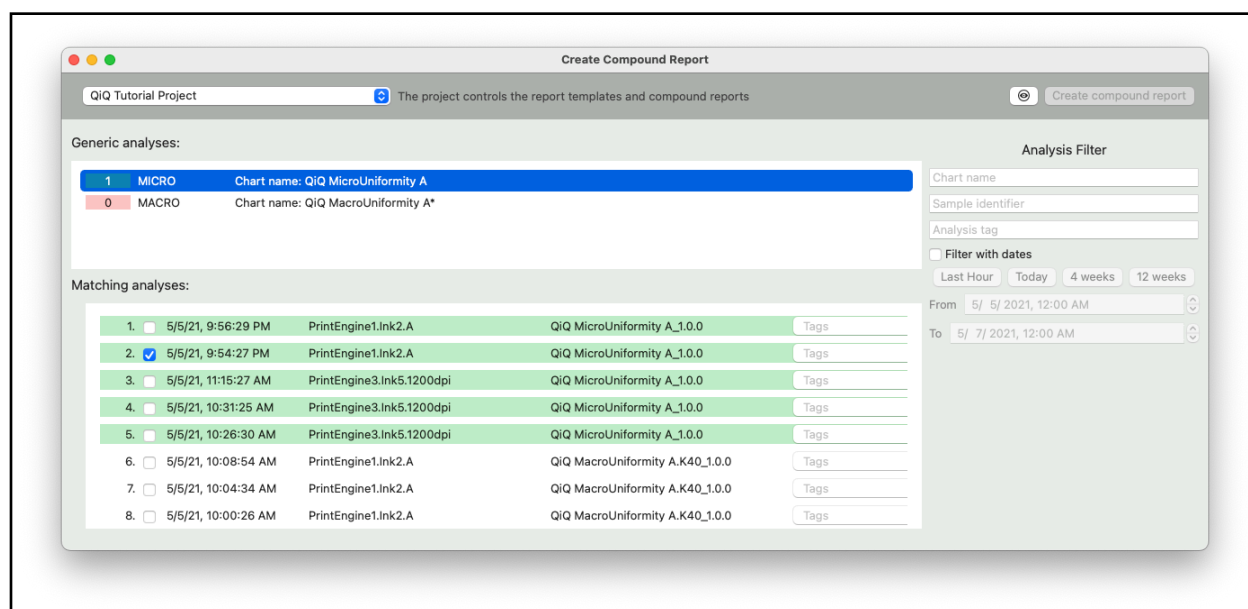


Figure: Selecting a generic analysis will highlight the matching options, where the specific analysis can be selected.

In this case we select entry #2 to be used for MICRO. Similarly, we select entry #6 to be used for MACRO.

This compound report requires exactly one analysis matching GAN=MICRO, and one analysis matching GAN=MACRO , so once those requirements are met, the generic analysis entries show the matching counts (1) highlighted in green, and the [Create compound report](#) button becomes enabled. Select the button, and the compound report will be created in the project reports folder.

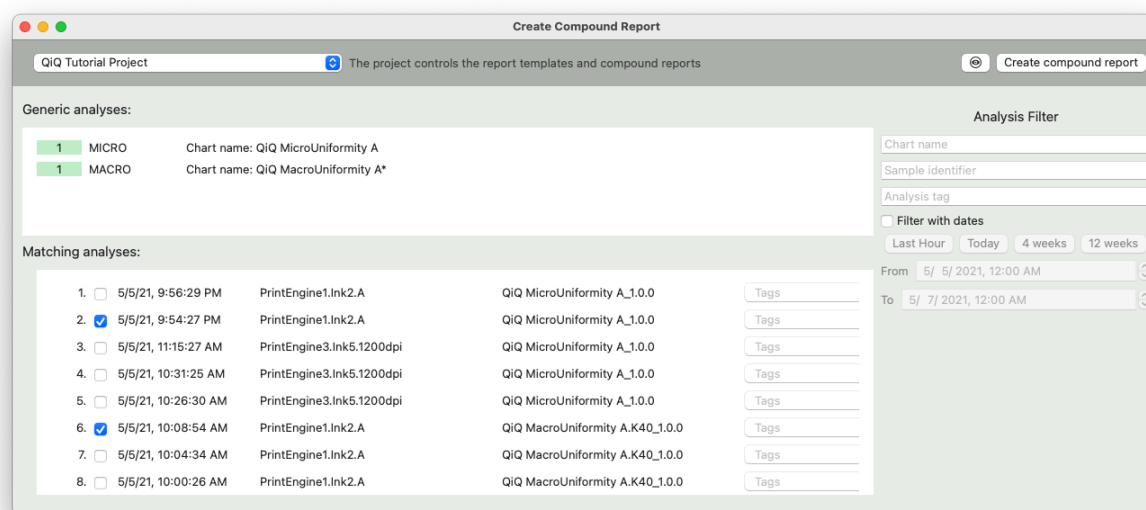


Figure: Ready to create a compound report.

Compound reports generally

To understand the logic behind how compound reporting works in QiQ, it is helpful to consider some examples. Here are examples of several quite different situations where compound reports are useful.

1. Time-zero quality study. The full characterization requires multiple test charts. "ChartA" has many, differently colored, regions of interest where graininess measurements are performed. "ChartB" has a single, large region of interest with a uniform color, where banding and mottle measurements are performed. Rather than two Excel reports, one for ChartA and one for ChartB, a single compound report is desired. This example involves two different charts, two different analysis scripts, and two different samples. A total of 2 analyses must be completed.
2. Light-fastness study. The full characterization requires two measurements on one and the same sample, separated by a 24 hour time period. The compound report should contain data from both the first and second analysis. This example involves only one

chart, one analysis script performed twice, and one sample. A total of 2 analyses must be completed.

3. Quality variation over long print run. The full characterization requires analysis of many (say 20) “identical” samples drawn from a long print run of ChartB (as in the first example). On each sample only a single measurement of mottle is performed. Rather than 20 reports each with just one measurement result, it is desirable that all 20 measurements are presented in a single compound report. This example involves only one chart, one analysis script, but 20 different samples. A total of 20 analyses must be completed.

In each of these examples (and in general) we need three types of information, in order to create the compound report:

- *Data selection criteria*: To select the completed analyses that are required as input to the compound report.
- An Excel report template that has been setup to hold results from multiple analyses.
- *Data mapping*: To specify where in the report (sheets, rows) the results from each of the selected analyses should be placed.

Details of compound reporting

The rules that govern how a compound report is created is in fact more flexible than the current user interface allows, and can support more complex reporting needs.

Data selection and mapping (GAN)

In general, there are two parts to selecting an analysis to contribute data to a compound report:

- **The GAN (generic analysis name).** The GAN is a text string that you choose. It serves to map the analysis data to positions in the template. It must be used in the GAN column of the report template to specify the rows where results from the analysis should be placed.
- **The analysis match criteria.** This serves to select one specific analysis. With the current user interface (v0.9, as explained above) you can specify a match pattern for the chart identifier. It is a text string with a *regular expression match pattern*. Regular expressions provide a powerful way to perform text matching, and full documentation can be found online. Suffice it here to say that ‘.’ (a period followed by an asterisk) will match any text. If there are multiple analyses which match the criterion, then additional criteria will be needed to select one specific analysis (e.g. by manual selection via the user interface).

Currently, the user interface allows you to specify analysis match criteria only on the chart identifier. However, criteria can also be set to match the sample identifier or the analysis tag.

Finally, a criterion can be set to match the UUID of a specific analysis, which will allow one single analysis to always become part of the compound report, for example to serve as reference data.

Examples of selection criteria and mappings

| Generic analysis name (GAN) | Selection criteria |
|-----------------------------|---------------------------------|
| CHART_A | Chart name must match "ChartA_* |
| CHART_B | Chart name must match "ChartB_* |

Example: Compound report that combines multiple different charts.

| Generic analysis name (GAN) | Selection criteria |
|-----------------------------|---------------------------------------|
| Reference | Sample name must match "**Time=0**" |
| Time_24Hour | Sample name must match "**Time=24H**" |
| Time_1Month | Sample name must match "**Time=1M**" |

Example: Compound report that combines the same chart from multiple samples.

Generating the compound reports

The user interface ([Results > Create Compound Report](#)) allows you to filter previously completed analyses from the database, select those that should be included in a compound report, and then generate the report.

However, in most cases, it is more convenient to let QiQ automatically generate the reports as new analysis results become available. From the projects view, select a project and select the [Reports](#) tab, as shown in the figure:

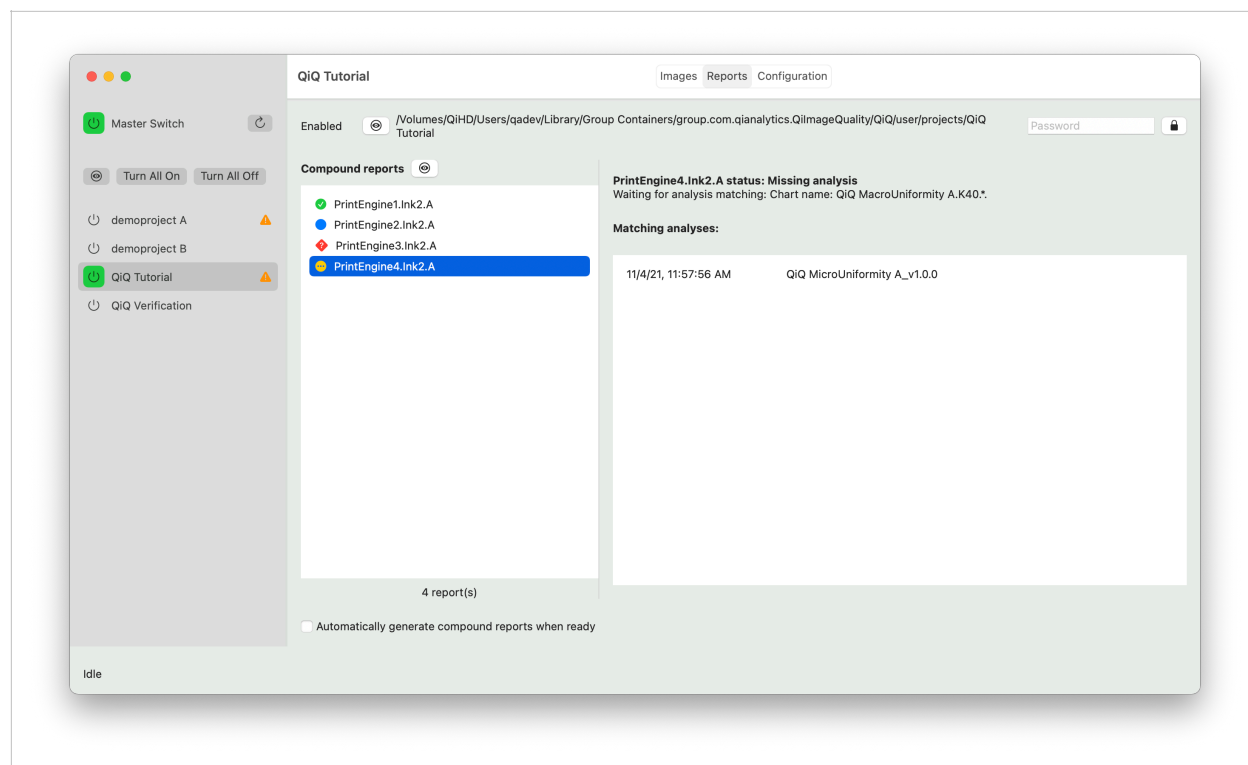


Figure: The project report tab.

