



This document contains the confidential and proprietary information of Nanotronics Imaging, Inc. Neither this document nor any of the information contained is to be reproduced, distributed, used or disclosed, either in whole or in part, except as specifically authorized by Nanotronics Imaging, Inc.

nSpec Version 0.24.0.0

Release Date: 23 Feb 2024

Documentation Updated: 06 Mar 2024

Major Features: Device Yield Analysis, Device Yield Report, Multi-Sided Sample Scans

Overview

nSpec v0.24.0.0 contains many long-awaited updates including Device Yield Analysis and support for multi-sided sample scanning. This version also includes numerous quality of life updates: users can add a Lot ID to scans, use string templates when naming analysis export files, and customize image overlays in Cropped Images Analysis exports.

Additionally, this update includes changes to the database schema, and will require updates to any nJson files referencing certain parts of the old database schema. Nanotronics Technical Service will assist with this upgrade process.

Upgrading to v0.24.0.0

Upgrading Instructions

Library Update Not Required

If upgrading from a version more than 1 release prior, please reference all intermediate release notes for upgrade steps *for each version*.

Automatic Database Upgrade

WARNING: This version contains an automatic database upgrade process to update the database schema. **Old nJson files that reference the old database schemas must be updated with the support of Nanotronics Technical Support.**

This database upgrade process happens once a database is used in nSpec v0.24.0.0. The first time an old scan is used, the upgrade will occur and make the typical analysis process a little slower.

Once the upgrade is complete, the new upgraded scan databases cannot be used in previous nSpec versions. If needed, backups of the old scan databases should be made before performing any scans in nSpec v0.24.0.0.

Major Enhancements

Device Yield Analysis

Overview

Device Yield Analysis allows users to quickly calculate the device yield for a given sample, based on the output of a previous analysis. Each device is sorted into a user-defined bin, and each bin has its own unique, user-defined passing or failing criteria. Lastly, users set a threshold, which is the percentage of devices on the sample that must pass in order for the entire sample to pass.

Parameters

Device Yield Analysis contains four parameters: Base Analysis ID Mode, Base Analysis ID, Device Yield Threshold, and Device Yield Setting.

Analysis Parameters

Analysis: Device Yield

Group: Default

Description: Device Yield

P.△	Name	Type	Low	PV	High	Default	Picklist	Desc
01	Base Analysis ID Mode	PickList		Relative		Relative	Absolute, Re...	Specifies the base analysis ID selection mode. ...
02	Base Analysis ID	String						Specifies the target base analysis. When Base A.
05	Device Yield Threshold	Long	0	85	100	85		Percentage of devices that must be passed for .
06	Device Yield Setting	PickFile		\\WSA...		\\WS...	nJson	Absolute path to device yield setting file

Reload Default Delete

Save Save As Close

For Device Yield Analysis, the Base Analysis ID Mode and Base Analysis ID parameters should point to a previous analysis that outputs devices, for example Device Inspection Analysis or a Custom Reporter Analysis that was performed on a Device Inspection Analysis. These parameters point to previous analyses using standard nSpec Post-Analysis Referencing.

Device Yield Threshold is an integer from 0 to 100, which represents the percentage of devices on a sample that must meet passing criteria in order for the entire sample to pass.

Device Yield Setting is an nJson file that defines bins and bin-specific passing criteria for a given sample. This is the most critical part of the Device Yield Analysis.

Device Yield Setting nJson File

Below is an example Device Yield Setting nJson file.

```

{
  "Devices Report": {
    "inputTablesNames": "src",
    "BinFallThrough": 1,
    "PassColor": "64FF64",
    "FailColor": "FF6464",
    "Filtering"      :
      {
        "Inclusive" : true,
        "query"      : "SELECT DeviceID FROM vwDevices"
      },
    "Bins": [
      {
        "description": "Perfect",
        "value": 1,
        "pass": true,
        "color": "ADD8E6"
      },
      {
        "description": "Very Good",
        "value": 2,
        "pass": true,
        "color": "E0FFFF"
      },
      {
        "description": "Good",
        "value": 3,
        "pass": true,
        "color": "FFB6C1"
      },
      {
        "description": "Almost Good",
        "value": 4,
        "pass": false,
        "color": "FFA07A"
      },
      {
        "description": "Not Good",
        "value": 5,
        "pass": false,
        "color": "20B2AA"
      }
    ],
    "BinsQueries": [
      {
        "bin": 1,
        "priority": 5,

```

```

        "query": "SELECT DeviceID, Count(*) AS AddDesc FROM src
GROUP BY DeviceID HAVING Count(*) < 300"
    },
    {
        "bin": 2,
        "priority": 4,
        "query": "SELECT DeviceID, Count(*) AS AddDesc FROM src
GROUP BY DeviceID HAVING Count(*) >= 300 AND Count(*) < 400"
    },
    {
        "bin": 3,
        "priority": 3,
        "query": "SELECT DeviceID, Count(*) AS AddDesc FROM src
GROUP BY DeviceID HAVING Count(*) >= 400 AND Count(*) < 500"
    },
    {
        "bin": 4,
        "priority": 2,
        "query": "SELECT DeviceID, Count(*) AS AddDesc FROM src
GROUP BY DeviceID HAVING Count(*) >= 500 AND Count(*) < 1000"
    },
    {
        "bin": 5,
        "priority": 1,
        "query": "SELECT DeviceID, Count(*) AS AddDesc FROM src
GROUP BY DeviceID HAVING Count(*) >= 1000"
    }
]
}
}

```

In the above Device Yield Setting nJson file, the user defines five bins. Devices sorted into "Perfect", "Very Good", and "Good" bins as are assigned a passing value, and devices sorted into "Almost Good" and "Not Good" bins are assigned a failing value. The `Bins` definition includes a `value` field used to identify and reference the bin, and also a `color` field, which is the hex value of the color used to identify devices that fall under a particular bin in Device Yield nView reports.

The bins are further defined via `BinQueries`. In this example, the devices are sorted into bins based on the device's defect count. Each bin is defined using a SQL query contained in the `query` field. The "Not Good" bin contains devices with more than 1000 defects, and "Perfect" contains devices with less than 300 defects.

`BinQueries` are also given priorities -- if a device can be sorted into more than one bin, it will be assigned to a bin via the order defined by the `priority`. In this example, the condition for bin 5 or "Not Good" is checked first because it has a `priority` of 1. It is recommended that the conditions that reference the most critical defects are checked first.

If every bin is checked and the given wafer does not meet the criteria for any of the bins, the bin is sorted into the bin defined via the `BinFallThrough`. In this example, the fall through bin is 1, or "Perfect".

This example has very simple logic, but bins can be defined with much more complexity using SQL queries to the given input table defined in `inputTablesNames`. For example, defects can be sorted into bins by the type of defects found and/or the size of those defects.

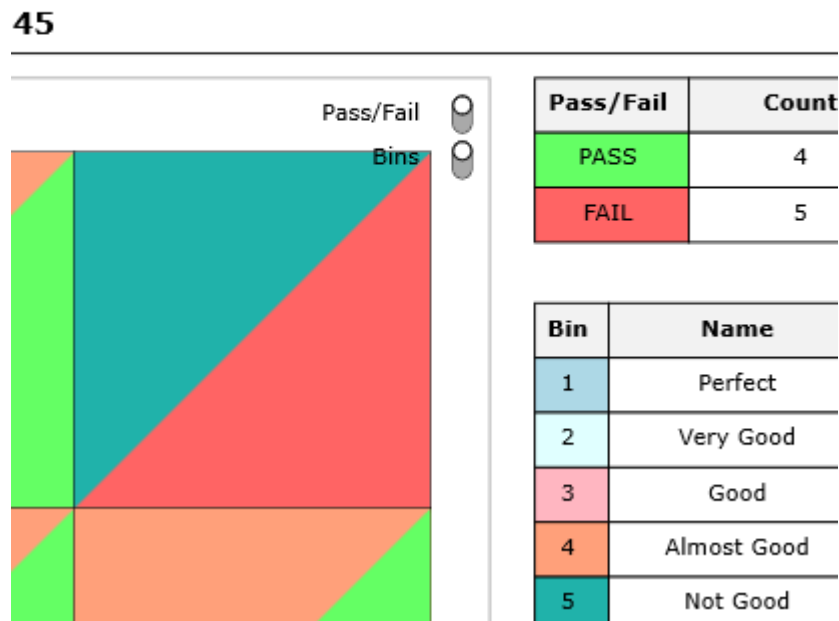
Below is an example of a bin query that checks for device type "EXTREMELY_BAD_DEFECT_TYPE" that is larger than 15 μm^2 .

```
"SELECT DeviceID, Area, ClassName FROM src WHERE Area >= 15 AND
ClassName = \"EXTREMELY_BAD_DEFECT_TYPE\" GROUP BY DeviceID"
```

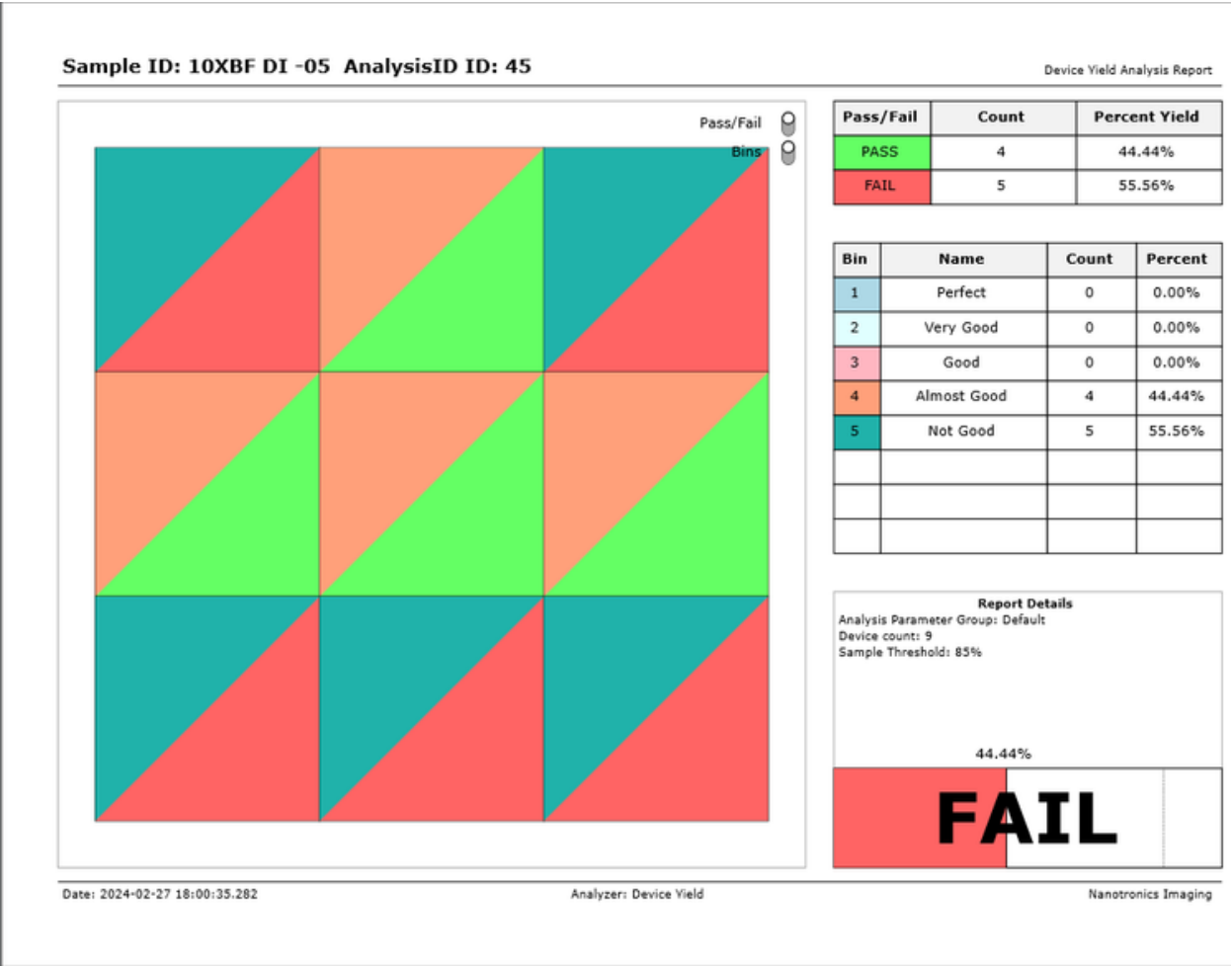
Reporting

After successfully running Device Yield, the analysis results can be viewed in nView. The Device Yield report can be viewed by accessing **nView > View > Device Yield Report** or **Ctrl+ 7**.

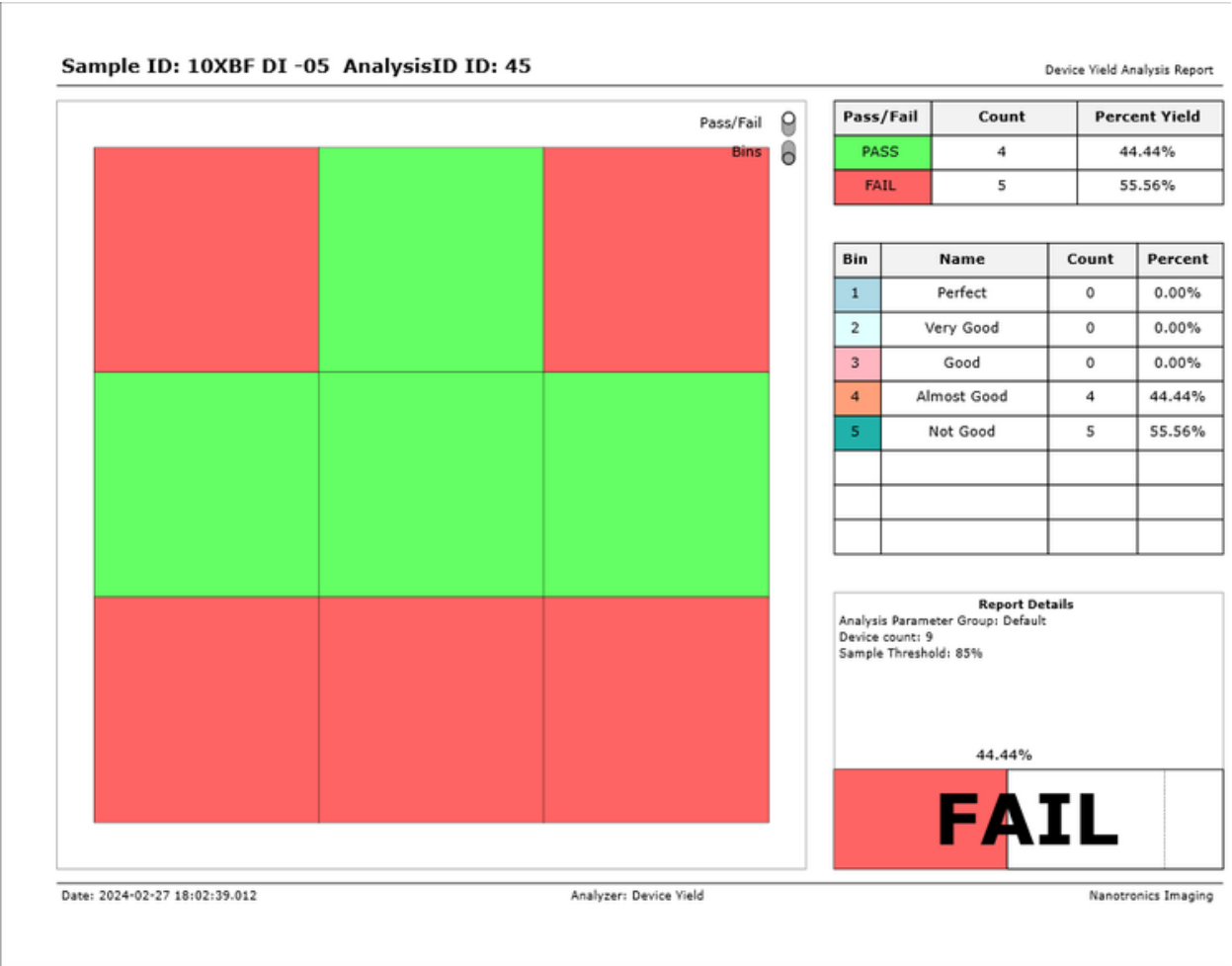
The Device Yield report has three different views that can be seen by toggling **Pass/Fail** and **Bins**, seen in the image below.



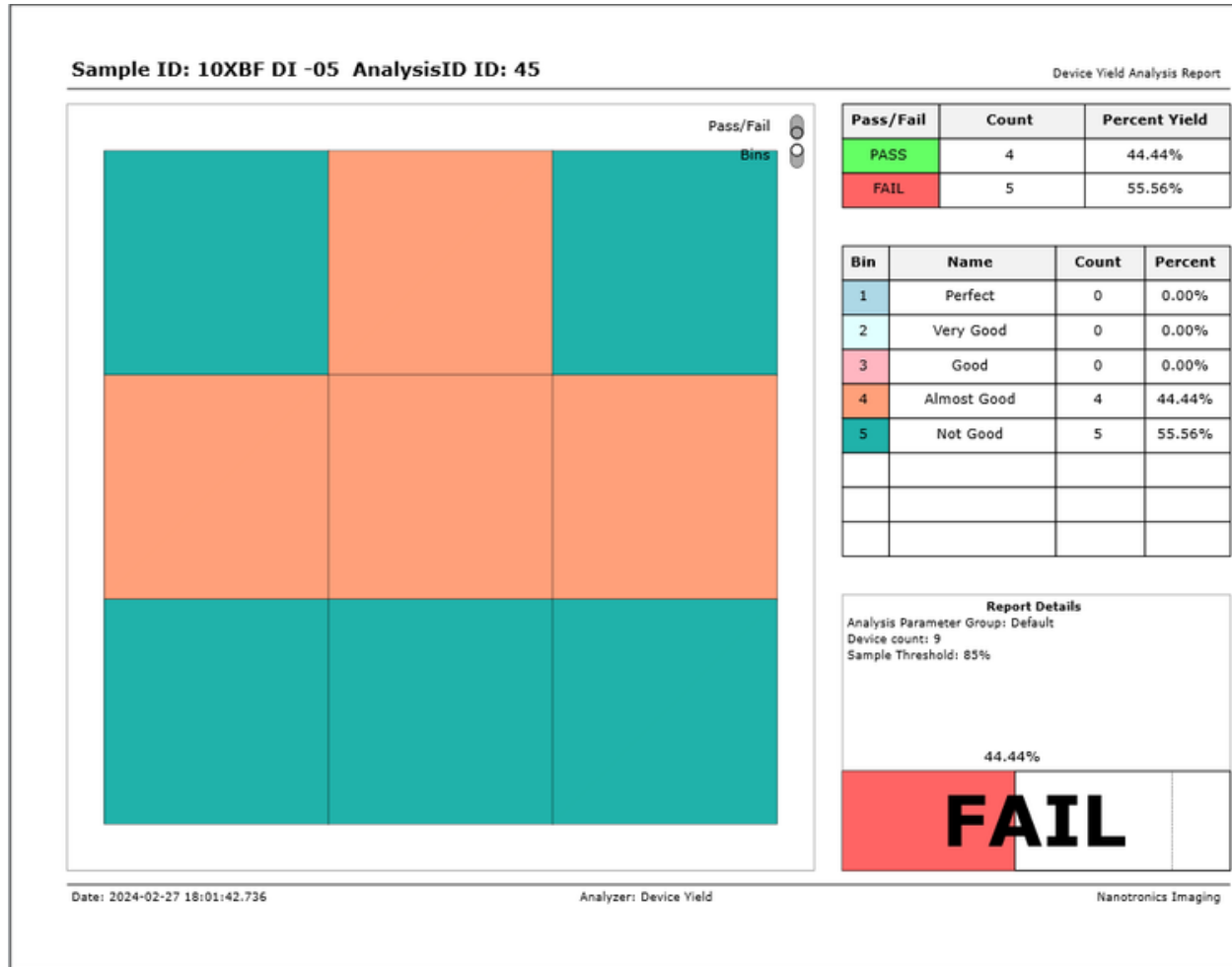
In the next image, both the **Pass/Fail** and **Bins** views are turned on. Each square represents a device, with the color in the top left corner of each device representing the bin associated with the device, and the color in the bottom right corner representing the pass/fail status of the device.



Bins can be toggled off to only show the pass/fail status of each device.



Similarly, **Pass/Fail** can be toggled off to show the device bins only.



Multi-Sided Sample Scanning

With a few hardware upgrades including the addition of a robot for handling samples, nSpec systems are capable of performing multi-sided sample scans.

In order to perform a multi-sided sample scan, nSpec needs to an autoloader to perform sample handling. During initial nSpec setup, a Nanotronics Technical Support Engineer will save autoloader calibration settings for handling and scanning each side of a multi-sided sample. For example, for a double-sided wafer, the engineer would save different autoloader settings for the top and bottom sides of the wafer.

These settings will populate in the **Autoloader** section of the **Scan Settings** dialog. Running a multi-sided sample scan necessitates a group job, and each side of the sample needs its own job or scan settings.

To continue using the previous example of a double-sided wafer, when setting up a job for the wafer's top side, the autoloader calibration setting for the top side of the wafer is set to **Side To Scan** value A.

Autoloader

Wafer Top Side Autoloader Settings

Description

Side To Scan Prealigner Angle (deg)

Job

... Props

Description

Similarly, the autoloader calibration setting for the wafer's bottom side is set to **Side To Scan** value B.

Autoloader

Wafer Bottom Side Autoloader Settings

Description

Side To Scan Prealigner Angle (deg)

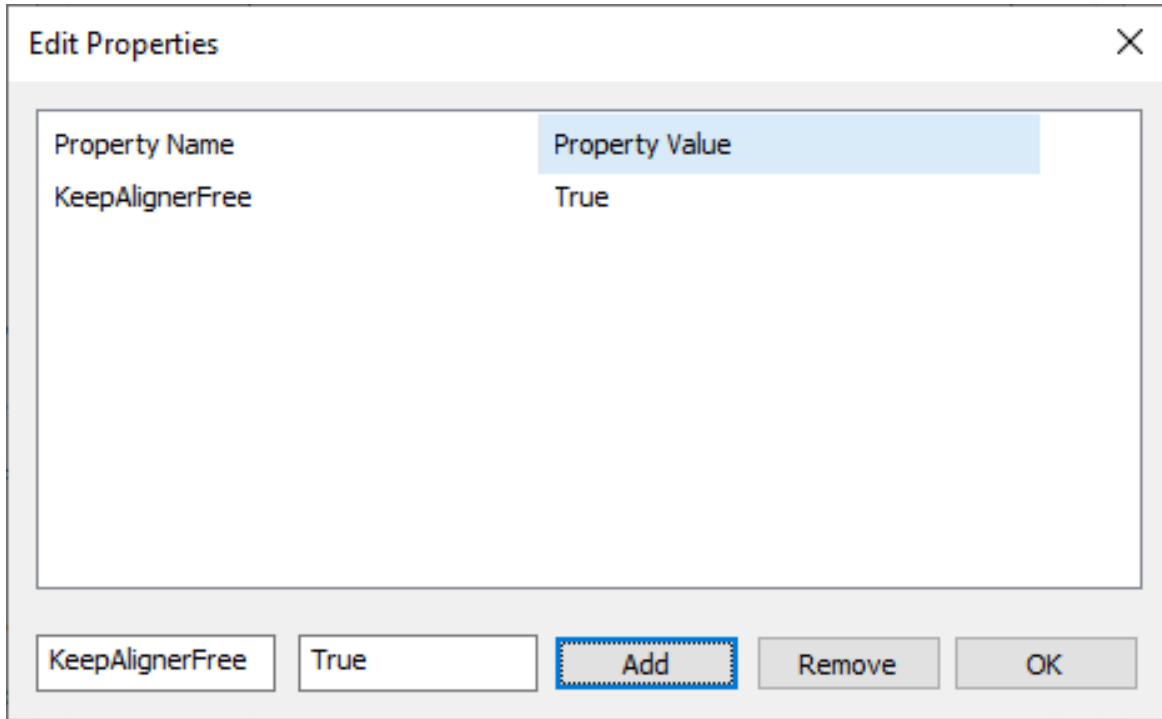
Job

... Props

Description

A and B are the default values for the **Side To Scan** setting, but are arbitrary and can be updated in `Calibration.db`.