The list of compound reports (in the project report tab)

This view lists all the sample IDs for which one or more analyses have been completed within the selected project. When completed, each sample ID should result in one compound report. The figure illustrates a case where there are four sample IDs. In front of the sample ID is an icon that indicates the status of the report:

- **Completed status.** A green filled circle with checkmark icon indicates the compound report has been completed.
- **Ready status.** A blue filled circle icon indicates that all the analyses required to generate the report are available (and without any ambiguity), so that the report is ready to be generated. If the toggle [Automatically generate compound reports when ready](#) had been selected, then this would immediately change to the completed status (and the report would be generated).
- **Problem status.** A red diamond with question mark icon indicates there is a problem that prevents a report from being generated. Typically, this is because there are more analyses for than required, and it is ambiguous which should be used for the report. See below how such a situation can be resolved.
- **Waiting status.** A yellow filled circle with ellipses icon indicates that only some, but not all, the required analyses have been completed.

From this list you can use a context menu on an entry and select [Generate report](#), to generate the report (if the status is either ready or completed). Using this is relevant only if the toggle to automatically generate the reports is turned off.

The detail view for a sample ID (right side of the project report tab)

The right side of the project report tab shows details for the compound report entry that is selected in the list. It has two parts:

- Status and explanation. This will explain which analyses are missing (if waiting for analyses), or explain the reason in case of a problem.
- List of matching analyses. This lists all the analyses that match the criteria to be included in the compound report for the selected entry. A context menu gives a couple of options:
 - [Delete analysis](#) can be useful in case of ambiguity. Note that this permanently deletes the analysis data from the database. An alternative option to deal with such ambiguity is to use [Results > Create Compound Report](#) to manually select analyses.
 - [Mark as not used for report](#) will change a marker that the analysis has not yet been used to generate a compound report. This is useful only in rare situations.

Test Charts—Digital Charts, Chart Scripts and Fiducial Marks

There are two components when we discuss test charts:

- The *digital chart* which is used as input to a printer, for example a PDF file.
- The JavaScript *chart script* that QiQ uses for a description of the content of the test chart.

Both the digital chart and the chart script must have version information, and the versions must be consistent, but not necessarily identical. It is important to keep the version information consistent in order to ensure a correct analysis. This is explained below.

What are chart scripts

A chart script provides QiQ with the information needed for an automatic analysis. It specifies the *measurable elements* of the chart:

- The type of element (color patches, lines, color registration marks, fiducial marks, etc.)
- The exact content (CMYK percentages, line width, etc.)
- The size and position on the chart.
- Name of the element, which can be used when reporting results.

QiQ uses the JavaScript programming language for chart scripts, and provides a set of functions and objects (an API) to help define the chart.

Optionally, QiQ can generate a PDF digital chart from the chart script, which can be used to print the chart. This means you can, in some cases, entirely avoid having to design the test chart using other tools, and at the same time be assured that the chart script is a completely accurate description of the test chart. When you design a test chart with a script, you have the added benefits of being able to use JavaScript to perform calculations of the sizes and placement of elements, and of being able to use control structures (loops) to avoid repetitive tasks.

The digital (and printed) chart name, version and identifier

The digital chart (and hence the printed chart) must have a name and a version. The name and version are combined into a *chart identifier*. For example, QiQ comes with a test chart named “QiQ Lines A” with the version “1.2.4” (current version at time of writing). The full chart identifier is “QiQ Lines A_v1.2.4”, composed in the form *NAME_vMAJOR.MINOR.PATCH*. The chart identifier should be part of the chart contents, ideally both in plain text and as a QR code.

The version must consist of 1, 2, or 3 integers separated by a period ‘.’, which should be used as follows:

- The first integer (mandatory) is called the *major version number*. A change of the major version must be used to indicate that the chart has changed very significantly, and may break all compatibility with the prior major version of the chart—just as if it was an entirely different chart (with a different name). The final reports may look very different after a change of the major version number. By changing the major version, rather than the name, you indicate that going forward this new major version is recommended instead of prior versions with the same name.
- The second integer (optional) is called the *minor version number*. A change of the minor version indicates that some change has occurred to the measurable elements of the chart, however, the changes are minor and “backwards compatible” in the sense that measurements of the chart should be directly comparable with measurements of the prior version. Typically, the same final report templates can be used with the new minor version, or they may need to be updated to include new, additional elements. Here are examples of changes that would require an increment to the minor version number:
 - An existing measurable element has changed position on the chart.
 - An existing measurable element has changed size, however, the size change is so small that measurement results should not be affected significantly.
 - New measurable elements have been added in previously empty regions of the chart, but existing elements have not been eliminated.
- The third integer (optional) is called the *patch number*. A change of the patch number should be used to indicate that changes were made, but no measurable elements were changed at all. Here are examples of changes that would require only an increment to the patch version number:
 - A company logo on the chart, that is not measurable, has been added or moved.
 - Text, that is not measurable, has been added or modified—for example, adding a copyright notice or correcting a typo.

Name and version of the chart script

The name, major version, and minor version of the chart script must be identical to those of the digital chart—they must not be changed except to reflect a corresponding change of the digital chart. However, the patch number of the chart script can be changed independently of the digital chart.

Notice, that the chart name and version of a chart script is given by the file name. The file name of the script must therefore be chosen accurately. For example, the file named “QiQ Lines A_v1.2.4.js” is version 1.2.4 of the script for the chart named “QiQ Lines A”.

For a chart script, the patch version indicates that the script has been corrected or updated. Here are examples of changes to the chart script that would require the **patch** version to be incremented, but would not require any change to the digital chart:

- A bug in the script was fixed, correcting the position of a measurable element to be in agreement with the digital (and printed) chart.
- A bug in the script was fixed, correcting the CMYK percentages of a measurable element to be in agreement with the digital (and printed) chart.
- The names of measurable elements were changed (this would typically require the final report templates to be modified as well).
- The script code was changed to be more legible, for example by adding comments (without any impact on the analysis).
- The script code was changed to be compatible with a new version of QiQ.

Installation and use of chart scripts

A chart script is installed by placing it in the folder QiQ / user / configuration / scripts / chart.

To analyze a digital chart, a chart script must be installed which exactly matches on the name, major, and minor version. **If multiple chart scripts are installed with the same name, major and minor versions**, then **QiQ always uses the script with the highest patch version**. (To keep the installation simple, it is recommended to remove any lower patch versions from the installation).

Thus, the patch versions of the digital chart and of the chart script are typically not “in sync”. The highest patch version always represents the newest, best, and most correct version—whether for the digital chart or for the chart script, but independently of each other.

If you need to analyze prints with various different patch versions of the same chart (same name, major and minor versions), that is possible by installing the highest patch version of the chart script: all print versions will be analyzed using the newest, most correct script.

| Digital chart (and print) | Chart script |
|---|--|
| Increment of major version (reflecting an essentially new test chart) | New script must be created with same name, major and minor versions. |
| Increment of minor version (reflecting a change to the measurable elements of the test chart) | |
| Increment of patch version (reflecting change that does not affect any measurable elements) | No change required. Continue to use same chart script without changes to content or version. |
| No change. | Increment of patch version reflecting changes in the script code (for example to correct a bug). |

Summary of digital chart and script version changes

Example of the life cycle of a chart and chart scripts

This is a hypothetical example of creation of a new test chart and script, and updating each for different reasons. The list describes the sequence of events, and the following table illustrates the QiQ installation uses chart scripts depending on which version of the chart is being analyzed. We will assume the digital chart is in the form of a PDF.

1. Created new chart named “MC” with version 1.0.0 and a corresponding script. The chart identifier is MC_v1.0.0, the digital file name is MC_v1.0.0.pdf, and the script file name is MC_v1.0.0.js.
2. Fixed a bug in the digital chart (a typo in the copyright notice). Since the change does not involve measurable elements, we need only increment the patch version of the digital chart. The new digital chart identifier is MC_v1.0.1. There is no need to change the script.
3. Fixed a bug in the chart script (the position of a line element is slightly wrong). The new script is named MC_v1.0.1.js. The old script is removed from the installation. The new script is simply a better (hopefully now correct) description of the test chart, and it should be used for analysis also of the prior version.
4. The chart script is updated again, this time just to add some comments and make the code more legible. The updated chart script is named MC_v1.0.2.js. By mistake, the previous chart script is not removed from the installation, but QiQ will automatically use only the script with highest patch version.
5. Fixed a “serious” bug in the digital chart (a line width was 0.2 mm rather than 0.1 mm). Since this change involves a measurable element, the minor version of the digital chart must be incremented. The corrected digital chart is called MC_v1.1.0. Even though the existing chart script already has the correct line width, we must create a script that matches the name, major, and minor version of the digital chart: we just copy MC_v1.0.2.js

to MC_v1.1.0.js (resetting the script patch version to 0). We decide that even though the old digital chart had an error, we still need to be able to analyze legacy prints based on the old chart, therefore we keep script MC_v1.0.2.js in the installation. The obsolete script MC_v1.0.1.js, that (in step 4) remained in the installation by mistake, is now removed.

6. In an empty region of the chart we add an element to measure color-color registration. The new version of the chart is MC_v1.2.0, where the increment of the minor version indicates that the measurable content of the chart has been changed. The chart can still serve all purposes of the prior version. A new chart script, MC_v1.2.0.js, is created, including information on the new color-color registration element. The old Excel report template can still be used, but to take advantage of the new measurement it should be updated.

At the same time we decide that prints of the legacy chart MC_1.0 should no longer be supported, because that version contained a line with the wrong width. We therefore remove all MC_1.0 chart script versions from the installation, preventing such prints from being analyzed.

7. The test chart is completely revised. It still serves the same basic purpose, but has been improved in various ways, eliminating some measurable elements and adding new ones. This calls for a change to the major version, and for a new chart script: MC_2.0.0 and MC_2.0.0.js, respectively.

We continue to support version 1.1 and 1.2 of the chart, but a new version of the scripts are created, changing some element names (to be more consistent with names used in the MC_2.0.0 test chart). The patch versions are incremented, naming the updated scripts MC_v1.1.1.js and MC_v1.2.1.js.

8. A new version of QiQ requires a change to the chart script, because one of the required function names has been changed. All the installed chart scripts must be updated. Each script is updated and its patch version incremented. No changes are required to the digital charts.

| # | New digital chart identifier | Installed chart scripts | Print analyzed | Script used | Reason script is used |
|---|------------------------------|-------------------------|----------------|--------------|----------------------------|
| 1 | MC_v1.0.0 | MC_v1.0.0.js | MC_v1.0.0 | MC_v1.0.0.js | Only matching script |
| 2 | MC_v1.0.1 | MC_v1.0.0.js | MC_v1.0.0 | MC_v1.0.0.js | Only matching script |
| | | | MC_v1.0.1 | MC_v1.0.0.js | Only matching script |
| 3 | | MC_v1.0.1.js | MC_v1.0.0 | MC_v1.0.1.js | Only matching script |
| | | | MC_v1.0.1 | MC_v1.0.1.js | Only matching script |
| | | | MC_v1.0.0 | MC_v1.0.2.js | Use highest patch version. |

| # | New digital chart identifier | Installed chart scripts | Print analyzed | Script used | Reason script is used |
|---|------------------------------|--|----------------|--------------|---|
| 4 | | MC_v1.0.1.js MC_v1.0.2.js | MC_v1.0.1 | MC_v1.0.2.js | MC_v1.0.1.js will never be used and might as well be removed from the installation. |
| 5 | MC_v1.1.0 | MC_v1.0.2.js MC_v1.1.0.js | MC_v1.0.0 | MC_v1.0.2.js | Only matching script |
| | | | MC_v1.0.1 | MC_v1.0.2.js | |
| | | | MC_v1.1.0 | MC_v1.1.0.js | Only matching script |
| 6 | MC_v1.2.0 | MC_v1.1.0.js MC_v1.2.0.js | MC_v1.0.0 | Fails | There is no script with acceptable match. |
| | | | MC_v1.0.1 | Fails | |
| | | | MC_v1.1.0 | MC_v1.1.0.js | Only matching script |
| | | | MC_v1.2.0 | MC_v1.2.0.js | Only matching script |
| 7 | MC_v2.0.0 | MC_v1.1.1.js MC_v1.2.1.js MC_v2.0.0.js | MC_v1.0.0 | Fails | There is no script with acceptable match. |
| | | | MC_v1.0.1 | Fails | |
| | | | MC_v1.1.0 | MC_v1.1.1.js | Only matching script |
| | | | MC_v1.2.0 | MC_v1.2.1.js | Only matching script |
| | | | MC_v2.0.0 | MC_v2.0.0.js | Only matching script |
| 8 | | MC_v1.1.2.js MC_v1.2.2.js MC_v2.0.1.js | MC_v1.0.0 | Fails | There is no script with acceptable match. |
| | | | MC_v1.0.1 | Fails | |
| | | | MC_v1.1.0 | MC_v1.1.2.js | Only matching script |
| | | | MC_v1.2.0 | MC_v1.2.2.js | Only matching script |
| | | | MC_v2.0.0 | MC_v2.0.1.js | Only matching script |