Additionally, all jobs for multi-sided samples must be saved with Job Property `KeepAlignerFree` set to `true`.



The job properties dialog is access via the **Props** button in the **Job** section of **Scan Settings**.

After the job for each sample side is saved, all jobs should be added to a Job Group in the order that they should be performed.

New Features

Highlights

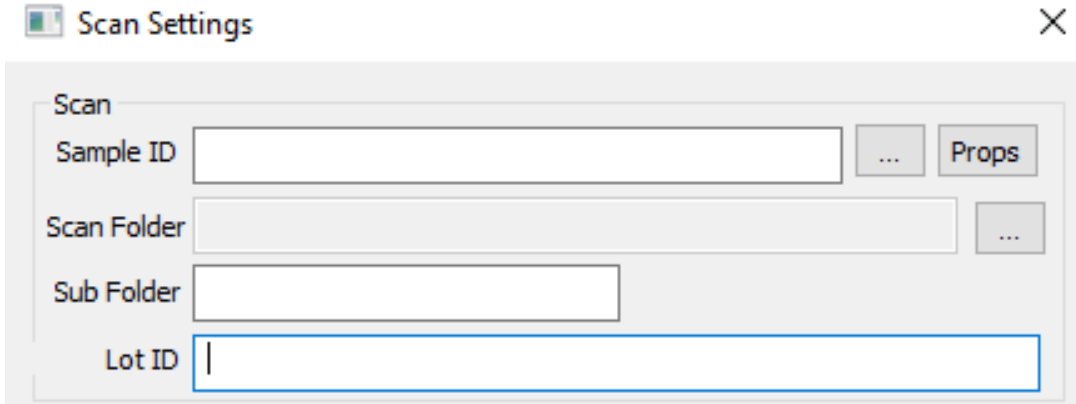
NSPEC-6905: Cropped Images Analysis migrated to use standard method of referencing previous analyses

When setting up a Cropped Images Analysis, previous analyses will be referenced with parameters Base Analysis ID Mode and Base Analysis ID. More about this reference system can be read in **Appendices: Post - Analysis Referencing**.

NSPEC-7957: Add Lot ID to Scan Dialog

Overview

Lot ID is an optional field that can be set in the **Scan Settings** dialog. It is saved to the database under Scan Properties.



The image shows a 'Scan Settings' dialog box with a title bar containing a file icon and a close button. The dialog has a 'Scan' section with four input fields: 'Sample ID', 'Scan Folder', 'Sub Folder', and 'Lot ID'. The 'Sample ID' field has a 'Props' button to its right. The 'Scan Folder' and 'Sub Folder' fields have ellipsis buttons to their right. The 'Lot ID' field is currently empty and has a blue border. The dialog is set against a light gray background.

Lot ID Input Requirements

- Must be 260 characters or less
- Cannot contain any of the following characters: / \ . : " | > < * ?
- Cannot contain tab characters
- Cannot end in a space
- Cannot contain any of the following reserved names: NULL, CON, PRN, AUX, NUL, COM0, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, COM¹, COM², COM³, LPT0, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, LPT9, LPT¹, LPT², and LPT³
- Empty strings are interpreted as "No Lot ID Specified"

Additionally, **Lot ID** is **not** case sensitive.

NSPEC-8003: Cropped Images Analysis includes Overlay File Configuration Parameter

Overlay Configuration File is an optional parameter for Cropped Images Analysis, which allows users to apply an customizable overlay to exported images. The parameter input type is a nJson file.

Analysis Parameters

Analysis: Cropped Images

Group: Default

Description: Cropped Image Generator Params

P.△	Name	Type	Low	PV	High	Default	Picklist	D
01	Base Analysis ID Mode	PickList		Relative ▾		Relative ▾	Absolute, Re...	S
02	Base Analysis ID	String						S
03	Image Output Limit Method	PickList		Absolu... ▾		Absol... ▾	Percentage, ...	T
04	Image Output Limit Percentage	Double	0	1	1	1		R
05	Image Output Limit Number	Long	1	1	2147483...	1		l
06	Image Size Limit Method	PickList		All ▾		All ▾	All, Larger E...	T
07	Image Size Limit	Double	0	10	200	10		S
08	Ranking Criterion	PickList		Rando... ▾		Rand... ▾	Random (32...	T
09	Ranking Criterion Ordering	PickList		Ascend... ▾		Ascen... ▾	Ascending, ...	U
10	Seed Random Number Generator	PickList		FALSE ▾		FALSE ▾	TRUE, FALSE	U
11	Random Number Generator Seed	Long	-214...	0	2147483...	0		U
12	Crop Padding (Pixels)	Long	0	20	100	20		P
13	Export File Full Path	StringBy...		\$(PATH... ▾		\$(PATH_T...		E
14	Export Many Images As Single File	PickList		FALSE ▾		FALSE ▾	TRUE, FALSE	If
15	Overlay Configuration File	PickFile		...		C:\Na... ...	nJson	If

< >

Reload Default Delete

Save Save As Close

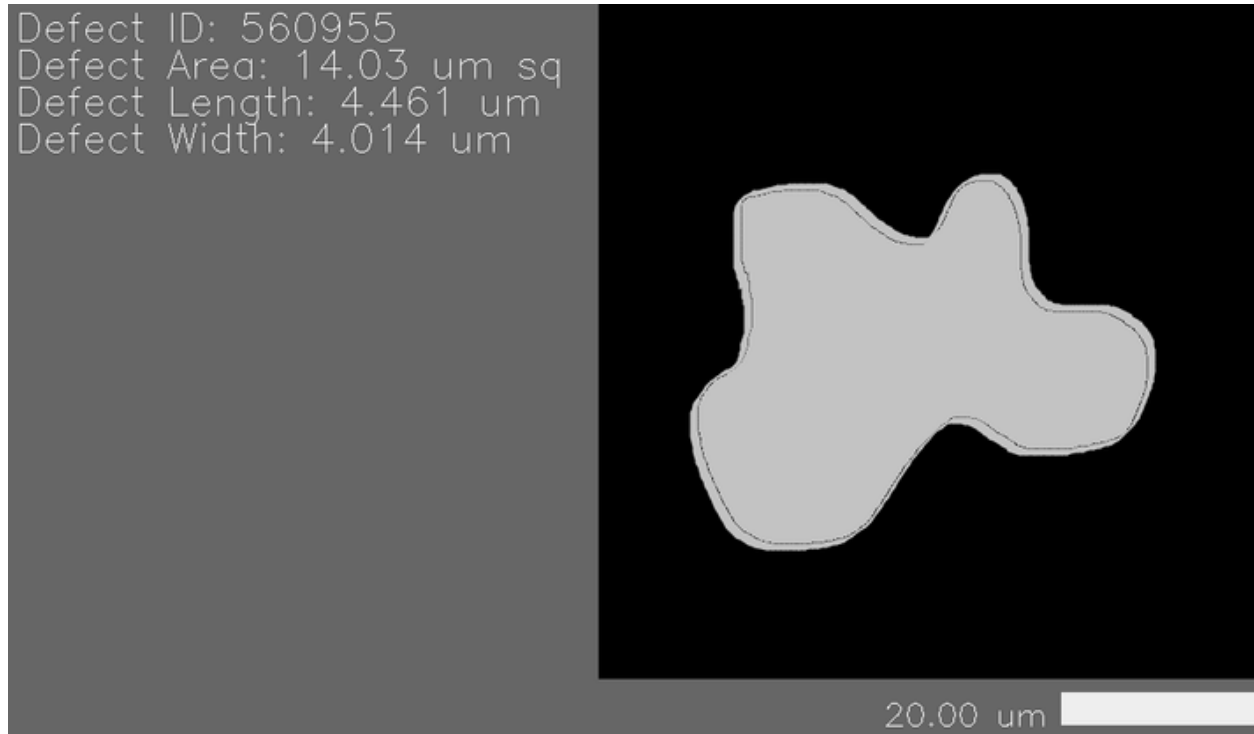
Below is an example nJson file for the Overlay Configuration File parameter.

```

{
  "Overlay Setting" :
  {
    "semver": "1.0.0",
    "viewport_dimensions": {
      "rows": 150,
      "output_width" : 512,
      "columns": 150
    },
    "background_color": "#666666",
    "sidebar_parameters": {
      "text_line_vertical_padding": 6,
      "text_line_horizontal_padding": 5,
      "sidebar_text_scale": 1,
      "sidebar_text_color": "#FFFFFF",
      "sidebar_width": 450,
      "sidebar_position": "left"
    },
    "scale_section_parameters": {
      "bar_top_padding": 10,
      "bar_height": 25,
      "bar_bottom_padding": 10,
      "horizontal_padding_between_label_and_bar": 10,
      "horizontal_padding_between_bar_and_edge_of_section": 10,
      "scale_bar_color": "#EEEEEE",
      "scale_section_justification": "right",
      "scale_bar_label_parameters": {
        "scale_label_text_color": "#FFFFFF",
        "scale_label_text_scale": 0.8,
        "vertical_padding_between_bottom_of_label_text_and_bottom_of_section": 10
      }
    }
  }
}

```

The above Overlay Configuration File produces the following cropped defect image with overlay. The cropped defect image dimensions, background color, sidebar defect information, and scale and their relative placements are all fully customizable.



`semver` is shorthand for semantic version, and refers to the version of the defect overlay schema used. Required.

`viewport_dimensions` sets the size of the viewport containing the cropped defect. Required.

`background_color` is the background color for the viewport, only used if the viewport size is larger than the cropped defect tile size. Required.

`sidebar_parameters` sets the size, font formatting, padding, and relative position of the image overlay sidebar, which contains defect information, including the ID, area, length, and width. Required.

`scale_section_parameters` sets the size and padding for the image overlay scale bar. Optional.

NSPEC-8098: Naming Export Files with String Templates

Export files from Custom Exporter, Cropped Images, and Report Summary Image Export now can be named with string template via the newly added Export File Full Path parameter.

- [Overview](#)
- [Example String Template Usage](#)
- [Table of Supported String Templates](#)

Overview

Users can dynamically name export files using string templates via the Export File Full Path parameter, available for Custom Exporter, Cropped Images, and Report Summary Image Export analyses.

Analysis Parameters

Analysis: Custom Exporter

Group: Default

Description: Custom Exporter

P.△	Name	Type	Low	PV	High	Default	Picklist
01	Base Analysis ID Mode	PickList		Relative		Relative	Absolute, Re.
02	Base Analysis ID	String					
03	Export Setting	PickFile		\\WSAMZ... ..		\\WSAM... ..	nJson
06	Export Type	PickList		CSV		CSV	CSV,KLARF,...
07	Export File Full Path	StringBy...		\$(PATH_TH...		\$(PATH_THI...	

Example String Template Usage

Here is an example of string template usage for the Export File Full Path parameter in Cropped Images Analysis.

The following input: `$(PATH_THIS_ANALYSIS)\$(SAMPLE_ID)_$(SCAN_ID)_$(ANALYSIS_ID)_$(ANALYSIS_SHORTNAME)\defect_$(DEFECT_ID)`

Results in the following export path:

C:
 \Scans\ExampleSampleID\Analysis\Analysis_001_002\ExampleSampleID_001_002_Crooped Image Generator\defect_12345



Table of Supported String Templates

Currently, under the Post-Analysis Support column, “All” refers to the three analyses that this feature has been implemented for: Custom Exporter, Cropped Images, and Report Summary Image Export.