Fiducial marks

QiQ currently supports only one kind of fiducial mark, called “BullsEye”. If you create a test chart directly with a script, QiQ will take care of creating the fiducial mark with correct dimensions. If you create the test chart using a different tool, then make sure to use these precise dimensions:

BullsEye fiducial mark specifications

| | |
|---------------------------------|---------|
| Radius of the dot in the center | 0.55 mm |
| Inner radius of the ring | 1.15 mm |
| Outer radius of the ring | 1.60 mm |

Working with distorted (or missing) fiducial marks

In practice, during testing, the images you wish to measure may be far from perfect. This can pose a problem if the fiducial marks cannot be recognized. For situations like that, QiQ provides two options:

- You can manually indicate the fiducial mark positions on the scanned image (suitable if the mark is severely distorted).
- You can ask QiQ to learn to recognize the distorted fiducial mark (suitable when the mark is not severely distorted, and you expect to encounter multiple similarly distorted marks). NOT YET SUPPORTED IN CURRENT VERSION.

Manually indicating the fiducial mark positions on a scanned image

If a fiducial mark is missing or distorted, QiQ will report that as an error and not proceed with the analysis. In the example figure (A) below, an error message is seen: “QiQ error: Expected 5 fiducial marks but found 4 ...”.


From the image context menu select [Fiducial Marks...](#), which will open the image in a new window. In the example figure (B) below we see that for one of the fiducial marks the outer ring is broken. Manually tap at the center of the fiducial mark, then a red filled circle will indicate the manually specified position (see figure (C)).

You must manually specify all the fiducial marks, even those that are not distorted.






Once you have specified all the fiducial marks, click the [Save](#) button. The manually specified positions of the 5 fiducial marks are now stored within the TIFF image file, and will be used for any subsequent analyses. Note on figure (D) below that when the image is selected a warning message is displayed “Image uses manual fiducial marks (5)”.

The [Reset](#) button in figure (B) will remove all the manually entered fiducial mark information. If you wish to change the position of a manual mark, then first remove it using option-click, then add a new position.

A) **QiQ Tutorial Project** Images Configuration

Enabled  /Volumes/QiHD/Users/qadev/Library/Group Containers/group.com.qianalytics.QiImageQuality/QiQ/user/projects/QiQ Tutorial Project

Archive completed images One image selected Deselect

| | | | |
|---|------------------------------------|---------------------------|---------|
|  | PrintEngine1.Ink2.A | QiQ MacroUniformity A.K40 | v0.10.0 |
|  | PrintEngine1.Ink2.A | QiQ MacroUniformity A.K40 | v1.0.0 |
|  | PrintEngine1.Ink2.A | QiQ MicroUniformity A | v1.0.0 |
|  | PrintEngine1.Ink3.missing fiducial | QiQ MacroUniformity A.K40 | v1.0.0 |
|  | PrintEngine2.Ink3.wrong QR code | QiQ MicroUniformity A | v1.0.1 |

PrintEngine1.Ink3.missing fiducial_QiQ MacroUniformity A.K40_v1.0.0.tif
Modification date: 6/21/21, 10:38 AM

Analysis status: Failed
QiQ error: Expected 5 fiducial marks but found 4 at match(image:)
[#0148]:


Analyses of this image:

B) **PrintEngine1.Ink3.missing fiducial_QiQ MacroUniformity A.K40_v1.0.0.tif**

Manually specified fiducial marks: none

Reset Save

Tap to mark fiducial mark. Option-tap to remove.




C) **PrintEngine1.Ink3.missing fiducial_QiQ MacroUniformity A.K40_v1.0.0.tif**

Manually specified fiducial marks: 1: (196.5, 28.6) 2: (164.4, 28.5) 3: (12.5, 28.5) 4: (12.5, 246.1) 5: (196.6, 246.2)

Reset Save

Tap to mark fiducial mark. Option-tap to remove.



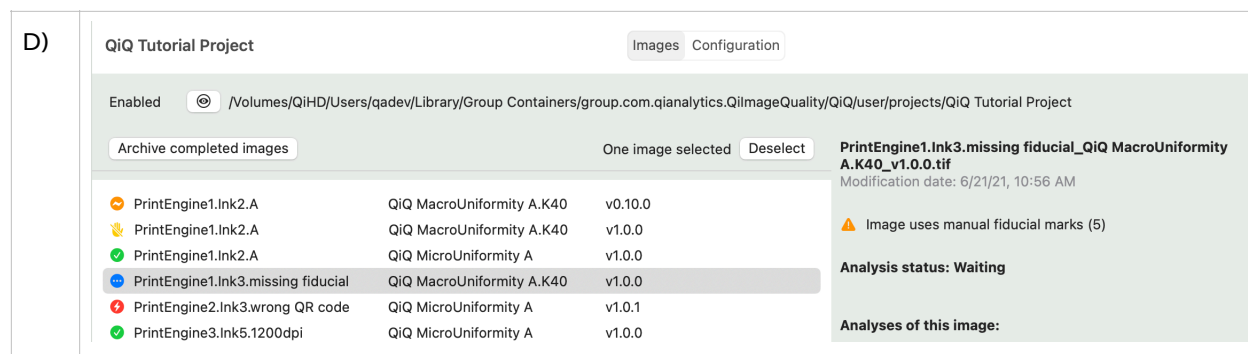


Figure: Distorted fiducial mark

Teaching QiQ to recognize distorted fiducial marks

NOT SUPPORTED BY CURRENT VERSION

QiQ allows you to teach it to recognize fiducial marks that are distorted more than is accepted by default. This is particularly useful if you expect to often encounter similarly distorted fiducial marks.

If you encounter an error message about not being able to find all the fiducial marks, then look at the image and notice which of the fiducial marks have not been found (those that are found are indicated with a blue circle). Tell QiQ to accept the fiducial mark as follows:

- Zoom in on the fiducial mark.
- Select a region of the image that contains only the fiducial mark.
- Select [Learn Fiducial Mark](#) from the [Charts](#) menu.
- You should then see a message that the custom fiducial mark data has been updated.

This creates a system file `CustomSystemData/fiducials/bullseye/bullseye.json`. If you wish to return to the default tolerances for distortion, you can simply delete this file.

If you repeat this process with other distorted fiducial marks, the custom data will continue to be “extended” to accept them all.

Notice, that there are limits to the amount of distortion that will work, and the mark should always consist of a dot within a ring.

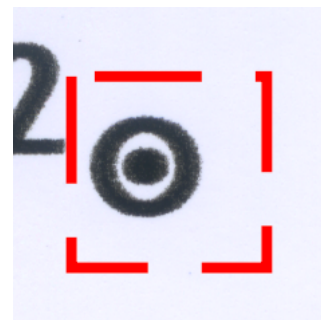


Chart example

The following example chart is available from [Charts > System Charts > Example Chart A](#). The generated PDF is shown in the figure below.

```
let chartWidth = 210;
let chartHeight = 297;
```

These lines define the chart to be A4 size.

```
function defineChart(chartName, chartVersion) {

    let chart = QQTestChart.create(chartName, chartVersion,
                                   chartWidth, chartHeight);

    chart.verbose = true; // show log information as we add elements to the chart.
```

A chart object is created with a name, a version, width, and height. With verbose on, the progress view will show output that may be useful for debugging while developing the chart.

```
// fiducial marks:
let fiducialInset = 10;
chart.addFiducial("Bullseye", fiducialInset, fiducialInset);
chart.addFiducial("Bullseye", chartWidth - fiducialInset, fiducialInset);
chart.addFiducial("Bullseye", chartWidth - fiducialInset,
                  chartHeight - fiducialInset);
chart.addFiducial("Bullseye", fiducialInset, chartHeight - fiducialInset);
chart.addFiducial("Bullseye", chartWidth/3, chartHeight - fiducialInset);
```

Here we define the positions of the fiducial marks. First we define a constant, `fiducialInset`, which is the distance in mm units, that we want from the chart edges to the fiducial marks. We then add the fiducial marks to the chart, giving a name “BullsEye”, which is the only currently supported type of fiducial mark. The positions of the center of the marks are given as left and top, measured from the left, top corner of the chart. So the first fiducial mark will be centered at (10, 10).

```
// upper swatch (green)
var roi = QQROI.create(20, 20, 50, 50); // left, top, width, height
let swatch1 = QQChartFlatField.createKCMY(roi, 0, 0.5, 0.0, 0.5);
swatch1.identifier = "Green 50%";
chart.addElement(swatch1);
```

Here we define a green swatch. First we create a ROI (region-of-interest). All distances are measured from the left / top edges of the chart. The first parameter (20) is the distance from the left chart edge to the left edge of the ROI. Similarly, the second parameter (20) is the distance from the top chart edge to the top edge of the ROI. The next two parameters (50, 50) are the width and height of the ROI.

We then create a chart element called `QQChartFlatField`, which represents a region with a uniform color fill. The first parameter is the roi that we had just created, specifying the position and size of the element. The last 4 parameters are K, C, M, Y levels on a scale from 0.0 to 1.0. Here we have 50% cyan and 50% yellow.

```
// lower swatch (gray)
roi = QQROI.create(20, 70, 50, 50);
```



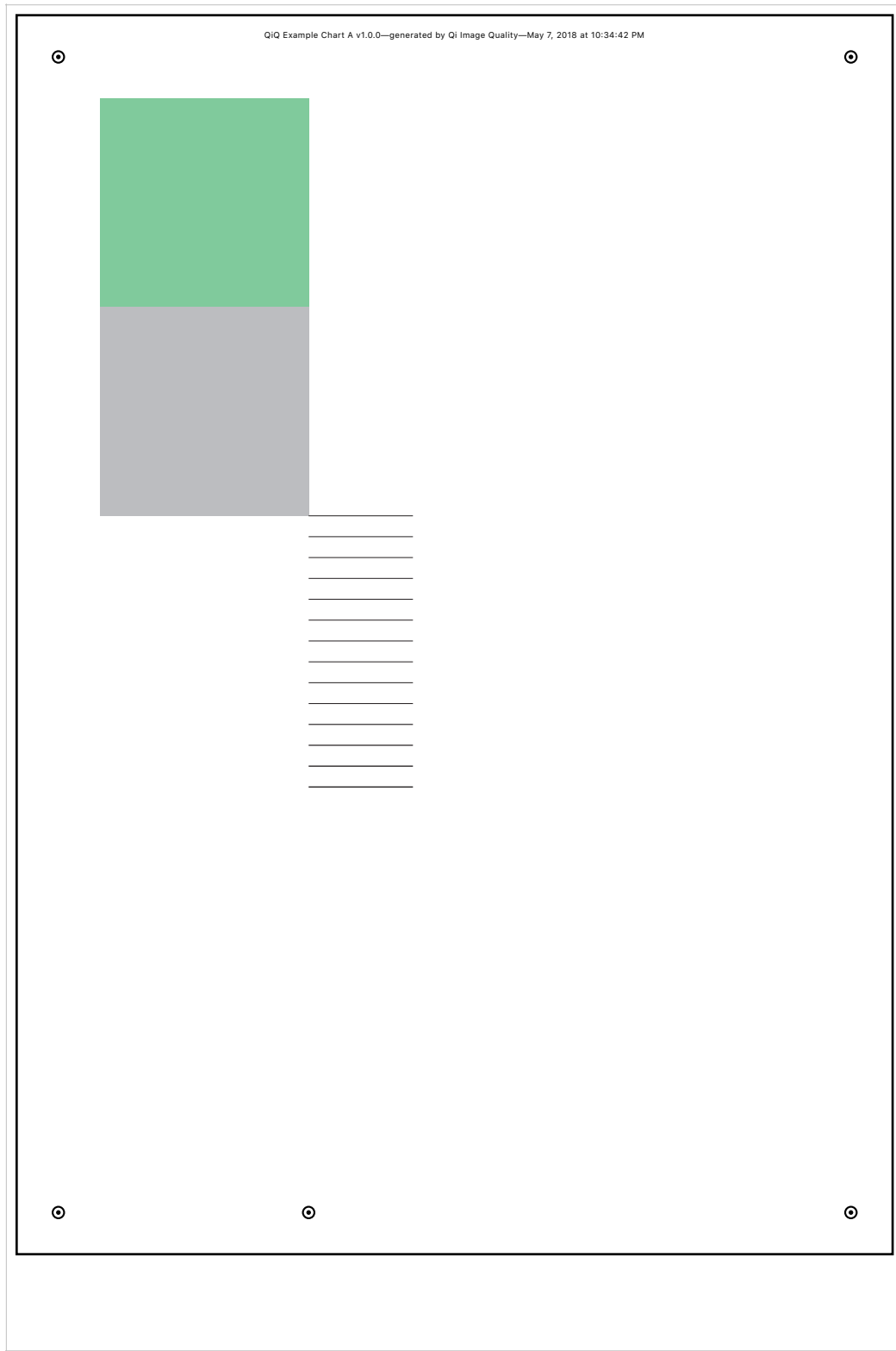
```
let swatch2 = QQChartFlatField.createKCMY(roi, 0.3, 0.0, 0.0, 0.0);  
swatch2.identifier = "K 30%";  
chart.addElement(swatch2);
```

This is just as above, but creating a swatch with 30% K, positioned just below the green swatch, 70 mm below the top of the chart.

The next segment of code illustrates how the power of JavaScript can be used to perform calculations and use loops to create a series of chart elements. We create 14 horizontal lines of

width from 1/1200 of an inch incrementing by 1/1200 of an inch for each line.

Figure



```

// lines
let linesLeft = 70;           // distance from chart to line left start
let firstLineY = 120;         // distance from chart top to first line
let lineDeltaY = 5;           // distance between lines
let lineWidth = 25.4 / 1200;
let widthIncrement = 25.4 / 1200;
let angle = 0;
let length = 25;
let padLeft = 0.0;
let padRight = 0.0;
let padTop = 2.5;
let padBottom = 2.5;
for (i=0; i<14; i++) {
    let lineElement = QQChartLine.create(linesLeft, firstLineY + i*lineDeltaY,
                                          angle, length,
                                          lineWidth + i*widthIncrement,
                                          1.0, 0, 0, 0, // K, C, M, Y
                                          padLeft, padRight, padTop, padBottom);

    chart.addElement(lineElement);
}

```

First we setup three constants related to the position of the lines. Then we set the line width for the first line to 1/1200 inch. We set a constant, `widthIncrement`, that will be the incremental line width.

The angle of the line is set to zero. The angle is given in degrees, counter-clockwise from the positive x-axis. This means a line of angle 0 points horizontally to the right, while, for example, an angle 90 would point up.

The length of the line is set to 25 mm.

Then follows a loop that creates the 14 lines. Inside the loop a `lineElement` is created, where the vertical position is calculated from the distance between the lines. The line width is incremented as well. The next 4 parameters are the K, C, M, Y colors (1, 0, 0, 0). The last four parameters specify padding around the lines (see the section on `QQChartLine` for more information on this).

The last line of code adds the line element to the chart.

```

//
return chart;
}

```

Finally, at the very end of the script we must return the chart we just created.

By selecting [Chart > Create PDF...](#) you can generate a PDF from this script. It is shown in the figure below.

Color swatches: QQChartFlatField

The type QQChartFlatField is used to create an element with a constant (flat) color.

```
let roi = QQROI.create(left, top, width, height);
let element = QQChartFlatField.createKCMY(roi, K, C, M, Y);
```

The position and size are specified in mm units. The color is specified by K, C, M, Y in the range from 0.0 to 1.0.

Lines: QQChartLine

The type QQChartLine is used to create a line element.

```
let lineElement = QQChartLine.create(x, y, angle, length, width,
                                     K, C, M, Y,
                                     padLeft, padRight, padTop, padBottom);

lineElement.backgroundK = c;
lineElement.backgroundC = m;
lineElement.backgroundM = y;
lineElement.backgroundY = k;
```

(x, y) are coordinates of one end of the line. The angle is in degrees, with zero indicating a horizontal line extending rightwards of (x, y), and positive values counted counter-clockwise. Padding defines the ROI surrounding the lines that should be used when the line is analyzed.

Currently only horizontal (angle = 0) and vertical (angle = 90) lines can be analyzed by the Lines module.

Color-color registration: QQChartColorRegH and QQChartColorRegV

The types QQChartColorRegH and QQChartColorRegV are used to create elements for measurement of color-to-color registration in the horizontal and vertical direction, respectively.

```
let roi = QQROI.create(left, top, width, length);
let element = QQChartColorRegH.create(roi);
```

The element consists of 10 parallel lines. The width parameter is the distance from the first to the last line, while the length parameter is the length of each line.

The measurement module QQColorReg is optimized for width = 12 (mm) and length = 10 (mm), and **it is strongly recommended that these values be used.**

Scripts

QiQ allows you to use JavaScripts to control all aspects of the analysis. The program comes with builtin scripts for a set of standard analysis tasks, but as a user you can customize or create your own scripts. JavaScript is a general-purpose programming language.

To effectively use scripts it is good to have a basic understanding of the key components of the QiQ system.

- The QiQ Engine provides all the core capabilities: image access via the file system; optical conversion functions; machine vision technology; test chart semantics; general image analysis functionality; analysis algorithms for specific image quality attributes (such as mottle and graininess). It is by far the largest part of the QiQ system.
- The QiQ JavaScript API provides an interface that gives scripts access to some of the functionality of the QiQ Engine. It defines objects and methods which are tied to the QiQ Engine, and which can be used with the standard JavaScript language.
- QiQ uses two types of scripts: chart and analysis. They all follow standard JavaScript language syntax, and use the QiQ JavaScript API to get things done. The purpose of each type of script will be described a little later. From the user interface you can manually load and execute scripts, which is useful for testing while developing scripts for new purposes. For a fully automated operation you would only load the process script, which in turn would load chart and analysis scripts as necessary.
- The QiQ app provides the user interface, as described elsewhere in this manual.
- The QiQ Scanner app is a stand-alone tool that to use flatbed scanners to scan documents for subsequent analysis by the QiQ app.

There are two kinds of scripts: chart scripts at the lowest level and analysis scripts.

- Chart scripts define the content and layout of a test chart. Using objects defined by the QiQ Engine, such as fiducial marks, flat fields and lines, you can specify the types and positions of the elements that make up a test chart. You can also assign names and attributes (e.g. color) to the elements. In this way, the chart script defines a *chart object* that contains all information about the test chart, and this chart object can be used by the analysis script.
- The purpose of an analysis script is to perform the complete analysis of one image. If the image is a scan of a small one inch square flat field, a complete analysis could mean just one mottle measurement on that region. If the image is a scan of an entire test chart, it could mean hundreds of measurements of different types and on different elements of the test chart. The analysis script has various methods by which it can access regions of interest

(ROIs) on the current image, and then use other methods to perform a specific measurements on the ROI. For example:

- If you manually select a region on the image, the analysis script can access that region, and could perform one or more measurements on the region.
- If the loaded image is a scan of a full test chart, and if a chart script has defined a corresponding chart object, then the analysis script can use the chart object to select ROIs for the various elements of the test chart.

Chart scripts

The purpose of the chart script is to define a **chart** object that can provide all information about the content and layout of the test chart that needs to be analyzed. Once you have defined the chart object, it can be used to select ROIs on the image. Chart scripts are explained in detail in the chapter [Creating Test Charts](#)

Table scripting.1: Chart methods to select ROIs

| Method call | Meaning |
|--|--|
| <code>chart.selectElement(name)</code> | Returns the ROI for the element with the specified name. |
| <code>chart.selectElementsMatching(matchString)</code> | Returns a list of ROIs for those elements with a name that match the matchString. The matchString can use '*' as wildcard. |
| <code>chart.selectAllElements()</code> | Returns a list of ROIs for all elements defined by the chart, excluding fiducial marks. |
| | |
| | |

To be completed

Analysis scripts

To be completed

-

Measurements: Metric Modules

The system has several distinct modules to perform measurements; these modules are called *metric modules*, or sometimes for short just *metrics*. This chapter discusses details of each metric module. Some principles are common to all metrics and are explained here.

Using metric modules, parameters and outputs in the analysis scripts

Parameters allow you to control specific aspects of the analysis, selecting options different than the defaults. You must set them before calling the calculate function on the metric module. Outputs are calculated and should be accessed in the script only after the calculate function has been called. Each metric module defines its own particular set of parameters and outputs, as documented for each metric.

A typical fragment of an analysis script may look like this (omitting much-recommended error checking code which will be discussed elsewhere).

```
function analyze() {  
    let roi = imageSource.selection;  
    let measurement = QQGraininess.create();  
    measurement.pUseUnlimitedTileCount = false;  
  
    measurement.calculate(roi);  
    qqlog.log("D=" + measurement.oDensity);  
    qqlog.log("Graininess=" + measurement.oGraininess);  
}  
  
analyze();
```

This is Javascript code, using components defined by QiQ for image quality analysis. First a function is written, called *analyze*, which defines what the analysis should do. Then the *analyze* function is called, to do that.

Inside the *analyze* function several steps take place:

- The region-of-interest, that the user has selected from the image view, is put into a variable called *roi*. If the user has not selected an image region then the entire image will be used.

- An object to measure graininess is created and put into the variable named `measurement`.
- A parameter for the graininess measurement, named `pUseUnlimitedTileCount`, is set to `false`. This will affect the way the graininess analysis is performed.
- The graininess measurement object is told to perform a calculation on the `roi`.
- Two of the outputs of the graininess measurement object (`oDensity` and `oGraininess`) are written to the log.

Image requirements

Each metric module will specify certain requirements to the input image.

When using a metric module, failure to meet the specified image requirements (e.g. image size), will lead to an error during execution of the analysis script.