Variable Name	Description	Post-Analysis Support
PATH_DB	Folder that Contains the image database ex: C:\Scans\	All
PATH_SCAN	Scan destination folder, the folder that contains "Scan_001" etc ex: C:\Scans\SampleID\	All
PATH_ANALYSIS	The incremented analysis destination folder within the scan folder. ex: C:\Scans\SampleID\Analysis\	All
PATH_THIS_ANALYSIS	The incremented analysis destination folder within the scan folder. ex: C:\Scans\SampleID\Analysis\Analysis_\${SCAN_ID}_\${ANALYSIS_ID}	All
SAMPLE_ID	Parent scan's sample ID	All
MACHINE_NAME	Name of machine (Device name in PC's settings System About)	All
JOB_NAME	Name of job that created parent scan	All
SCAN_ID	Parent scan's ID	All
ANALYSIS_GROUP_ID	Parent group analysis' ID (error if analysis is not part a group analysis)	All
ANALYSIS_GROUP_NAME	Parent group analysis' name (error if analysis is not part a group analysis)	All
ANALYSIS_ID	Analysis' ID	All
ANALYSIS_SHORTNAME	Analysis' IntName as shown in <i>Main View Utility</i> -> Edit Analyses...	All
ANALYSIS_DISPLAYNAME	Analysis' Name as shown in <i>Main View Utility</i> -> Edit Analyses...	All
DATETIME	Analysis' completion time with format: %Y%M%DT%h%m%s	All
TARGET_ANALYZER_DISPLAYNAME	Target analysis' Name as shown in <i>Main View Utility</i> -> Edit Analyses...	All
TARGET_ANALYZER_SHORTNAME	Target analysis' IntName as shown in <i>Main View Utility</i> -> Edit Analyses...	All

TARGET_ANALYSIS_DATETIME	Target analysis' completion time with format: %Y%M%DT%h%m%s	All
TARGET_ANALYSIS_ID	Target analysis' ID	All
DEFECT_IMAGE_ID	Cropped defect's parent image ID	Cropped Images
DEFECT_DEVICE_ID	Cropped defect's parent device ID	Cropped Images
DEFECT_ID	Cropped defect's ID	Cropped Images
DEFECT_CLASSIFICATION	Cropped defect's classification name (N/A if defect is not classified)	Cropped Images
TARGET_ANALYSIS_DESCRIPTION	Reporting image export's Report Type	Reporting Summary Image Export






New Features Changelog

T	Key	Release Notes Summary
	NSPEC-6453	Scan Database now includes SampleRegions data
	NSPEC-6455	Implement Device Yield Analysis
	NSPEC-6634	Automatic upgrade of scan databases to new database schema
	NSPEC-6905	Cropped Images Analysis migrated to use standard method of referencing previous analyses
	NSPEC-7124	Device Yield Report can be selected in nView
	NSPEC-7909	Overlay in Cropped Images Analysis output now contains Device ID
	NSPEC-7957	Add Lot ID to Scan Dialog
	NSPEC-7979	Update Sample ID, Scan folder, Subfolder, and Lot ID to be Editable at the Operator Level
	NSPEC-8003	Cropped Images Analysis includes Overlay File Configuration Parameter
	NSPEC-8057	Ability to add Custom Filename Suffix to Report Summary Image Export
	NSPEC-8098	Naming Export Files with String Templates

11 issues

Bug Fixes

Changelog

T	Key	Release Notes Summary	Affected Releases
	 NSPEC-7976	When running any analyzer that has an nJson setting file as a parameter, the setting file contents are not saved to parameter groups	0.23.0.0, 0.23.1.0
	 NSPEC-8049	Cassette OCR group jobs with empty sample IDs try to pick up every wafer	0.23.1.1
	 NSPEC-8200	Unable to operate RGBW Transmitted Light Source	0.23.1.0, 0.23.0.2, 0.23.1.1
	 NSPEC-8220	ProgramOptions Compare utility not working	0.23.1.0, 0.23.1.1
	 NSPEC-8223	Error running GEM cassette jobs with skipped slots	0.23.1.0

5 issues

Appendices

Post-Analysis Referencing

There are a few nSpec Analyses that reference other, previously run analyses. As of version `v0.23.0.0` this includes **Custom Reporter**, **Custom Exporter**, and **Defect Classifier** post-analyses.

The fields pertaining to prior referencing include:

Base Analysis IDs Mode	Absolute, Relative	The user selects the nature of analysis ID(s) to be specified in the following field. Relative = most recently run, while absolute = the numeric integer reference of a prior run analysis
Base Analysis IDs		<p>Comma separated set of IDs. If <i>Mode</i> is Absolute, it is a set of integers (example: "1,2").</p> <p>If <i>mode</i> is Relative, it is a set of source analysis names. For example, Device Inspection, Gen IV Analysis means that first view (DI from the nJson) will be the Defects of the last analysis of type Device Inspection, while the second source analysis (GenIV from the nJson) will be populated with the Defects from the most recently run Gen IV Analysis.</p>

Relative Reference Notes

When using Relative mode, you also have the option to reference any previously run analysis, by adding a numeric indicator at the end of the analysis name. Let's say an image database ran 3 analyses:

1. Gen IV AI Analysis (parameter set 1)
2. Gen IV AI Analysis (parameter set 2)
3. Device Inspection

To point at #2 and #3, you would input the following for the **Base Analysis IDs** field:

```
"DeviceInspection, Gen IV Analysis"
```

To point at #1 and #3, you would input the following for the **Base Analysis IDs** field:

```
"DeviceInspection, Gen IV Analysis.2"
```

The .2 at the end of Gen IV Analysis informs nSpec to reference the second most recently run Gen VI Analysis. In this way, you have complete flexibility to point at any previously run analysis using both absolute or relative modes.

If you wish to reference the most recent analysis, there is no need to add the “.1” after the analysis internal name reference. This means that “DeviceInspection” is the same as “DeviceInspection.1”

Analysis Reference List

The user-facing name of an analysis is not necessarily identical to the ID that should be used to reference that analysis.

Analysis Name (user facing)	Analysis Reference ID (IntName)
AI Analysis	AIAnalysis
Basic Selection (Contrast, Morphological Range)	nrad_rangeselect
Basic Selection (Intensity, Exclusive)	nrad_intensityselectexclusive
Basic Selection (Intensity, Inclusive)	nrad_intensityselectinclusive
Crack Detection V2	CrackDetectionV2
Custom Exporter	Custom Exporter
Custom Reporter	Custom Reporter
Device Inspection	DeviceInspection
Die Yield	nrad_dieyield
Dislocation Defect Analyzer	Dislocation
Gen IV AI Analysis	Gen IV Analysis
High Resolution Mosaic Generation	High Resolution Mosaic Generation Analysis

Surface Scattering Analysis	Surface Scattering Analysis
Utility: Launch Executable	LaunchScript

When to use Absolute, and when to use Relative

When testing a new script, it might be useful to use absolute mode. However, in the majority of cases, we believe that relative mode is most likely to be the optimal mode, especially when running in production. Consider a case where you scan a wafer and perform a Group Analysis on this scan. The first time, using absolute references to point at the 1st and 2nd analyses may work fine. But what happens if you run that group analysis a second time? Now, the post-analysis is still performed on the first and second analyses ever performed, rather than the ones most recently performed as part of that new analysis group.

When running in production, we recommend always using **relative mode**

nSpec v0.24.0.0 Agent Release Notes

nSpec Version 0.24.0.0

Release Date: 23 Feb 2024

Documentation Updated: 06 Mar 2024

Major Features: Device Yield Analysis, Device Yield Report, Multi-Sided Sample Scans

Overview

nSpec v0.24.0.0 contains many long-awaited updates including Device Yield Analysis and support for multi-sided sample scanning. This version also includes numerous quality of life updates: users can add a Lot ID to scans, use string templates when naming analysis export files, and customize image overlays in Cropped Images Analysis exports.

Additionally, this update includes changes to the database schema, and will require updates to any nJson files referencing certain parts of the old database schema. Nanotronics Technical Service will assist with this upgrade process.

Upgrading to v0.24.0.0

Upgrading Instructions

Library Update Not Required

If upgrading from a version more than 1 release prior, please reference all intermediate release notes for upgrade steps *for each version*.

Automatic Database Upgrade

WARNING: This version contains an automatic database upgrade process to update the database schema. **Old nJson files that reference the old database schemas must be updated with the support of Nanotronics Technical Support.**

This database upgrade process happens once a database is used in nSpec v0.24.0.0. The first time an old scan is used, the upgrade will occur and make the typical analysis process a little slower.

Once the upgrade is complete, the new upgraded scan databases cannot be used in previous nSpec versions. If needed, backups of the old scan databases should be made before performing any scans in nSpec v0.24.0.0.

Notes on Automatic Database Upgrade

[Automatic Database Upgrade](#)

[Overview](#)

[Prerequisites for Upgrading to nSpec v0.24.0.0](#)

Changes in v0.24.0.0 Database Schema

[Tables](#)

[Views](#)

[Indices](#)

[Triggers](#)

Example Custom Exporter Script Change

[nSpec v0.23.1.1 Custom Exporter Script](#)

[nSpec v0.24.0.0 Custom Exporter Script](#)

Automatic Database Upgrade

Overview

Databases in nSpec v0.24.0.0 have significant schema changes, and this documentation details how to safely upgrade a customer to nSpec v0.24.0.0.

Prerequisites for Upgrading to nSpec v0.24.0.0

The automatic database upgrade process in v0.24.0.0 happens once a database is used in nSpec. Before upgrading to nSpec v0.24.0.0, there are a number of checks that should be done to ensure a successful software update.

1. Review all customer scripts for references to the database . Any references to the old database schema must be replaced with appropriate references to the new database schema. An example script change can be found further below.
 - a. All Custom Exporter scripts must be reviewed and test run to ensure they do not break in v0.24.0.0.
2. [Recommended] Make a backup of the customer's databases. **Once their databases are upgraded, the new upgraded databases cannot be used in previous nSpec versions.**
3. Upgrade customer to nSpec v0.24.0.0, launch nSpec, and perform a scan. During this first scan, the scan database upgrade will automatically start and thus will make the typical analysis process a little slower.

Changes in v0.24.0.0 Database Schema

Tables

Changelist between Tables from v0.23.1.1 to v0.24.0.0



Defects



Devices



Images



DefectBase
































































DefectDevice











- + DefectImage
- + DefectRegion
- + DeviceYieldData
- + DeviceYieldFilteredDevices
- + DevicesBins
- + DevicesBinsDef
- + RegionDevices
- + RegionImages
- + RegionROI
- + SampleRegion

v0.23.1.1 Tables	v0.24.0.0 Tables

Name
▼  Tables (29)
>  Analysis
>  AnalysisDefinitions
>  AnalysisGroups
>  AnalysisProperties
>  Analyzers
>  CustomPatterns
>  Defects
>  DetectionClasses
>  Devices
>  DevicesInImages
>  GoldenTiles
>  Images
>  LineDefects
>  ManualExclusions
>  Measurements
>  ParameterGroups
>  ParameterTypes
>  Parameters
>  PostAnalysisDefectFilter
>  PostAnalysisDefectFilterAreaRange
>  PreFocusPoints
>  RegionsOfInterest
>  ScanProperties
>  Scans
>  TileValues
>  Transformations
>  XRef_ScanId_PatternId
>  _dbProperties
>  sqlite_sequence








Name
▼  Tables (38)
>  Analysis
>  AnalysisDefinitions
>  AnalysisGroups
>  AnalysisProperties
>  Analyzers
>  CustomPatterns
>  DefectBase
>  DefectDevice
>  DefectImage
>  DefectRegion
>  DetectionClasses
>  DeviceYieldData
>  DeviceYieldFilteredDevi
>  DevicesBins
>  DevicesBinsDef
>  DevicesInImages
>  GoldenTiles
>  LineDefects
>  ManualExclusions
>  Measurements
>  ParameterGroups
>  ParameterTypes
>  Parameters
>  PostAnalysisDefectFilter
>  PostAnalysisDefectFilter
>  PreFocusPoints
>  RegionDevices
>  RegionImages
>  RegionROI
>  RegionsOfInterest

- >  **SampleRegion**
- >  ScanProperties
- >  Scans
- >  TileValues
- >  Transformations
- >  XRef_ScanId_PatternId
- >  _dbProperties
- >  sqlite_sequence

Views

- vwDefectsLegacy is a **temporary** view only version of the Defects table from v0.23.1.1. This view will be deleted sometime in the future.
 - vwDefects is the same between v0.23.1.1 and 0.24.0.0, which is a join of Defects and Classifications, adding ClassName.
- vwDevices and vwImages are view only versions of the Devices and Images tables from v0.23.1.1.

Changelist between Views from v0.23.1.1 to v0.24.0.0

-  vwDefectsLegacy
-  vwDefectsRegionIDs
-  vwDevices
-  vwDevicesBinsPassFail
-  vwDevicesDefects
-  vwImageDefects
-  vwImages

v0.23.1.1 Views

v0.24.0.0 Views

Indices

✓








- Defects_GetDefectList_Index
- + DefectBase_GetDefectList_Index
- + DefectImage_GetDefectList_Index
- + DefectRegion_DefectID_Index
- + RegionImages_RegionID_Index

v0.23.1.1 Indices

Indices (17)	Indices (20)
> AnalysisDefinitions_GroupGUID_Index	> AnalysisDefinitions_Group
> AnalysisDefinitions_Index	> AnalysisDefinitions_Index
> AnalysisGroups_GroupGUID_Index	> AnalysisGroups_GroupGU
> AnalysisProperties_AnalysisImages_In...	> AnalysisProperties_Analy
> Analysis_vwAnalysis_Index	> Analysis_vwAnalysis_Indi
> Analyzers_vwAnalysis_Index	> Analyzers_vwAnalysis_In
> Defects_GetDefectList_Index	> DefectBase_GetDefectLis
> DevicesInImagesDevices_Index	> DefectImage_GetDefectL
> DevicesInImagesImages_Index	> DefectRegion_DefectID_I
> Images_GetImageFilename_Index	> DevicesInImagesDevices
> Images_GetImageInfo_Index	> DevicesInImagesImages
> Images_vwScan_Index	> Images_GetImageFilenar
> LineDefects_GetDefectList_Index	> Images_GetImageInfo_In
> Measurements_GetMeasurements_Index	> Images_vwScan_Index
> ScanProperties_ScanImages_Index	> LineDefects_GetDefectLis
> Scans_Timestamp_Index	> Measurements_GetMeas
> Scans_vwScan_Index	> RegionImages_RegionID
	> ScanProperties_ScanIma
	> Scans_Timestamp_Index
	> Scans_vwScan_Index

Triggers

Changelist between Triggers from v0.23.1.1 to v0.24.0.0

-  OnDelete_DefectBase
-  OnDelete_DefectDevice
-  OnDelete_DefectImage
-  OnDelete_DefectRegion
-  OnDelete_RegionDevices
-  OnDelete_RegionImages
-  OnDelete_RegionROI



OnDelete_SampleRegion

v0.23.1.1 Triggers	v0.24.0.0 Triggers
<div><div>Triggers (3)</div><div><div>delete_scan_manual_exclusions</div><div>delete_scan_pattern</div><div>delete_scan_pattern_xref</div></div></div>	<div><div>Triggers (11)</div><div><div>OnDelete_Defect</div><div>OnDelete_Defect</div><div>OnDelete_Defect</div><div>OnDelete_Defect</div><div>OnDelete_Region</div><div>OnDelete_Region</div><div>OnDelete_Region</div><div>OnDelete_Sampl</div><div>delete_scan_mar</div><div>delete_scan_patt</div><div>delete_scan_patt</div></div></div>

Example Custom Exporter Script Change

In this example, a script used in v0.23.1.1 referencing tables Defects, Devices, and Images are replaced with references to views vwDefects, vwDevices, vwImages.

nSpec v0.23.1.1 Custom Exporter Script

```

{
  "Custom Exporter": {
    "inputTablesNames": "source",
    "Queries": [
      {
        "Title": "DefectList",
        "Query": "WITH cte_coordinates AS ( SELECT D.*, I.X AS IX,
I.Y AS IY FROM Defects AS D JOIN Images AS I ON D.ImageID = I.
ImageID WHERE AnalysisID = (SELECT AnalysisID FROM source) AND
ScanProperties.ScanID = (SELECT MAX(ScanID) FROM ScanProperties)),
cte_centerX AS ( SELECT ScanProperties.PropertyData AS CX FROM
ScanProperties WHERE ScanProperties.PropertyName = 'SampleCenterX'
AND ScanProperties.ScanID = (SELECT ScanID FROM cte_coordinates) ),
cte_centerY AS ( SELECT ScanProperties.PropertyData AS CY FROM
ScanProperties WHERE ScanProperties.PropertyName = 'SampleCenterY'
AND ScanProperties.ScanID = (SELECT ScanID FROM cte_coordinates) ),
Stage_Limit AS ( SELECT ScanProperties.PropertyData AS YLim FROM
ScanProperties WHERE ScanProperties.PropertyName = 'Stage Height
Microns' AND ScanProperties.ScanID = (SELECT MAX(ScanID) FROM
ScanProperties) ) SELECT defect.DefectID AS DEFECTID, (image.X +
defect.X) AS X, ( (SELECT Stage_Limit.YLim FROM Stage_Limit) -
image.Y - defect.Y) AS Y, (device.X + defect.X) AS XREL , (0 -
device.Y - defect.Y) AS YREL , device.XIndex AS XINDEX, device.
YIndex AS YINDEX ,defect.W AS XSIZE, defect.H AS YSIZE, defect.Area
AS DEFECTAREA , (SELECT (-190 /((defect.W * defect.W) + (defect.H *
defect.H) + 20))+ 10) AS DSIZE, (SELECT ID FROM DetectionClasses
WHERE ID = defect.ClassID) AS CLASSNUMBER, 0 AS TEST, 0 AS
CLUSTERNUMBER, 0 AS ROUGHBINNUMBER, 0 as FINEBINNUMBER, 0 AS
REVIEWSAMPLE, 1 AS IMAGECOUNT, (SELECT SUBSTR(Filename,1,9) FROM
Images WHERE ImageID = defect.ImageID) AS IMAGELIST FROM Defects AS
defect LEFT OUTER JOIN Devices AS device ON defect.DeviceID =
device.DeviceID LEFT OUTER JOIN Images AS image ON defect.ImageID =
image.ImageID WHERE AnalysisID = (SELECT AnalysisID FROM source);"
      }
    ]
  }
}

```

nSpec v0.24.0.0 Custom Exporter Script

```
{
  "Custom Exporter": {
    "inputTablesNames": "source",
    "Queries": [
      {
        "Title": "DefectList",
        "Query": "WITH cte_coordinates AS ( SELECT D.*, I.X AS IX,
I.Y AS IY FROM vwDefects AS D JOIN vwImages AS I ON D.ImageID = I.
ImageID WHERE AnalysisID = (SELECT AnalysisID FROM source) AND
ScanProperties.ScanID = (SELECT MAX(ScanID) FROM ScanProperties)),
cte_centerX AS ( SELECT ScanProperties.PropertyData AS CX FROM
ScanProperties WHERE ScanProperties.PropertyName = 'SampleCenterX'
AND ScanProperties.ScanID = (SELECT ScanID FROM cte_coordinates) ),
cte_centerY AS ( SELECT ScanProperties.PropertyData AS CY FROM
ScanProperties WHERE ScanProperties.PropertyName = 'SampleCenterY'
AND ScanProperties.ScanID = (SELECT ScanID FROM cte_coordinates) ),
Stage_Limit AS ( SELECT ScanProperties.PropertyData AS YLim FROM
ScanProperties WHERE ScanProperties.PropertyName = 'Stage Height
Microns' AND ScanProperties.ScanID = (SELECT MAX(ScanID) FROM
ScanProperties) ) SELECT defect.DefectID AS DEFECTID, (image.X +
defect.X) AS X, ( (SELECT Stage_Limit.YLim FROM Stage_Limit) -
image.Y - defect.Y) AS Y, (device.X + defect.X) AS XREL , (0 -
device.Y - defect.Y) AS YREL , device.XIndex AS XINDEX, device.
YIndex AS YINDEX ,defect.W AS XSIZE, defect.H AS YSIZE, defect.Area
AS DEFECTAREA , (SELECT (-190 /((defect.W * defect.W) + (defect.H *
defect.H) + 20))+ 10) AS DSIZE, (SELECT ID FROM DetectionClasses
WHERE ID = defect.ClassID) AS CLASSNUMBER, 0 AS TEST, 0 AS
CLUSTERNUMBER, 0 AS ROUGHBINNUMBER, 0 as FINEBINNUMBER, 0 AS
REVIEWSAMPLE, 1 AS IMAGECOUNT, (SELECT SUBSTR(Filename,1,9) FROM
vwImages WHERE ImageID = defect.ImageID) AS IMAGELIST FROM
vwDefects AS defect LEFT OUTER JOIN vwDevices AS device ON defect.
DeviceID = device.DeviceID LEFT OUTER JOIN vwImages AS image ON
defect.ImageID = image.ImageID WHERE AnalysisID = (SELECT
AnalysisID FROM source);"
      }
    ]
  }
}
```

Major Enhancements

Device Yield Analysis

Overview

Device Yield Analysis allows users to quickly calculate the device yield for a given sample, based on the output of a previous analysis. Each device is sorted into a user-defined bin, and each bin has its own unique, user-defined passing or failing criteria. Lastly, users set a threshold, which is the percentage of devices on the sample that must pass in order for the entire sample to pass.

Parameters

Device Yield Analysis contains four parameters: Base Analysis ID Mode, Base Analysis ID, Device Yield Threshold, and Device Yield Setting.

Analysis Parameters

Analysis: Device Yield ▾

Group: Default ▾ ...

Description: Device Yield

P.△	Name	Type	Low	PV	High	Default	Picklist	Desc
01	Base Analysis ID Mode	PickList		Relative ▾		Relative ▾	Absolute, Re...	Specifies the base analysis ID selection mode. ...
02	Base Analysis ID	String						Specifies the target base analysis. When Base A.
05	Device Yield Threshold	Long	0	85	100	85		Percentage of devices that must be passed for .
06	Device Yield Setting	PickFile		\\WSA... ...		\\WS... ...	n/json	Absolute path to device yield setting file

Reload Default Delete Save Save As Close

For Device Yield Analysis, the Base Analysis ID Mode and Base Analysis ID parameters should point to a previous analysis that outputs devices, for example Device Inspection Analysis or a Custom Reporter Analysis that was performed on a Device Inspection Analysis. These parameters point to previous analyses using standard nSpec Post-Analysis Referencing.

Device Yield Threshold is an integer from 0 to 100, which represents the percentage of devices on a sample that must meet passing criteria in order for the entire sample to pass.

Device Yield Setting is an nJson file that defines bins and bin-specific passing criteria for a given sample. This is the most critical part of the Device Yield Analysis.

Device Yield Setting nJson File

Below is an example Device Yield Setting nJson file.

```
{
  "Devices Report": {
    "inputTablesNames": "src",
    "BinFallThrough": 1,
  }
}
```



```

"PassColor": "64FF64",
"FailColor": "FF6464",
"Filtering"      :
  {
    "Inclusive" : true,
    "query"      : "SELECT DeviceID FROM vwDevices"
  },
"Bins": [
  {
    "description": "Perfect",
    "value": 1,
    "pass": true,
    "color": "ADD8E6"
  },
  {
    "description": "Very Good",
    "value": 2,
    "pass": true,
    "color": "E0FFFF"
  },
  {
    "description": "Good",
    "value": 3,
    "pass": true,
    "color": "FFB6C1"
  },
  {
    "description": "Almost Good",
    "value": 4,
    "pass": false,
    "color": "FFA07A"
  },
  {
    "description": "Not Good",
    "value": 5,
    "pass": false,
    "color": "20B2AA"
  }
],
"BinsQueries": [
  {
    "bin": 1,
    "priority": 5,
    "query": "SELECT DeviceID, Count(*) AS AddDesc FROM src
GROUP BY DeviceID HAVING Count(*) < 300"
  },
  {

```

```

        "bin": 2,
        "priority": 4,
        "query": "SELECT DeviceID, Count(*) AS AddDesc FROM src
GROUP BY DeviceID HAVING Count(*) >= 300 AND Count(*) < 400"
    },
    {
        "bin": 3,
        "priority": 3,
        "query": "SELECT DeviceID, Count(*) AS AddDesc FROM src
GROUP BY DeviceID HAVING Count(*) >= 400 AND Count(*) < 500"
    },
    {
        "bin": 4,
        "priority": 2,
        "query": "SELECT DeviceID, Count(*) AS AddDesc FROM src
GROUP BY DeviceID HAVING Count(*) >= 500 AND Count(*) < 1000"
    },
    {
        "bin": 5,
        "priority": 1,
        "query": "SELECT DeviceID, Count(*) AS AddDesc FROM src
GROUP BY DeviceID HAVING Count(*) >= 1000"
    }
]
}
}

```

In the above Device Yield Setting nJson file, the user defines five bins. Devices sorted into "Perfect", "Very Good", and "Good" bins as are assigned a passing value, and devices sorted into "Almost Good" and "Not Good" bins are assigned a failing value. The `Bins` definition includes a `value` field used to identify and reference the bin, and also a `color` field, which is the hex value of the color used to identify devices that fall under a particular bin in Device Yield nView reports.