Generic metric module scripting support

Functions, parameters and outputs that are common to all metric modules are listed in the table.

| Name | Type | Usage and meaning |
|------------|------|-------------------|
| Parameters | | |

| Name | Type | Usage and meaning |
|-----------------|------------------|--|
| pEdgeInsets | Array of 4 Float | <p>Usage: metric.pEdgeInsets = [left, right, top, bottom];</p> <p>The edgeInset is used to inform the metric that when analyzing a test element, the image ROI should be modified from the boundary of the test element. For example, a uniformity measure should typically not use the image all the way up the the edge of an element. Each metric module has a default edgeInset, that is suitable in most cases. The parameters are physical distances (mm) from the edges. Positive values will move the edge inside the element, while negative values will mode the edge outside the element.</p> <p>If the array has fewer than 4 values, then the missing values are interpreted as zero. For example, metric.pEdgeInsets = []; means the exact test element boundary should be used.</p> |
| Outputs | | |
| Version | String | The version of the metric module. This is different from the system version. |
| oROICoordinates | String | <p>The coordinates of the region of the image that was analyzed (ROI). The coordinates are the distances from the left-top corner of the overall image to the left-top corner of the ROI; and the width and height of the ROI. All are in mm units, and are listed as:</p> <p>(left, top, width, height)</p> |

Mottle metric module

Image requirements

| | | |
|-----------------------|-------------|---|
| Color space | Device RGB | Must be consistent with the system OECF for conversion to reflectance and CIE Y. |
| Scan resolution (dpi) | 1200 | For ISO 24790 compliance. |
| | TBD | |
| Minimum size (pixels) | 1200 x 1200 | For ISO 24790 compliance. Excess image will be ignored, and only central part being analyzed. |
| | TBD | |

Scripting support—parameters and outputs

| Name | Type | Usage and meaning |
|-----------------------------|-------------|---|
| QQMottle | Class name | |
| Parameters | | |
| pEdgeInsets | See generic | Default = [1.5, 1.5, 1.5, 1.5]. |
| pIso24790Conformance | Boolean | Default = true. If true, the calculation will be performed to achieve maximal conformance with ISO 24790, possibly by overriding other parameter values. |
| pEnforceIsoMinimumImageSize | Boolean | Default = true. If true, calculation will fail if the image size is less than specified by ISO 24790. If false, the image size can be smaller, and the ISO requirement of at least 9 by 9 tiles is also ignored. |
| pUseUnlimitedTileCount | Boolean | Default = true. If false, the number of tiles will be restricted to 9 x 9 (consistent with the minimum required by ISO 24790 graininess). If set to false, there could be significant “gaps” on the image that do not contribute to the graininess calculation. |

| Name | Type | Usage and meaning |
|------------------|-------|--|
| Outputs | | |
| oCIEYAverage | Float | The average CIE Y of the original image. |
| oCIEYSD | Float | The standard deviation of CIE Y of the original image. |
| oDensity | Float | The average optical density of the image. |
| oFilteredAverage | Float | The average CIE Y of the filtered image. |
| oFilteredSD | Float | The standard deviation of CIE Y of the filtered image. |
| mottle_iso | Float | Output defined by ISO 24790. Not yet supported. |
| oMottle | Float | The calculated mottle value. |
| oMottleSE | Float | Estimate of the standard error of mottle. The 95% confidence interval is: $\text{oMottle} \pm 1.96 \cdot \text{oMottleSE}$. |

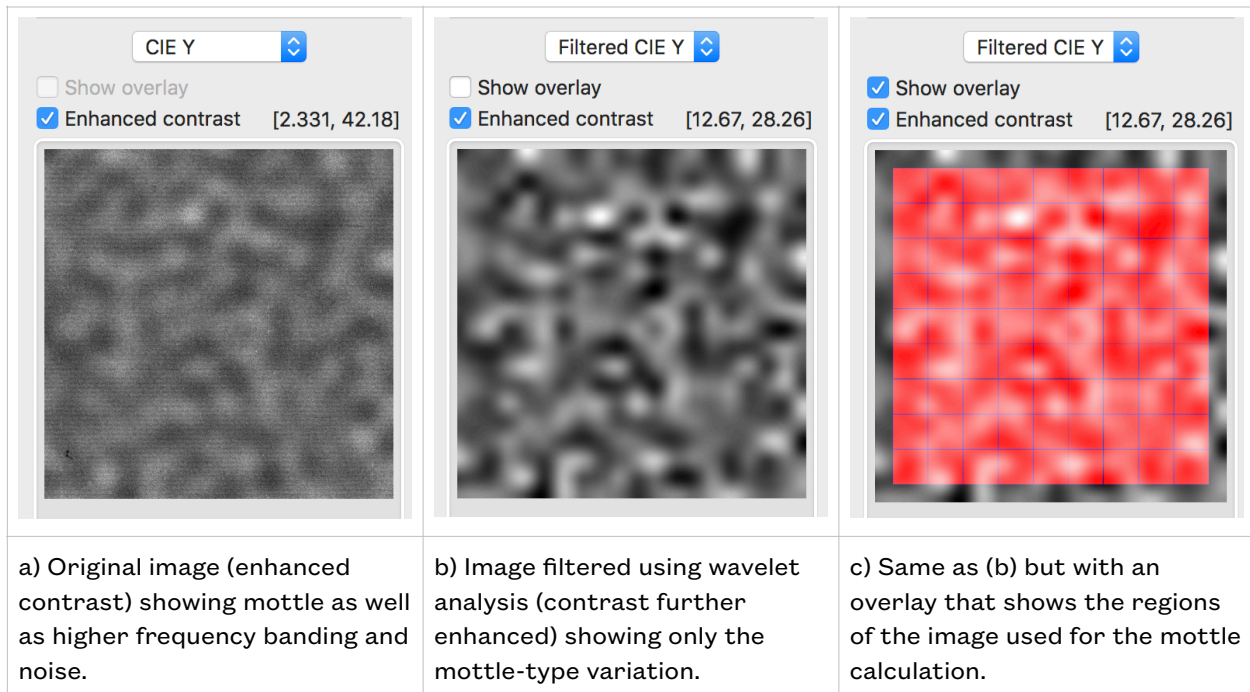
Analysis method

The mottle metric uses wavelet filtering of the image to exclude variations in the image that should not be counted towards measurements of mottle. For example, the filtering excludes variation at very small spatial scales, which is better characterized as noise or graininess.

By default, the mottle metric will perform an analysis that attempts to be consistent with ISO 24790 mottle (see conformance section for more information).

The RGB image is first converted into CIE Y, using the system OECF, and the remaining analysis is based on the image with pixel values that represent CIE Y.

Figure mottle.1



The ISO 24790 mottle algorithm is defined based on a specific image size and sampling resolution (1200x1200 pixels at 1200dpi). For reference, spatial scales and frequencies for wavelet analysis of such an image are given in table mottle.1.

We use a nomenclature consistent with Matlab, where level = 1 refers to the first decomposition step of the wavelet transformation. Thus, level 1 contains *detail* for the pixel-to-pixel differences (corresponding to a “wavelength” of 2 pixels = 0.0423mm or a frequency of 23.6cy/mm), and also contains the *approximation* image with lower frequencies.

Table mottle.1. 9-level wavelet decomposition of 1200dpi image.

| Level | Detail frequency (cy/mm) | Detail spatial scale (mm) (half wavelength) | Approximation frequency limit (cy/mm) | Approximation scale (mm) | ISO 24790 scale level |
|-------|--------------------------|---|---------------------------------------|--------------------------|-----------------------|
| 1 | 23.6220 | 0.021 | 11.811 | 0.042 | 8 |
| 2 | 11.8110 | 0.042 | 5.906 | 0.085 | 7 |
| 3 | 5.9005 | 0.085 | 2.950 | 0.169 | 6 |
| 4 | 2.9526 | 0.169 | 1.476 | 0.339 | 5 |
| 5 | 1.4763 | 0.339 | 0.738 | 0.677 | 4 |
| 6 | 0.7382 | 0.677 | 0.369 | 1.355 | 3 |

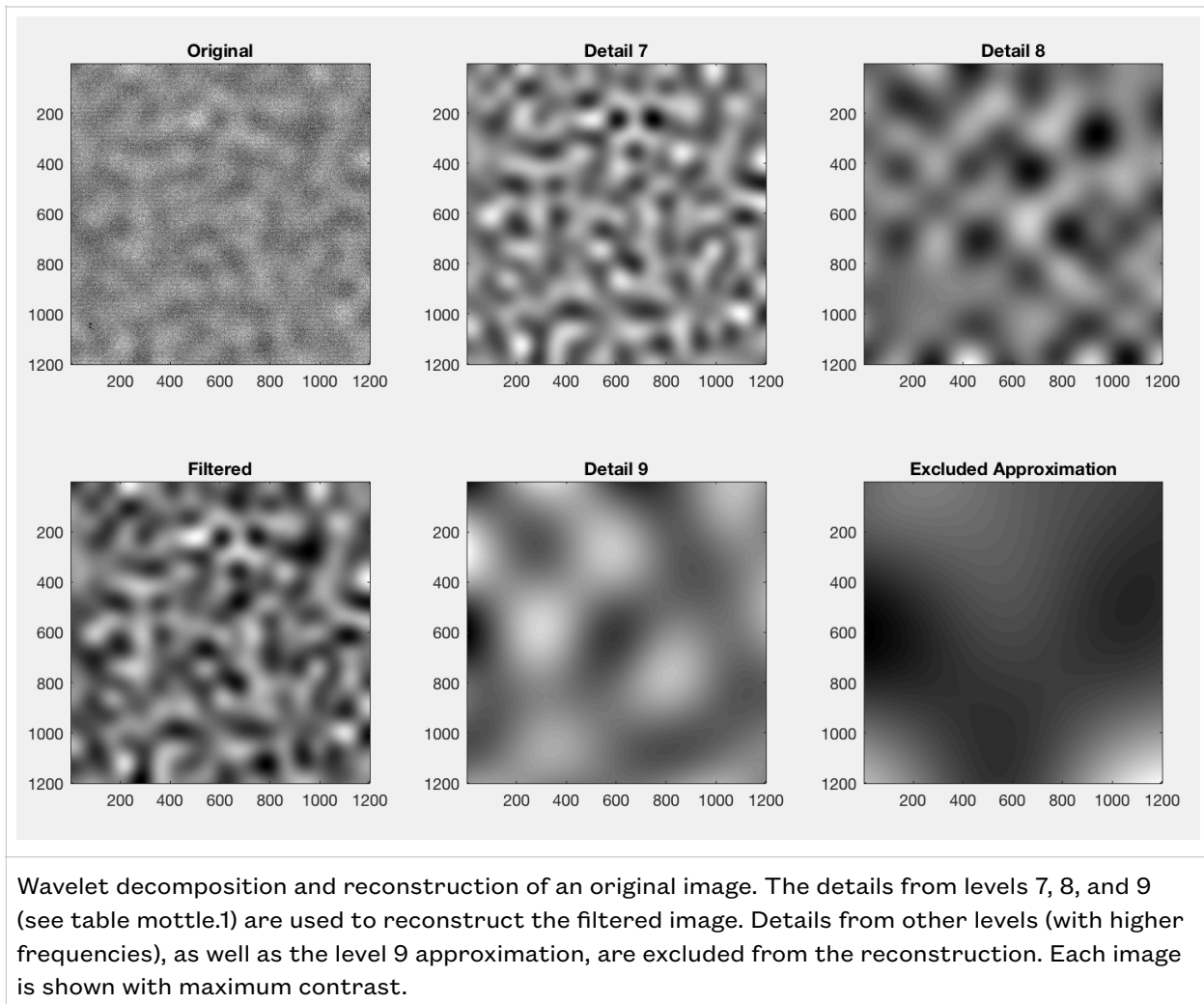
| Level | Detail frequency (cy/mm) | Detail spatial scale (mm) (half wavelength) | Approximation frequency limit (cy/mm) | Approximation scale (mm) | ISO 24790 scale level |
|-------|--------------------------|---|---------------------------------------|--------------------------|-----------------------|
| 7 | 0.3691 | 1.355 | 0.185 | 2.709 | 2 |
| 8 | 0.1846 | 2.709 | 0.092 | 5.417 | 1 |
| 9 | 0.0923 | 5.417 | 0.046 | 10.834 | 0 |

By retaining only certain components of the wavelet decomposition, we can retain the specific image variation that we wish to quantify. For example, retaining only the components highlighted yellow in the table yields an image with spatial variations on a scale from around 1 to 6mm. This filtering is illustrated in the figure mottle.2, where the image labeled “Filtered” is the sum of the three detail images.

Having filtered the image to exclude defects other than mottle, the remaining analysis is mainly a calculation of the standard deviation of CIE Y within the image. The filtered image is split into tiles, as shown in figure mottle.1.c. For each tile independently, the variance of CIE Y is calculated. Notice that, even though low frequencies have already been filtered out (see figure mottle.2 “Excluded Approximation”), considering each tile independently will again prevent lower frequencies from contributing to the mottle calculation, and will in practice potentially reduce the standard deviation a bit further, compared to the standard deviation calculated across the entire filtered image.