The bins are further defined via `BinQueries`. In this example, the devices are sorted into bins based on the device's defect count. Each bin is defined using a SQL query contained in the `query` field. The "Not Good" bin contains devices with more than 1000 defects, and "Perfect" contains devices with less than 300 defects.

`BinQueries` are also given priorities -- if a device can be sorted into more than one bin, it will be assigned to a bin via the order defined by the `priority`. In this example, the condition for bin 5 or "Not Good" is checked first because it has a `priority` of 1. It is recommended that the conditions that reference the most critical defects are checked first.

If every bin is checked and the given wafer does not meet the criteria for any of the bins, the bin is sorted into the bin defined via the `BinFallThrough`. In this example, the fall through bin is 1, or "Perfect".

This example has very simple logic, but bins can be defined with much more complexity using SQL queries to the given input table defined in `inputTablesNames`. For example, defects can be sorted into bins by the type of defects found and/or the size of those defects.

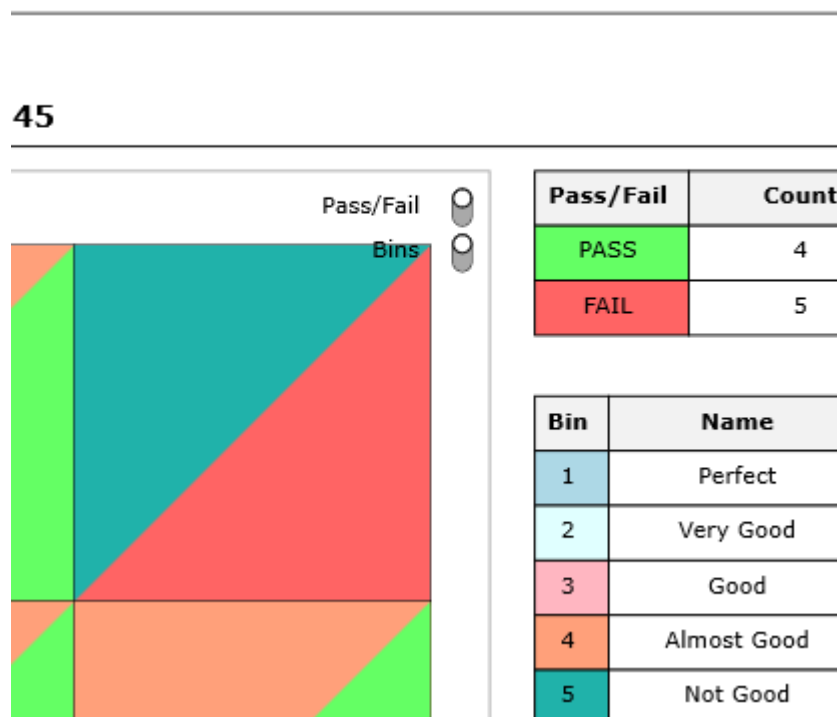
Below is an example of a bin query that checks for device type "EXTREMELY_BAD_DEFECT_TYPE" that is larger than 15 μm^2 .

```
"SELECT DeviceID, Area, ClassName FROM src WHERE Area >= 15 AND
ClassName = \"EXTREMELY_BAD_DEFECT_TYPE\" GROUP BY DeviceID"
```

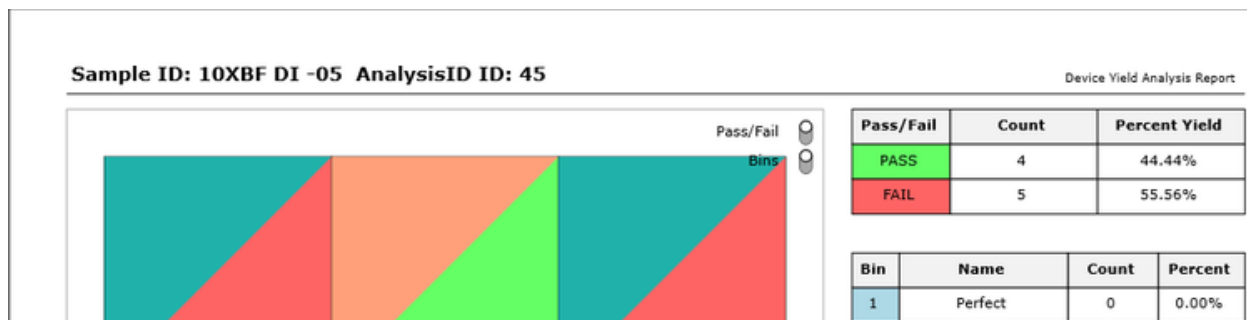
Reporting

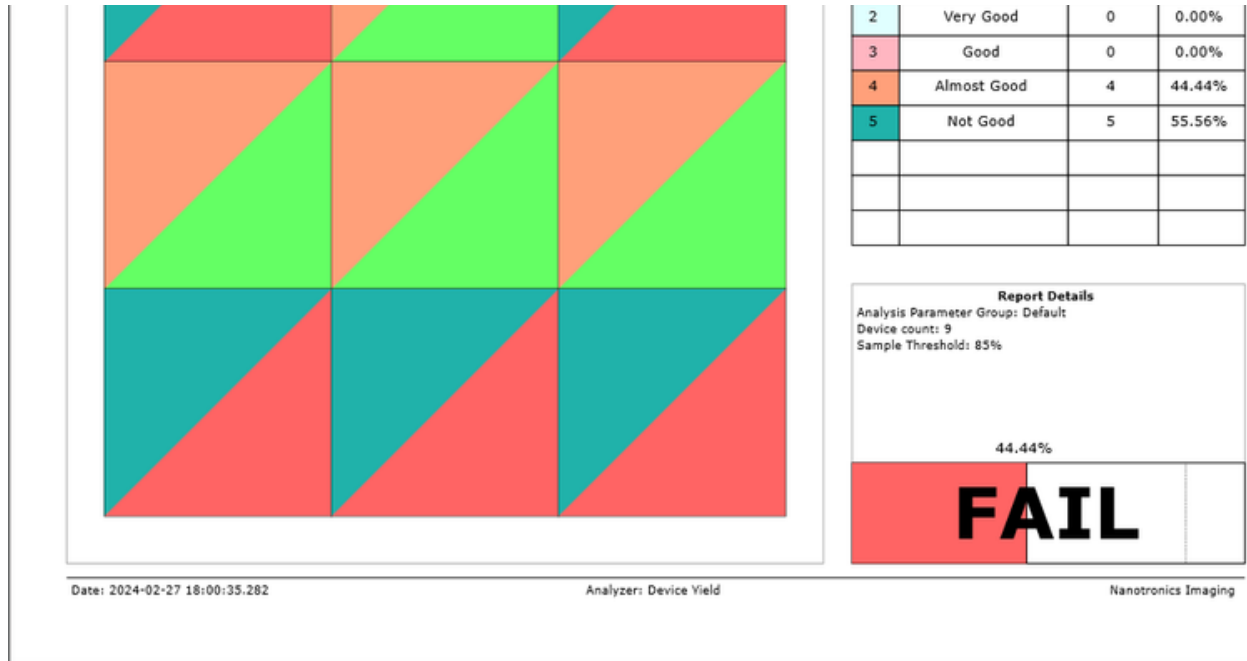
After successfully running Device Yield, the analysis results can be viewed in nView. The Device Yield report can be viewed by accessing **nView > View > Device Yield Report** or **Ctrl+ 7**.

The Device Yield report has three different views that can be seen by toggling **Pass/Fail** and **Bins**, seen in the image below.

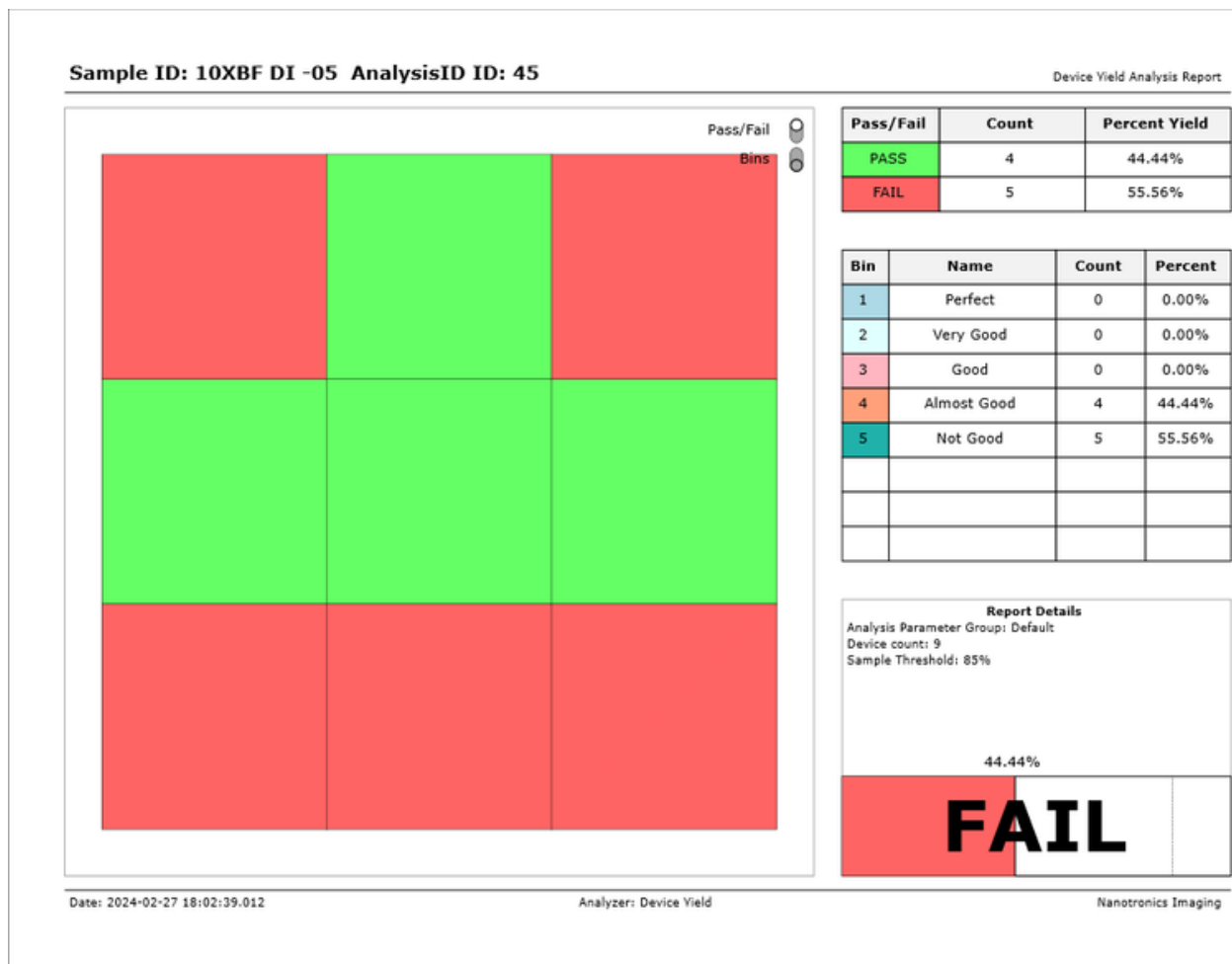


In the next image, both the **Pass/Fail** and **Bins** views are turned on. Each square represents a device, with the color in the top left corner of each device representing the bin associated with the device, and the color in the bottom right corner representing the pass/fail status of the device.





Bins can be toggled off to only show the pass/fail status of each device.



Similarly, **Pass/Fail** can be toggled off to show the device bins only.