Given the variance of CIE Y for each tile, the average variance is calculated, and the square root of this average is reported as *mottle_iso*. Currently, this is reported as *oMottle*, and depending on how the discrepancy with ISO 24790 is resolved this may be renamed as *mottle_iso*.

Figure mottle.2



Conformance to ISO 24790

The current implementation is intended to replicate the ISO 24790 mottle algorithm. We are not aware of any discrepancies between the current implementation and the ISO 24790 algorithm, as far as the details of the ISO 24790 algorithm have been documented. Matlab source for algorithm implemented by QiQ has been provided to the ISO 24790 team for their review.

However, it has not been yet been possible to directly confirm compliance with the standard, and in fact certain results seem to indicate there may be a discrepancy. Due to this discrepancy this metric module reports the result as *oMottle*, rather than *mottle_iso*. Measurements on a ISO 24790 conformance test chart yields $oMottle = 1.18$ rather than the value 2.83 ± 0.04 expected from the standard.

Extended analysis options

The metric module has parameters that allow you to enhance or optimize the measurements for specific purposes.

Incomplete

Influence of average density on the visual perception of mottle

Incomplete

Graininess metric module

Image requirements

| | | |
|-----------------------|------------|--|
| Color space | Device RGB | |
| Scan resolution (dpi) | 1200 | |
| Minimum size (pixels) | 600 x 600 | |

Scripting support—parameters and outputs

| Name | Type | Usage and meaning |
|-----------------------------|-------------|---|
| QQGraininess | Class name | |
| Parameters | | |
| pEdgeInsets | See generic | Default = [1.5, 1.5, 1.5, 1.5]. |
| pIso24790Conformance | Boolean | Default = true. If true, the calculation will be performed to achieve maximal conformance with ISO 24790, possibly by overriding other parameter values. |
| pEnforceIsoMinimumImageSize | Boolean | Default = true. If true, calculation will fail if the image size is less than specified by ISO 24790. If false, the image size can be smaller, and the ISO requirement of at least 9 by 9 tiles is also ignored. |
| pUseUnlimitedTileCount | Boolean | Default = true. If false, the number of tiles will be restricted to 9 x 9 (consistent with the minimum required by ISO 24790 graininess). If set to false, there could be significant “gaps” on the image that do not contribute to the graininess calculation. |
| Outputs | | |
| oCIEYAverage | Float | The average CIE Y of the original image. |
| oCIEYSD | Float | The standard deviation of CIE Y of the original image. |
| oDensity | Float | The average optical density of the image. |

| Name | Type | Usage and meaning |
|------------------|-------|--|
| oFilteredAverage | Float | The average CIE Y of the filtered image. |
| oFilteredSD | Float | The standard deviation of CIE Y of the filtered image. |
| graininess_iso | Float | Output defined by ISO 24790. Not yet supported. |
| oGraininess | Float | The calculated graininess value. |
| oGraininessSE | Float | Estimate of the standard error of graininess. The 95% confidence interval is: $\text{oGraininess} \pm 1.96 \cdot \text{oGraininessSE}$. |

Analysis method

The analysis is in many ways similar to mottle, except that noise at smaller spatial scales are targeted.

The graininess metric uses wavelet filtering of the image to exclude variations in the image that should not be counted towards measurements of graininess. By default, the graininess metric will perform an analysis that attempts to be consistent with ISO 24790 mottle (see conformance section for more information).

The RGB image is first converted into CIE Y, using the system OECF, and the remaining analysis is based on the image with pixel values that represent CIE Y.

The image is filtered with wavelet to level 6, and only the detail components at levels 5 and 6 are retained. The spatial frequency content that remains in the image are highlighted in yellow in table graininess.1.

Table graininess.1. 6-level wavelet decomposition of 1200dpi image.

| Level | Detail frequency (cy/mm) | Detail spatial scale (mm) (half wavelength) | Approximation frequency limit (cy/mm) | Approximation scale (mm) | ISO 24790 scale level |
|-------|--------------------------|---|---------------------------------------|--------------------------|-----------------------|
| 1 | 23.6220 | 0.021 | 11.811 | 0.042 | 5 |
| 2 | 11.8110 | 0.042 | 5.906 | 0.085 | 4 |
| 3 | 5.9005 | 0.085 | 2.950 | 0.169 | 3 |
| 4 | 2.9526 | 0.169 | 1.476 | 0.339 | 2 |
| 5 | 1.4763 | 0.339 | 0.738 | 0.677 | 1 |
| 6 | 0.7382 | 0.677 | 0.369 | 1.355 | 0 |

The filtered image is split into square tiles of size 1.27mm^1 . For each tile independently, the variance of CIE Y is calculated. Given the minimum image size, there will be at least 81 tiles, and the average variance is calculated. The square root of this average is reported as `graininess_iso`. Currently, this is reported as `oGraininess`, and depending on how the discrepancy with ISO 24790 is resolved this may be renamed as `graininess_iso`.

Conformance to ISO 24790

The current implementation is intended to replicate the ISO 24790 graininess algorithm. We are not aware of any discrepancies between the current implementation and the ISO 24790 algorithm, as far as the details of the ISO 24790 algorithm have been documented.

However, it has not been yet been possible to directly confirm compliance with the standard, and in fact **certain results seem to indicate there may be a discrepancy**. Due to this discrepancy the graininess metric reports the result as `oGraininess`, rather than `graininess_iso`.

¹ ISO 24790 specifies only that the tile size be at least 1.27mm . Using a larger tile size—although consistent with the specified algorithm—can lead to significantly different (typically larger) graininess values by including lower frequency variations.

VBS (Visual Banding and Streaking) metric module

Image requirements

| | | |
|-----------------------|------------|--|
| Color space | Device RGB | |
| Scan resolution (dpi) | ≥ 600 | |
| Minimum size (mm) | 100 x 20 | For “horizontal” analysis which measures defects running parallel to the vertical direction. |
| | 20 x 100 | For “vertical” analysis which measures defects running parallel to the horizontal direction. |

Scripting support—parameters and outputs

| Name | Type | Usage and meaning |
|----------------------|-----------------|--|
| QQVBS | Class name | |
| Parameters | | |
| pEdgeInsets | See generic | Default = [3, 3, 3, 3]. |
| pDoHorizontal | Boolean | Default = true. Carry out analysis of a horizontal profile, analyzing defects (bands, streaks) that run parallel to the vertical direction. |
| pDoVertical | Boolean | Default = true. Carry out analysis of a vertical profile, analyzing defects (bands, streaks) that run parallel to the horizontal direction. |
| pHorizontalSkewAngle | Float (radians) | Optional input : Used only if pUseAutomaticDeskew is false. The skew angle is counter-clockwise from horizontal. For example, if value is 0.003 then the profile is calculated along a direction that is 0.003 radians counter-clockwise from horizontal. Output : If pUseAutomaticDeskew is true, then the detected skew angle will be given as an output. |

| Name | Type | Usage and meaning |
|---|-----------------|---|
| pIso24790Conformance | Boolean | Default = false. If true, the calculation will be performed to achieve maximal conformance with ISO 24790, possibly by overriding other parameter values. This will force pUseEdgeRobustness = false. |
| pUseEdgeRobustness | Boolean | Default = true. Defects that are very close to the edges of the image will not be taken into account. This increases repeatability, since potential, artificial defects, created as part of the image processing, will not be taken into account. |
| pUseAutomaticDeskew | Boolean | Default = true. The fiducial marks will be used to determine an affine transformation (rotation, shear, scale, translation) and from that pHorizontalSkewAngle and pVerticalSkewAngle are automatically determined. |
| pVerticalSkewAngle | Float (radians) | Equivalent to pHorizontalSkewAngle. |
| Outputs | | |
| oCieX, oCieY, oCieZ, oCieLs, oCieAs, oCieBs | Float | Standard CIE outputs. |
| oVbsHorizontal | Float | VBS value calculated from profile in the horizontal direction, measuring defects running in the vertical direction. This is consistent with the ISO 24790 algorithm. |
| oVbsVertical | Float | VBS value calculated from profile in the vertical direction, measuring defects running in the horizontal direction. This is consistent with the ISO 24790 algorithm. |
| oVbsHorizontalHS | Float | A “high sensitivity” variant of oVbsHorizontal. This is recommended for very high quality images, but will also increase the sensitivity to noise in the image which is not in the form of streaks or bands (e.g. mottle). |
| oVbsVerticalHS | Float | Equivalent to pVbsHorizontalHS. |

Analysis method

This analysis follows that of the ISO 24790 banding metric, and is described in greater detail in a 2010 paper in the Journal of Electronic Imaging².

The RGB image is first converted into CIE Y, using the system OECF, and the remaining analysis is based on the image with pixel values that represent CIE Y. Independent analysis of the horizontal and / or vertical direction can be performed.

For a horizontal analysis a horizontal profile of CIE Y is calculated, by averaging the image along the vertical direction. The profile is converted to a CIE L* profile, and individual “defects”, in the form of deviations from the L* average, are detected. The magnitude of those defects are added up using a “tent-pole summation” rule, such that the worst defects have relatively greater impact than smaller defects.

Practical use of the VBS measurement

The VBS metric was designed to correlate with human visual assessment of uniformity of an A4 (or US Letter) sized print, where almost the entire area of the print was evaluated. The human visual impression of uniformity is significantly impacted by both the horizontal and vertical dimensions of the image. Even though the VBS measurement can be carried out on smaller images, the measurement is more susceptible to two-dimensional noise (e.g. mottle) when the image is smaller. For these reasons, it is highly recommended that the measurement be carried out on large area images, such as with the “W1.1 Macro.v4” test chart.

Conformance to ISO 24790

The implementation is intended to replicate the ISO 24790 banding algorithm, however, no guarantee can be given as to compliance with the ISO 24790 standard.

² “Tent-pole spatial defect pooling for prediction of subjective quality assessment of streaks and bands in color printing,” D. René Rasmussen, *Journal of Electronic Imaging* 19(1), 011017 (Jan-Mar 2010).

Lines metric module

Image requirements

| | | |
|-----------------------|---|--|
| Color space | Device RGB | |
| Scan resolution (dpi) | 1200 | This is the recommended and minimum resolution. |
| Minimum size (mm) | 1 in direction parallel to the line | |
| Line orientation | Horizontal or vertical | Small deviations are acceptable and will be measured; compensation for small deviations are performed. |
| Colors | Black lines on white or colored surround. | |