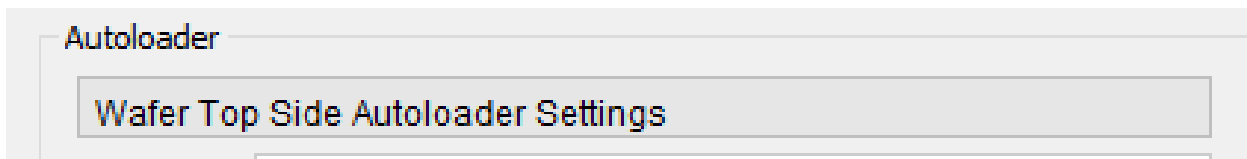
Multi-Sided Sample Scanning

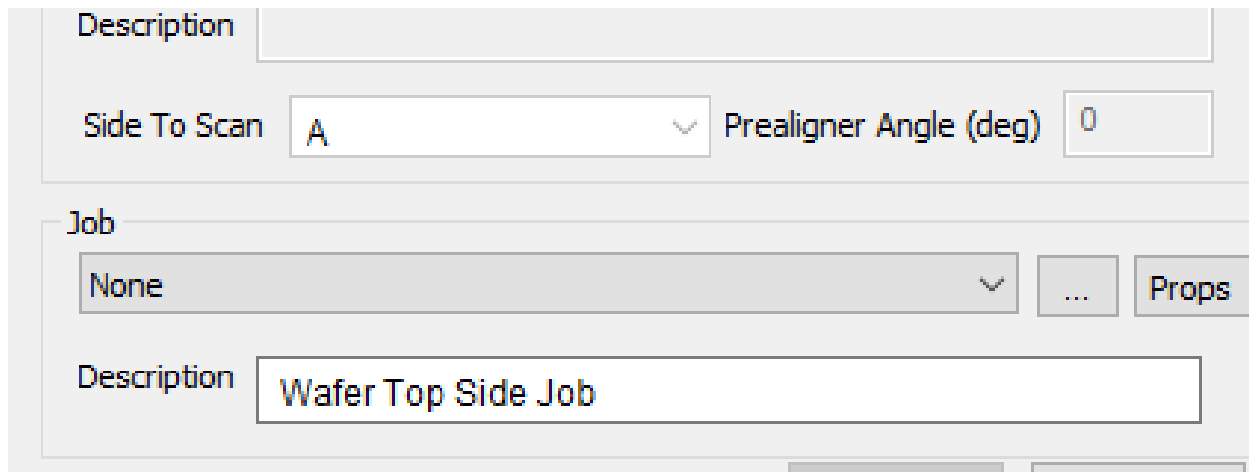
With a few hardware upgrades including the addition of a robot for handling samples, nSpec systems are capable of performing multi-sided sample scans.

In order to perform a multi-sided sample scan, nSpec needs to an autoloader to perform sample handling. During initial nSpec setup, a Nanotronics Technical Support Engineer will save autoloader calibration settings for handling and scanning each side of a multi-sided sample. For example, for a double-sided wafer, the engineer would save different autoloader settings for the top and bottom sides of the wafer.

These settings will populate in the **Autoloader** section of the **Scan Settings** dialog. Running a multi-sided sample scan necessitates a group job, and each side of the sample needs its own job or scan settings.

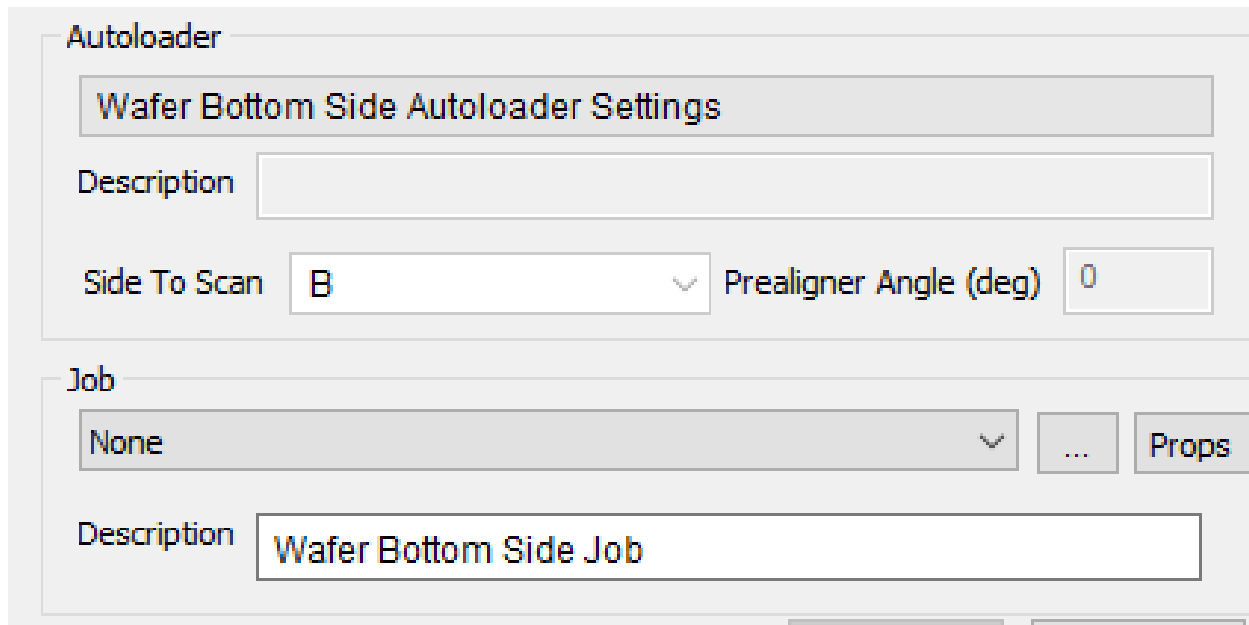
To continue using the previous example of a double-sided wafer, when setting up a job for the wafer's top side, the autoloader calibration setting for the top side of the wafer is set to **Side To Scan** value A.





The screenshot shows a configuration window for a job. At the top, there is a 'Description' field. Below it, 'Side To Scan' is set to 'A' and 'Prealigner Angle (deg)' is set to '0'. The 'Job' section has a dropdown menu set to 'None', a '...' button, and a 'Props' button. The 'Description' field for the job is 'Wafer Top Side Job'.

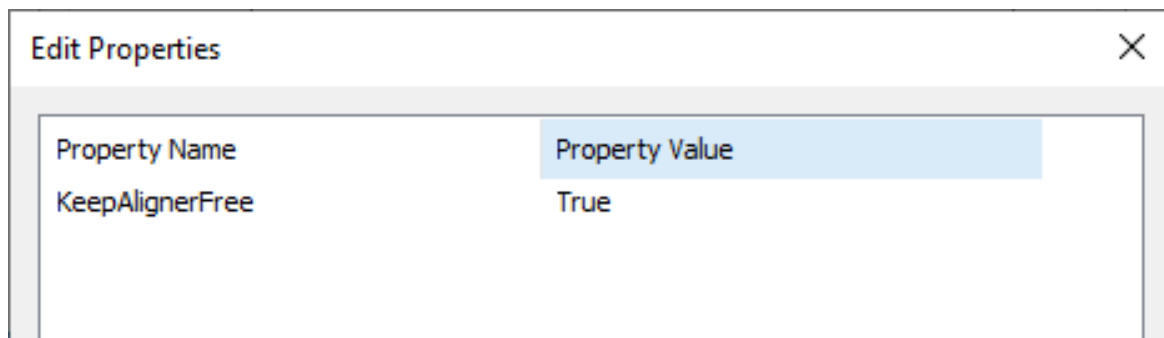
Similarly, the autoloader calibration setting for the wafer's bottom side is set to **Side To Scan** value B.



The screenshot shows an 'Autoloader' configuration window. The title bar says 'Autoloader' and the main title is 'Wafer Bottom Side Autoloader Settings'. Below this, there is a 'Description' field. 'Side To Scan' is set to 'B' and 'Prealigner Angle (deg)' is set to '0'. The 'Job' section has a dropdown menu set to 'None', a '...' button, and a 'Props' button. The 'Description' field for the job is 'Wafer Bottom Side Job'.

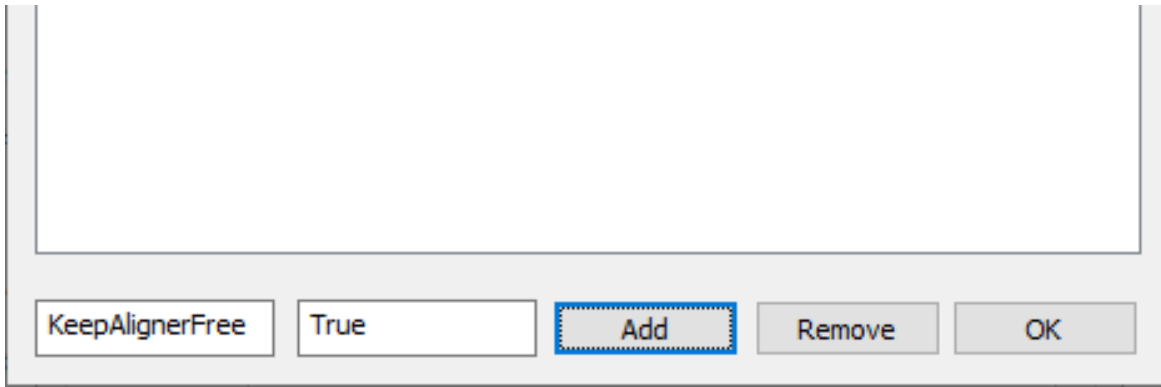
A and B are the default values for the **Side To Scan** setting, but are arbitrary and can be updated in `Calibration.db`.

Additionally, all jobs for multi-sided samples must be saved with Job Property `KeepAlignerFree` set to `True`.



The screenshot shows a dialog box titled 'Edit Properties'. It contains a table with two columns: 'Property Name' and 'Property Value'. The table has one row with 'KeepAlignerFree' as the property name and 'True' as the property value.

Property Name	Property Value
KeepAlignerFree	True



The job properties dialog is access via the **Props** button in the **Job** section of **Scan Settings**.

After the job for each sample side is saved, all jobs should be added to a Job Group in the order that they should be performed.

New Features

Highlights

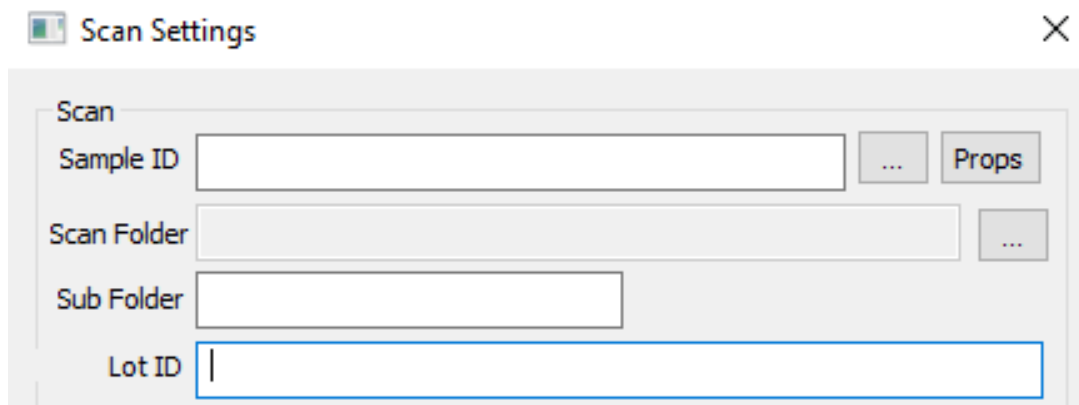
NSPEC-6905: Cropped Images Analysis migrated to use standard method of referencing previous analyses

When setting up a Cropped Images Analysis, previous analyses will be referenced with parameters Base Analysis ID Mode and Base Analysis ID. More about this reference system can be read in **Appendices: Post - Analysis Referencing**.

NSPEC-7957: Add Lot ID to Scan Dialog

Overview

Lot ID is an optional field that can be set in the **Scan Settings** dialog. It is saved to the database under Scan Properties.



Lot ID Input Requirements

- Must be 260 characters or less
- Cannot contain any of the following characters: / \ . : " | > < * ?

- Cannot contain tab characters
- Cannot end in a space
- Cannot contain any of the following reserved names: NULL, CON, PRN, AUX, NUL, COM0, COM1, COM2, COM3, COM4, COM5, COM6, COM7, COM8, COM9, COM¹, COM², COM³, LPT0, LPT1, LPT2, LPT3, LPT4, LPT5, LPT6, LPT7, LPT8, LPT9, LPT¹, LPT², and LPT³
- Empty strings are interpreted as “No Lot ID Specified”

Additionally, **Lot ID** is **not** case sensitive.

NSPEC-8003: Cropped Images Analysis includes Overlay File Configuration Parameter

Overlay Configuration File is an optional parameter for Cropped Images Analysis, which allows users to apply an customizable overlay to exported images. The parameter input type is a nJson file.