Scripting support—parameters and outputs

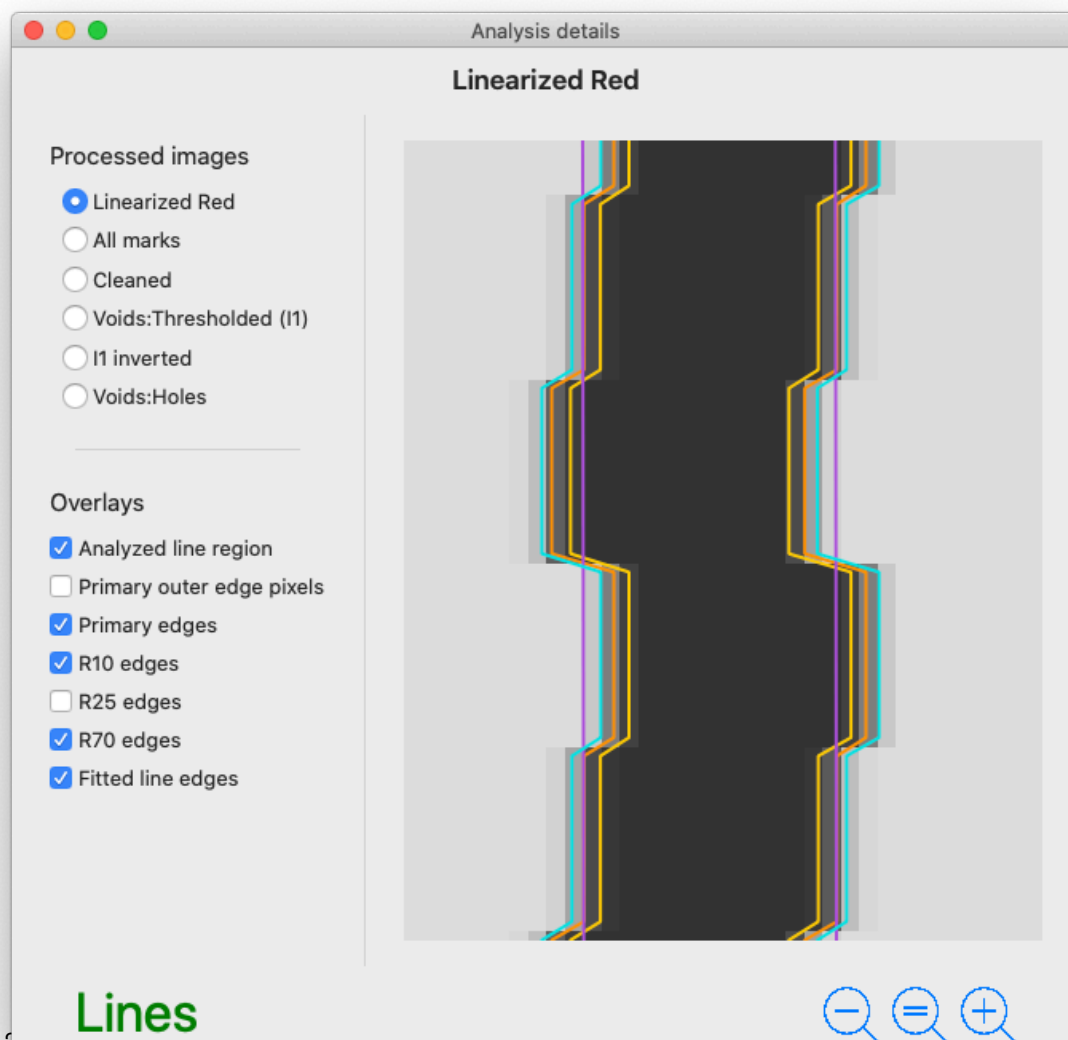
| Name | Type | Usage and meaning |
|--------------------|-------------|--|
| QQLines | Class name | |
| Parameters | | |
| pEdgeInsets | See generic | Default = [0, 0, 0, 0]. |
| pContrastMethod | String | Options are: MaxContrast (the default) RedChannel GreenChannel BlueChannel |
| pLineAngle | Float | Allowed values: 0 (horizontal line) — default. 90 (vertical line) |
| pNominalWidth (mm) | Float | The nominal (expected) line width. Default = 0. |

| Name | Type | Usage and meaning |
|-------------------------------|--------|---|
| pEMMinimumDiameter (micron) | Float | For measurement of extraneous marks: the minimum equivalent diameter for extraneous marks. Default = 100 (per ISO 24790) |
| pEMMaximumDistance (micron) | Float | For measurement of extraneous marks: the maximum distance from the line edge. Default = 500 (per ISO 24790) |
| | | |
| Outputs | | |
| oChannel | String | The scanner channel that was used for the measurement (useful when pContrastMethod = MaxContrast). |
| oRmin | Float | The minimum (“black”) reflectivity as determined during the measurement. |
| oRmax | Float | The maximum (“white”) reflectivity as determined during the measurement. |
| oLineWidth (mm) | Float | The measured line width. |
| oLineWidthUncorrected (mm) | Float | The “raw” measured line width, not compensated for small tilts away from horizontal or vertical. |
| oLineAngle (degrees) | Float | The measured line angle (degrees counter-clockwise from horizontal). |
| oLineDarkness (\sqrt{mm}) | Float | Line darkness (see ISO 24790 section 5.3.4) |
| oBlurrinessLeft (mm) | Float | Blurriness of the leading edge of the line (top or left). (See ISO 24790 section 5.3.5) |
| oBlurrinessRight (mm) | Float | Blurriness of the trailing edge of the line (bottom or right). (See ISO 24790 section 5.3.5) |
| oBlurriness (mm) | Float | Average edge blurriness. (See ISO 24790 section 5.3.5) |
| oRaggednessLeft (mm) | Float | Raggedness of the leading edge of the line. (See ISO 24790 section 5.3.6) |
| oRaggednessRight (mm) | Float | Raggedness of the trailing edge of the line. (See ISO 24790 section 5.3.6) |
| oRaggedness (mm) | Float | Average edge raggedness. (See ISO 24790 section 5.3.6) |
| oVoidsRatio (unitless) | Float | Line voids. (See ISO 24790 section 5.3.7) |

| Name | Type | Usage and meaning |
|------------------------|-------|--|
| oExtraMarks (unitless) | Float | Extraneous marks measurement. (See ISO 24790 section 5.3.8) |

Analysis method

This analysis follows that of the ISO 24790 closely, but not exactly. In particular, the scanner MTF compensation is not performed. The purpose of the MTF compensation is to obtain greater agreement between measurements performed with scanning devices with a MTF that deviates from the ideal 1200 dpi scan MTF. For best agreement with ISO24790, it is recommended that images are scanned at 1200 dpi; however, better sensitivity may be obtained with higher resolutions.



The analysis detail view illustrates the intermediate results, for example:

Specifying the nominal line orientation (horizontal or vertical)

By default the Lines module assumes the line is horizontal. A vertical line must be indicated by setting the parameter `pLineAngle = 90`. When working with a test chart, this is most conveniently done, by assigning the line angle in the chart script—then in the analysis script copy the line angle from the element to the Lines metric object. This is illustrated by the system scripts for lines chart and analysis. See also the [section on the ChartLine element](#).

Coverage metric module

Image requirements

| | | |
|-----------------------|------------|--|
| Color space | Device RGB | |
| Scan resolution (dpi) | ≥ 300 | |
| Minimum size (mm) | N/A | Module requires input of 3 image elements: target, reference, and white. |

Scripting support—parameters and outputs

| Name | Type | Usage and meaning |
|-------------------|-------------|--|
| QQCoverage | Class name | |
| Parameters | | |
| pContrastMethod | String | Options are: MaxContrast (the default) Gray RedChannel GreenChannel BlueChannel |
| pEdgeInsets | See generic | Default = [1, 1, 1, 1]. |
| pUseLinearizedRGB | Boolean | Default = true. |
| Outputs | | |
| oChannel | String | Value will be one of: Gray RedChannel GreenChannel BlueChannel according to how calculation was performed (see contrastMethod parameter). |

| Name | Type | Usage and meaning |
|------------------|---------|--|
| olsReversed | Boolean | Indicates whether the calculated used reversed contrast. Reversed contrast means that the “reference” and “white” elements were switched. 0, false: Contrast was not reversed. 1, true: Contrast was reversed. |
| oFractionRemoved | Float | The fraction coverage removed from the target element. Range is [0, 1]. |
| pReferenceLevel | Float | The average image level of the reference element. |
| oTargetLevel | Float | The average image level of the target element. |
| oWhiteLevel | Float | The average image level of the white element. |

Analysis method

Three image element must be provided to the analyzeCoverage function: A *target* element, a *reference* element, and a *white* element. Normally, the reference element contains 100 covered image, the white element contain fully non-covered image, and the coverage of the target element will calculated based on comparison to the other two elements.

The algorithm will automatically determine if “reverse contrast” should be used, where the coverages of the reference and the white elements are switched. `olsReversed` indicates whether the calculation used reversed contrast.

By default, the device RGB values are first linearized, using the system OECF. This will lead to slightly different results than if the raw device RGB values were used, and the results will be more consistent from one scanner to another. When linearized RGB values are used, the levels are reported in the range [0, 1]. The input parameter `pUseLinearizedRGB` can be used to select use of device RGB.

The analysis is done either using one of the three red, green, and blue channels, or by using a gray level calculated as a weighted sum of the RGB channels. The input parameter `pContrastMethod` can be used to control this. When `pContrastMethod` = “MaxContrast” the algorithm will automatically pick the R, G, or B channel which has greatest contrast; for example, for a cyan image, it will pick the red channel.

Voids metric module

Image requirements

| | | |
|-----------------------|--|--|
| Color space | Device RGB | |
| Scan resolution (dpi) | ≥ 600 | Recommended, but can be run in test mode at lower resolutions (see <code>pMinimumImageResolution</code>). |
| Minimum size (mm) | Target must be at least 161mm ² , and at least 12.7mm on each side. | Module requires input of 3 image elements: target, solid, and white. There are no size requirements to the solid and white elements. |

Scripting support—parameters and outputs

| Name | Type | Usage and meaning |
|-----------------------------|-------------|---|
| QQVoids | Class name | |
| Parameters | | |
| pAlgorithm | String | Default = "ISO24790" Options: "ISO24790" "VisualVoidsA1" |
| pBackgroundThreshold | Float | Default = 20 (relative reflectance). Only used by VisualVoidsA1 algorithm. |
| pContrastMethod | String | Options are: MaxContrast (the default) RedChannel GreenChannel BlueChannel |
| pEdgeInsets | See generic | Default = [1, 1, 1, 1]. |
| pEnforceIsoMinimumImageSize | Boolean | Default = true. ISO requires the region examined be at least 161mm ² , and at least 12.7mm on each side. |

| Name | Type | Usage and meaning |
|--------------------------------|--------------------------|--|
| pIso24790Conformance | Boolean | Default = false. This will enforce the following: <ul style="list-style-type: none"> • minimum image dimensions (see elsewhere); • scan resolution 1200 dpi; • only voids with an area at least that of a circle with 100 micrometer diameter ($7850\mu\text{m}^2$) will be taken into account. |
| pMinimumAdjustedCount | Float | Default = 0.2. Only for the VisualVoidsA1 algorithm. The unit is such that a single void of diameter $100\mu\text{m}$, and with relative reflectance 100, has value = 1.0. With relative reflectance 40 (= the ISO threshold), the adjustedCount is 0.4, so the default value, 0.2, accepts voids half the “size” of the ISO threshold. |
| pMinimumImageResolution | Int [dpi] | Default = 600 dpi. Can be reduced e.g. for process testing purposes. |
| pReflectanceThreshold | Float | Default = 40 (as specified by the ISO standard). Used only by the ISO24790 algorithm. |
| pSeedThreshold | Float | Default = 30 (relative reflectance). Only for the VisualVoidsA1 algorithm. At least one pixel within a region must exceed this level, in order for the region to qualify as a void. |
| pVoidMinimumEquivalentDiameter | Float, [μm] | Default = 70. ISO 24790 specifies $100\mu\text{m}$, corresponding to an area of $\pi(100/2)^2 \mu\text{m} = 7854\mu\text{m}^2$. |
| Outputs | | |
| oChannel | String | Value will be one of: RedChannel GreenChannel BlueChannel according to how calculation was performed (see contrastMethod parameter). |
| oDeviceSolidLevel | Int | The solid (device) level corresponding to the solid reflectance level. |
| oDeviceThreshold | Int | The threshold below which pixels are considered void (if part of sufficiently large void). This is reported in device level, calculated from pReflectanceThreshold using the linearized red, green, or blue. |
| oDeviceWhiteLevel | Int | The white (device) level corresponding to the substrate reflectance level. |

| Name | Type | Usage and meaning |
|--------------------------------|-------------|--|
| oISOLargeAreaVoid | Float | The large area void metric as defined by ISO 24790. This is the ration between the total area of the voids, and the area of the region examined. |
| oROIArea | Float | [mm ²] The area of the region of interest that was examined. |
| oTotalVoidArea | Float | [μm ²] The sum of the areas of the voids. |
| oVoidAdjustedCount | Float | For algorithm VisualVoidsA1 only. An adjust “count” of voids. The measurement is such that a void of diameter 100μm, and with relative reflectance 100, contributes 1.0 unit. |
| oVoidAreaAverage | Float | [μm ²] The average area of the voids. |
| oVoidAreaSD | Float | [μm ²] The standard deviation of area of the voids. |
| oVoidCount | Int | The count of voids for algorithm ISO24790 only. |
| oVoidEquivalentDiameterAverage | Float, [μm] | The average of the equivalent diameters calculated from the voids’ areas. |
| oVoidEquivalentDiameterSD | Float, [μm] | The variance of the equivalent diameters calculated from the voids’ areas. |
| oVoidEquivalentDiameterMaximum | Float, [μm] | The maximum of the equivalent diameters calculated from the voids’ areas. |
| oVoidEquivalentDiameterMinimum | Float, [μm] | The minimum of the equivalent diameters calculated from the voids’ areas. |

Analysis method when `pIso24790Conformance = true`

Three image element must be provided to the `analyzeCoverage` function: A *target* element, a *solid* element, and a *white* element. The solid element should contain 100 covered image, the white element contain fully non-covered image. The target element will be examined for voids (light spots on the darker background).