Analysis Parameters

Analysis: Cropped Images

Group: Default

Description: Cropped Image Generator Params

P.	Name	Type	Low	PV	High	Default	Picklist	
01	Base Analysis ID Mode	PickList		Relative		Relative	Absolute, Re...	S
02	Base Analysis ID	String						S
03	Image Output Limit Method	PickList		Absolu...		Absol...	Percentage, ...	T
04	Image Output Limit Percentage	Double	0	1	1	1		R
05	Image Output Limit Number	Long	1	1	2147483...	1		lq
06	Image Size Limit Method	PickList		All		All	All, Larger E...	T
07	Image Size Limit	Double	0	10	200	10		S
08	Ranking Criterion	PickList		Rando...		Rand...	Random (32...	T
09	Ranking Criterion Ordering	PickList		Ascend...		Ascen...	Ascending, ...	U
10	Seed Random Number Generator	PickList		FALSE		FALSE	TRUE, FALSE	U
11	Random Number Generator Seed	Long	-214...	0	2147483...	0		U
12	Crop Padding (Pixels)	Long	0	20	100	20		P
13	Export File Full Path	StringBy...		\$(PATH...		\$(PATH_T...		E
14	Export Many Images As Single File	PickList		FALSE		FALSE	TRUE, FALSE	If
15	Overlay Configuration File	PickFile				C:\Na... nJson		If

<
>

Reload
Default
Delete

Save
Save As

Close

Below is an example nJson file for the Overlay Configuration File parameter.


```
{
  "Overlay Setting" :
  {
    "semver": "1.0.0",
    "viewport_dimensions": {
      "rows": 150,
      "output_width" : 512,
      "columns": 150
    },
    "background_color": "#666666",
    "sidebar_parameters": {
      "text_line_vertical_padding": 6,
      "text_line_horizontal_padding": 5,
      "sidebar_text_scale": 1,
      "sidebar_text_color": "#FFFFFF",
      "sidebar_width": 450,
      "sidebar_position": "left"
    },
    "scale_section_parameters": {
      "bar_top_padding": 10,
      "bar_height": 25,
      "bar_bottom_padding": 10,
      "horizontal_padding_between_label_and_bar": 10,
      "horizontal_padding_between_bar_and_edge_of_section": 10,
      "scale_bar_color": "#EEEEEE",
      "scale_section_justification": "right",
      "scale_bar_label_parameters": {
        "scale_label_text_color": "#FFFFFF",
        "scale_label_text_scale": 0.8,
      }
    }
  }
}
```

The above Overlay Configuration File produces the following cropped defect image with overlay. The cropped defect image dimensions, background color, sidebar defect information, and scale and their relative placements are all fully customizable.

```
Defect ID: 560955
Defect Area: 14.03 um sq
Defect Length: 4.461 um
Defect Width: 4.014 um
```





`semver` is shorthand for semantic version, and refers to the version of the defect overlay schema used. Required.

`viewport_dimensions` sets the size of the viewport containing the cropped defect. Required.

`background_color` is the background color for the viewport, only used if the viewport size is larger than the cropped defect tile size. Required.

`sidebar_parameters` sets the size, font formatting, padding, and relative position of the image overlay sidebar, which contains defect information, including the ID, area, length, and width. Required.

`scale_section_parameters` sets the size and padding for the image overlay scale bar. Optional.

NSPEC-8098: Naming Export Files with String Templates

Export files from Custom Exporter, Cropped Images, and Report Summary Image Export now can be named with string template via the newly added Export File Full Path parameter.

- [Overview](#)
- [Example String Template Usage](#)
- [Table of Supported String Templates](#)

Overview

Users can dynamically name export files using string templates via the Export File Full Path parameter, available for Custom Exporter, Cropped Images, and Report Summary Image Export analyses.

Analysis Parameters

Analysis: Custom Exporter

Group: Default

Description: Custom Exporter

	P.△	Name	Type	Low	PV	High	Default	Picklist
01		Base Analysis ID Mode	PickList		Relative		Relative	Absolute, Re.
02		Base Analysis ID	Custom					

02	Base Analysis ID	String				
03	Export Setting	PickFile	\\WSAMZ...	...	\\WSAM...	nJson
06	Export Type	PickList	CSV		CSV	CSV,KLARF,...
07	Export File Full Path	StringBy...	\$(PATH_TH...		\$(PATH_THI...	

<
>

Reload
Default
Delete

Save
Save As
Close

Example String Template Usage

Here is an example of string template usage for the Export File Full Path parameter in Cropped Images Analysis.

The following input: `$(PATH_THIS_ANALYSIS)\$(SAMPLE_ID)_$(SCAN_ID)_$(ANALYSIS_ID)_$(ANALYSIS_SHORTNAME)\defect_$(DEFECT_ID)`

Results in the following export path:

C:
 \Scans\ExampleSampleID\Analysis\Analysis_001_002\ExampleSampleID_001_002_Cropped Image Generator\defect_12345



Table of Supported String Templates

Currently, under the Post-Analysis Support column, “All” refers to the three analyses that this feature has been implemented for: Custom Exporter, Cropped Images, and Report Summary Image Export.

Variable Name	Description	Post-Analysis Support
PATH_DB	Folder that Contains the image database ex: C:\Scans\	All
PATH_SCAN	Scan destination folder, the folder that contains “Scan_001” etc ex: C:\Scans\SampleID\	All
PATH_ANALYSIS	The incremented analysis destination folder within the scan folder. ex: C:\Scans\SampleID\Analysis\	All
PATH_THIS_ANALYSIS	The incremented analysis destination folder within the scan folder.	All

	ex: C : \Scans\SampleID\Analysis\Analysis_\${SCAN_ID}_\${ANALYSIS_ID}	
SAMPLE_ID	Parent scan's sample ID	All
MACHINE_NAME	Name of machine (Device name in PC's settings System About)	All
JOB_NAME	Name of job that created parent scan	All
SCAN_ID	Parent scan's ID	All
ANALYSIS_GROUP_ID	Parent group analysis' ID (error if analysis is not part a group analysis)	All
ANALYSIS_GROUP_NAME	Parent group analysis' name (error if analysis is not part a group analysis)	All
ANALYSIS_ID	Analysis' ID	All
ANALYSIS_SHORTNAME	Analysis' IntName as shown in <i>Main View Utility -> Edit Analyses...</i>	All
ANALYSIS_DISPLAYNAME	Analysis' Name as shown in <i>Main View Utility -> Edit Analyses...</i>	All
DATETIME	Analysis' completion time with format: %Y%M%DT%h%m%s	All
TARGET_ANALYZER_DISPLAYNAME	Target analysis' Name as shown in <i>Main View Utility -> Edit Analyses...</i>	All
TARGET_ANALYZER_SHORTNAME	Target analysis' IntName as shown in <i>Main View Utility -> Edit Analyses...</i>	All
TARGET_ANALYSIS_DATETIME	Target analysis' completion time with format: %Y%M%DT%h%m%s	All
TARGET_ANALYSIS_ID	Target analysis' ID	All
DEFECT_IMAGE_ID	Cropped defect's parent image ID	Cropped Images
DEFECT_DEVICE_ID	Cropped defect's parent device ID	Cropped Images
DEFECT_ID	Cropped defect's ID	Cropped Images
DEFECT_CLASSIFICATION	Cropped defect's classification name (N/A if defect is not classified)	Cropped Images
TARGET_ANALYSIS_DESCRIPTION	Reporting image export's Report Type	Reporting Summary Image Export





New Features Changelog

T	Key	Release Notes Summary
	NSPEC-6453	Scan Database now includes SampleRegions data
	NSPEC-6455	Implement Device Yield Analysis
	NSPEC-6634	Automatic upgrade of scan databases to new database schema
	NSPEC-6905	Cropped Images Analysis migrated to use standard method of referencing previous analyses
	NSPEC-7124	Device Yield Report can be selected in nView
	NSPEC-7909	Overlay in Cropped Images Analysis output now contains Device ID
	NSPEC-7957	Add Lot ID to Scan Dialog
	NSPEC-7979	Update Sample ID, Scan folder, Subfolder, and Lot ID to be Editable at the Operator Level
	NSPEC-8003	Cropped Images Analysis includes Overlay File Configuration Parameter
	NSPEC-8057	Ability to add Custom Filename Suffix to Report Summary Image Export
	NSPEC-8098	Naming Export Files with String Templates

11 issues

Bug Fixes

Changelog

T	Key	Release Notes Summary	Affected Releases
	NSPEC-7976	When running any analyzer that has an nJson setting file as a parameter, the setting file contents are not saved to parameter groups	0.23.0.0, 0.23.1.0
	NSPEC-8049	Cassette OCR group jobs with empty sample IDs try to pick up every wafer	0.23.1.1
	NSPEC-8200	Unable to operate RGBW Transmitted Light Source	0.23.1.0, 0.23.0.2, 0.23.1.1
	NSPEC-8220	ProgramOptions Compare utility not working	0.23.1.0, 0.23.1.1



NSPEC-8223

Error running GEM cassette jobs with skipped slots

0.23.1.0

5 issues

Appendices

Post-Analysis Referencing

There are a few nSpec Analyses that reference other, previously run analyses. As of version v0.23.0.0 this includes **Custom Reporter**, **Custom Exporter**, and **Defect Classifier** post-analyses.

The fields pertaining to prior referencing include:

Base Analysis IDs Mode	Absolute, Relative	The user selects the nature of analysis ID(s) to be specified in the following field. Relative = most recently run, while absolute = the numeric integer reference of a prior run analysis
Base Analysis IDs		<p>Comma separated set of IDs. If <i>Mode</i> is Absolute, it is a set of integers (example: "1,2").</p> <p>If <i>mode</i> is Relative, it is a set of source analysis names. For example, Device Inspection, Gen IV Analysis means that first view (DI from the nJson) will be the Defects of the last analysis of type Device Inspection, while the second source analysis (GenIV from the nJson) will be populated with the Defects from the most recently run Gen IV Analysis.</p>

Relative Reference Notes

When using Relative mode, you also have the option to reference any previously run analysis, by adding a numeric indicator at the end of the analysis name. Let's say an image database ran 3 analyses:

1. Gen IV AI Analysis (parameter set 1)
2. Gen IV AI Analysis (parameter set 2)
3. Device Inspection

To point at #2 and #3, you would input the following for the **Base Analysis IDs field**:

```
"DeviceInspection, Gen IV Analysis"
```

To point at #1 and #3, you would input the following for the **Base Analysis IDs field**:

```
"DeviceInspection, Gen IV Analysis.2"
```

The .2 at the end of Gen IV Analysis informs nSpec to reference the second most recently run Gen VI Analysis. In this way, you have complete flexibility to point at any previously run analysis using both absolute or relative modes.

If you wish to reference the most recent analysis, there is no need to add the “.1” after the analysis internal name reference. This means that “DeviceInspection” is the same as “DeviceInspection.1”

Analysis Reference List

The user-facing name of an analysis is not necessarily identical to the ID that should be used to reference that analysis.

Analysis Name (user facing)	Analysis Reference ID (IntName)
AI Analysis	AIAnalysis
Basic Selection (Contrast, Morphological Range)	nrad_rangeselect
Basic Selection (Intensity, Exclusive)	nrad_intensityselectexclusive
Basic Selection (Intensity, Inclusive)	nrad_intensityselectinclusive
Crack Detection V2	CrackDetectionV2
Custom Exporter	Custom Exporter
Custom Reporter	Custom Reporter
Device Inspection	DeviceInspection
Die Yield	nrad_dieyield
Dislocation Defect Analyzer	Dislocation
Gen IV AI Analysis	Gen IV Analysis
High Resolution Mosaic Generation	High Resolution Mosaic Generation Analysis
Surface Scattering Analysis	Surface Scattering Analysis
Utility: Launch Executable	LaunchScript

When to use Absolute, and when to use Relative

When testing a new script, it might be useful to use absolute mode. However, in the majority of cases, we believe that relative mode is most likely to be the optimal mode, especially when running in production. Consider a case where you scan a wafer and perform a Group Analysis on this scan. The first time, using absolute references to point at the 1st and 2nd analyses may work fine. But what happens if you run that group analysis a second time? Now, the post-analysis is still performed on the first and second analyses ever performed, rather than the ones most recently performed as part of that new analysis group.

When running in production, we recommend always using **relative mode**