The analysis is done using only one of the three channels, either red, green, or blue. The input parameter `pContrastMethod` can be used to control this. When `pContrastMethod = “MaxContrast”` the algorithm will automatically pick the R, G, or B channel which has greatest contrast; for example, for a cyan image, it will pick the red channel.

The calculation is according to the ISO 24790 standard.

Analysis method when pAlgorithm = “VisualVoidsA1”

The main purpose of this analysis is to provide an overall measure of voids that correlates well with an overall visual perception of voids, taking into account also small voids that may not be recognized as voids by the ISO 24790 algorithm.

This algorithm has the following components:

- Detection of voids within the image.
- For each void, determination of the boundary of the void (that is, the set of image pixels that constitute the void).
- For each void, calculation of a measure of the severity of the void, using an “adjusted count” unit.
- A “summation” of all the voids, given the severities, into an overall measure, the oVoidAdjustedCount.

CIEColor metric module

Image requirements

| | | |
|-----------------------|------------|--|
| Color space | Device RGB | |
| Scan resolution (dpi) | >= 600 | |
| Minimum size (mm) | N/A | |

Scripting support—parameters and outputs

| Name | Type | Usage and meaning |
|---|-------------|--|
| QQCIEColor | Class name | |
| Parameters | | |
| pEdgeInsets | See generic | Default = [1, 1, 1, 1]. |
| Outputs | | |
| oCieX, oCieY, oCieZ, oCieLs, oCieAs, oCieBs | Float | Estimated CIE color values, calculated using the OECF and assumption that the device produces sRGB values. |
| oLinearR, oLinearG, oLinearB | Float | The average, linearized RGB values calculated using the OECF. In current version, these are raw RGB values, not linearized. |
| oDensity | Float | The “gray density”, calculated from oCieY. oDensity = $-\log_{10}(\text{oCieY}/100)$. |
| oRCDensity | Float | “Red complementary density” calculated from the red scanner device reflectivity. This may be useful to create correlations to cyan ink density. |
| oGCDensity | Float | “Green complementary density” calculated from the green scanner device reflectivity. This may be useful to create correlations to magenta ink density. |

| Name | Type | Usage and meaning |
|------------|-------|---|
| oBCDensity | Float | “Blue complementary density” calculated from the blue scanner device reflectivity. This may be useful to create correlations to yellow ink density. |

Analysis method

Standard ink densities cannot be calculated from the scanner RGB values, since the scanner R, G, B filters’ spectral responses are not consistent with the standard for ink densities. However, it is usually possible for a given ink formulation (with a given spectral reflectivity spectrum) to establish a good correlation between device measurements and ink densities, for example between red channel values and cyan ink density. To facilitate this, this module provides “complementary density” outputs, calculated as follows.

Green channel complementary density: The raw device green values are converted, pixel-by-pixel, to linearized green values, using the scanner OECF, scaled to [0, 1]. This yields device green reflectivity pixel values. The green reflectivity pixel values are averaged over the image, yielding the average device green reflectivity: R_g . The green complementary density is calculated as the negative log base 10 of R_g :

$$oGCDensity = -\log_{10}(R_g)$$

oRCDensity and oBCDensity are calculated similarly.

Analysis Jobs and Data Flow

This section talks about the *data* directly related to the analysis of images: the input data (e.g. scripts), the intermediate data generated during image analysis (e.g. positions of fiducial marks, line edges) and the final results calculated by metric modules. The section explains the flow of such data from the start of an analysis until it is used to create final reports (e.g. in Excel).

Input to the analysis—analysis jobs

The input to analysis of an image consists of these parts:

- A file URL to a scanner device OECF.
- A file URL for a TIFF image, optionally with an ROI specifier.
- A file URL for an analysis script.
- Optionally, a file URL for a chart script.

These parts combined specify an *analysis job*. A job can be created interactively through the user interface, or jobs can be created for batch processing of image files.

Intermediate analysis output—IAOB

An analysis may generate intermediate output which can be used to gain further insight into the details of the analysis.

As an example, consider analysis of an image with a test chart containing multiple lines. For each line analysis, the intermediate output may consist of the image crop that was used, several processed images, as well as line edge profiles. This data can be viewed in near-real time during the analysis, or after the analysis has completed.

The intermediate output is stored on the file system (in a directory or bundle), and can be deleted without impact on the final analysis results. We refer to it as IAOB (short for intermediate analysis output bundle).

Raw analysis output—JRAF

The raw analysis output are the results of the analysis, such as mottle values, line widths, etc. But for completeness, it also includes the input to the analysis.

The output is stored on the file system in JSON format, in one file per analysis. This is where output data is stored immediately after the analysis. The JSON files use a specific structure which we call JRAF (short for JSON Raw Analysis File). The JRAF files are stored inside QiQ/projects/<projectName>/data.

Even if final reports (e.g. Excel) are generated later on, it may be wise to keep the raw analysis output files, since it is easily portable and uses a format that is robust against future changes of the software.

Analysis data base (QCD)

It is difficult to search and process the raw analysis output directly. Therefore, it can be imported into a data base. The data base is here called *QCD* and its structure is described elsewhere in this document.

QCD allows results to be efficiently searched, organized and exported for other uses. For example, the results from multiple related analyses can be combined in order to create compound reports.

Intermediate report files—JRAF and compound JRAF

Final reports (e.g. Excel) are created from intermediate report files. These intermediate files use the same JSON/JRAF format and structure, as is used for the raw analysis output. However, an intermediate report file may contain data from multiple analyses (for compound reports) and is then called a *compound JRAF*.

The intermediate report files can be generated from the QCD analysis data base.

Data Flow

Here are the possible data flows (where a final report may be an Excel document):

Input —> QiQ —> IAQB, JRAF

JRAF —> QCD

JRAF —> Final report

QCD —> JRAF or Compound JRAF —> Final report

IAQB —> QiQ Viewer

The QCD Database

This chapter is about the database used internally by QiQ. It will only be of interest for a user who wishes to develop custom tools for processing results, for example to transfer results to a custom external database system, or to generate different kinds of reports than currently supported by QiQ.

QiQ stores all results internally using an SQLite relational database. The data model is designed with extensibility in mind, in particular it supports new measurement modules without any modification.

Description of the data model entities

Overview of essential entities.

| Fundamental entities | | | |
|----------------------|--|----------------|--|
| Entity name | Description | Key attributes | Description |
| QCDSample | A physical sample (e.g. one hardcopy) which has been captured (one or more times) into an image. | identifier | A string of text that identifies the sample. The combination of (identifier, copyCount) <u>uniquely identifies the sample</u> ; however, the cannot be enforced by QiQ, and is the responsibility of the user. Note that the identifier can be composed of multiple parts according to user needs. |
| | | copyCount | Integer used to identify a specific sample within a series of otherwise “identical” samples. For example, in the case of an experiment that generates multiple samples with identical preparation. |
| | | sourceID | |
| | | images | List of images (QCImage) that have been captured of the sample. |
| QCDDevice | An image acquisition device, such as a flatbed scanner. | make | |
| | | model | |
| | | serialNumber | |
| | | nickname | |
| | | location | |
| QCDSystemOperator | A person who operates the QiQ system. | firstName | |
| | | lastName | |
| QCDOECF | An OECF characterizing an image acquisition device. | filepath | Location of the file on the computer file system. |
| | | dateModified | Date the OECF was last modified. |
| QCModule | A module that | name | The name of the module. |

| Fundamental entities | | | |
|----------------------|---|-------------------|---|
| Entity name | Description | Key attributes | Description |
| | performs a specific kind of image analysis (for example the Mottle module). | version | The version of the module. |
| QCDChartElement | An element (region of interest) of a test chart, which has been submitted to analysis by a module. | identifier | A string that uniquely identifies the element within the test chart. |
| | | | |
| Complex entities | | | |
| QCDSession | One session of use of QiQ, started, for example, by launch of the program or login by an operator. | dateStarted | Date and time session was started. |
| | | dateEnded | Date and time session was ended. |
| | | software | Name and version of QiQ software. |
| QCDImage | Reference to a digital image that has been subjected to analysis. | filePath | The file system path. |
| | | dateModified | The file modification date. |
| | | acquisitionDevice | The device (QCDDevice) that captured the image. This is not used to uniquely identify image. |
| | | imageName | Calculated from filePath (!) |
| | | analyses | List of analyses (QCDAalysis) that have been performed on this image. |
| | | sample | The sample (QCDSample) that is captured by the image. |
| QCDMeasurement | A single “measurement” performed by a module. The measurement may encompass multiple input parameters, as well as multiple output results (e.g. mottle as standard deviation of mottle). A measurement is | completionMessage | |
| | | group | Text that defines a group of measurements to which this measurement belongs. This is optional and defined by the analysis script. |
| | | index | Integer which in combination with group is unique within a QCDAalysis. |

| Fundamental entities | | | |
|----------------------|---|-----------------|---|
| Entity name | Description | Key attributes | Description |
| | A measurement is usually performed on a single chart element, but in some cases can involve multiple chart elements (e.g. a permanence measurement that compares different elements). | tags | A string of text which can be composed of one or more actual tags to help categories the measurement. |
| | | analysis | The analysis (QCDAAnalysis) that produced the measurement. |
| | | elements | The chart element (QCDCartElement) (or multiple elements) that is the subject of the measurement. |
| | | module | The module (QCModule) used to perform the measurement. |
| | | parameterValues | |
| | | resultValues | |
| QCDAAnalysis | The total analysis of one digital image. This may encompass just one single measurement of one chart element on the image; or it may encompass diverse measurements with multiple modules on multiple chart elements. | dateEnded | Date and time analysis was ended. |
| | | dateStarted | Date and time analysis was started. |
| | | tags | |
| | | analysisScript | The analysis script (if any) used for the analysis. |
| | | chartScript | The chart script (if any) used for the analysis. |
| | | image | The image (QCImage) used for the analysis. |
| | | measurements | List of measurements (QCMeasurement) produced by the analysis. |
| | | oecf | The OECF (QCDOECF) used for the analysis. |
| | | | |
| | | session | The active session (QCSession). |
| QCD | | | |
| | | | |

Troubleshooting

Program crashes

In case the program crashes, please find two files which can help resolve the issue. Attach both of these files in an email to support@QiAnalytics.com.

- Relaunch QiQ, and select **File > Locations > Show Error Logs**. Find the file named **QiQ error log.txt**. This is a human readable file describing the conditions that caused the failure. (This file will not exist until at least one crash or other serious error has occurred). Log information is appended to this file, so information from all crashes will be retained.
- From the Mac's Finder, open the Library folder by holding down the option key while clicking the **Go** menu, then selecting **Library**. Inside the Library folder navigate to Logs > DiagnosticReports and here find the most recent file with a name that starts with QiQ. This file contains system generated crash information. You can also find this file by opening the **Console** app (located in **Applications > Utilities**), selecting **Crash Reports**, then looking for QiQ, right-click and select **Reveal in Finder**:

