Compare Metrics Against Software Quality Objectives

After generating and viewing metrics from your analysis results, you can review the results in greater detail.

To focus your review, you can:

1. Define quality objectives that you or developers in your organization must meet.
2. Apply the quality objectives to your analysis results.
3. Review only those results that fail to meet those objectives.

Apply Predefined Objectives to Metrics

By default, the software quality objectives are turned off. To apply quality objectives:

1. In the Polyspace® Metrics interface, open the metrics page for a project.
2. From the **Quality Objectives** list in the upper left, select ON.
   A new group of **Software Quality Objectives** columns appears.
   - The **Overall Status** column shows the last used quality objective level to generate a status of **PASS** or **FAIL** for your results.
   - The **Level** column shows the quality objective level.
     To change your quality objective level, in this column, select a cell. From the drop-down list, select a quality level. For more information, see *Bug Finder Quality Objective Levels.*
3. For files with an **Overall Status** of **FAIL**, to see what causes the failure, view the entries in the other **Software Quality Objectives** columns. The failing levels are marked red.
   If the ⚠ icon appears next to the status, it means that Polyspace does not have enough information to compute the status. For instance, if you specify BF-QO-1, certain coding rules must be review. But, if you do not check coding rules during the analysis, Polyspace cannot determine whether your project satisfies the coding rule objectives specified in BF-QO-1.
4. To investigate the failing quality objectives, select the entries marked red for more details.
5. On the **Code Metrics, Coding Rules**, or **Bug-Finder** tab,
   a. Select the red column entries to download the results.
   b. Review the violations and fix or justify the results.
      See *Review and Fix Results* or *Annotate and Hide Known or Acceptable Results.*
   c. Upload your new justifications to the Polyspace Metrics web dashboard.
6. After your review, in the Polyspace Metrics interface, click  to view the updated metrics.
   If you change your code, to update the metrics, rerun your analysis and upload the results to the Polyspace Metrics repository. If you have justifications in your previous results, import them to the new results before uploading to the repository.

Bug Finder Quality Objective Levels

The Bug Finder Quality Objectives or BF-QOs are a set of thresholds against which you can compare your Bug Finder analysis results. These objectives are adapted from the Polyspace Code Prover™ *Software Quality Objectives* (Polyspace Code Prover). You can develop a review process based on the Quality Objectives.

You can use a predefined BF-QO level or define your own. Following are the predefined quality thresholds specified by each BF-QO.

- *BF-QO Level 1*
- *BF-QO Level 2 and 3*
- *BF-QO Level 4*
- *BF-QO Level 5*
- *BF-QO Level 6*
- *BF-QO Exhaustive*

Customize Software Quality Objectives

Instead of using a predefined objective, you can define your own quality objectives and apply them to your project.

1. Save the following content in an XML file. Name the file **`Custom-BF-QO-Definitions.xml`**.

```
<?xml version="1.0" encoding="UTF-8"?>
<MetricsDefinitions>

    <SQO ID="Custom-BF-QO-Level" ApplicableProduct="Bug Finder"
ApplicableProject="My _ Project">
        <comf>20</comf>
        <path>80</path>
        <goto>0</goto>
        <vg>10</vg>
        <calling>5</calling>
        <calls>7</calls>
        <param>5</param>
        <stmt>50</stmt>
        <level>4</level>
        <return>1</return>
        <vocf>4</vocf>
        <ap _ cg _ cycle>0</ap _ cg _ cycle>
        <ap _ cg _ direct _ cycle>0</ap _ cg _ direct _ cycle>
        <Num _ Unjustified _ Violations>Custom _ MISRA _ Rules _ Set</
Num _ Unjustified _ Violations>
        <Num _ Unjustified _ BF _ Checks>Custom _ BF _ Checks _ Set</
Num _ Unjustified _ BF _ Checks>
    </SQO>
```

```
    <CodingRulesSet ID="Custom _ MISRA _ Rules _ Set">
        <Rule Name="MISRA _ C _ 5 _ 2">0</Rule>
        <Rule Name="MISRA _ C _ 17 _ 6">0</Rule>
    </CodingRulesSet>

    <BugFinderChecksSet ID="Custom _ BF _ Checks _ Set">
        <Check Name="UNREACHABLE">0</Check>
        <Check Name="USELESS _ IF">0</Check>
    </BugFinderChecksSet>

</MetricsDefinitions>
```

2. Save this XML file in the folder where remote analysis data is stored, for example, `C:\Users\JohnDoe\AppData\Roaming\Polyspace _ RLDatas`.
   If you want to change the folder location, select **Metrics** > **Metrics and Remote Server Settings.**
3. To make the quality level `Custom-BF-QO-Level` applicable to a certain project, replace the value of the ApplicableProject attribute with the project name.
   If you want the quality objectives to apply to all projects, use `ApplicableProject=""`.
4. For specifying coding rules, begin the rule name with the appropriate string followed by the rule number. Use _ instead of a decimal point in the rule number.

Rule	String	Rule numbers
MISRA C: 2004	`MISRA _ C _`	MISRA C:2004 and MISRA AC AGC Coding Rules
MISRA C: 2012	`MISRA _ C3 _`	MISRA C:2012 Directives and Rules
ISRA C++	`MISRA _ Cpp _`	MISRA C++ Coding Rules
JSF® C++	`JSF _ Cpp _`	JSF C++ Coding Rules
Custom coding rules	`Custom _`	Custom Coding Rules

5. For specifying defects, use the defect acronym. For defect acronyms, see the individual *defect reference pages.*
6. After you have made your modifications, in the Polyspace Metrics interface, open the metrics for your project. From the **Quality Objectives** list in the upper left, select ON.
7. On the **Summary** tab, select an entry in the **Level** column. For the project name that you specified, your new quality objective **Custom-BF-QO-Level** appears in the drop-down list.
8. Select your new quality objective.
   The software compares the thresholds you had specified against your results and updates the **Overall Status** column with **PASS** or **FAIL**.
9. To define another set of custom quality objectives, add the following content to the `Custom-BF-QO-Definitions.xml` file:

```
<SQO ID="Custom-BF-QO-Level _ 2" ParentID="Custom-BF-QO-Level"
ApplicableProduct="Bug Finder" ApplicableProject="My _ Project">
  ...
</SQO>
```

- **`ID`** represents the name of the new set.
  You cannot have the same values of **`ID`** and **`ApplicableProject`** for two different sets of quality objectives. For example, if you use **`ID="Custom-BF-QO-Level"`** for two different custom sets, and **`ApplicableProject`** is either **`My _ Project`** or `""` for both sets, you see the following error:
  `The SQO level 'Custom-BF-QO-Level' is multiply defined.`
- **`ParentID`** specifies another level from which the current level inherits its quality objectives. In the preceding example, the level **`Custom-BF-QO-Level _ 2`** inherits its quality objectives from the level **`Custom-BF-QO-Level.`**
  If you do not want to inherit quality objectives from another level, omit this attribute.
- `...` represents the additional quality thresholds that you specify for the level **`Custom-BF-QO-Level _ 2.`** The quality thresholds that you specify override the thresholds that **`Custom-BF-QO-Level _ 2`** inherits from **`Custom-BF-QO-Level`**. For instance, if you specify **`<goto>1</goto>`**, this objective overrides the threshold specification **`<goto>0</goto>`** of **`Custom-BF-QO-Level.`**

Elements in Custom Quality Objective Files

- *HIS Metrics*
- *Non-HIS Metrics*

The following tables list the XML elements that can be added to the custom BF-QO file. The content of each element specifies a threshold against which the software compares analysis results. For each element, the table lists the metric to which the threshold applies. Here, HIS refers to the Hersteller Initiative Software.

HIS Metrics

Element	Metric
`comf`	`Comment Density`
`path`	`Number of Paths`
`goto`	`Number of Goto Statements`
`vg`	`Cyclomatic Complexity`
`calling`	`Number of Calling Functions`
`calls`	`Number of Called Functions`
`param`	`Number of Function Parameters`
`stmt`	`Number of Instructions`
`level`	`Number of Call Levels`
`return`	`Number of Return Statements`
`vocf`	`Language Scope`
`ap _ cg _ cycle`	`Number of Recursions`
`ap _ cg _ direct _ cycle`	`Number of Direct Recursions`
`Num _ Unjustified _ Violations`	Number of unjustified violations of coding rules specified by entries under the element `CodingRulesSet`
`Num _ Unjustified _ BF _ Checks`	Number of unjustified defects of types specified by entries under the element `BugFinderChecksSet`

Non-HIS Metrics

Element	Description of metric
`fco`	`Estimated Function Coupling`
`flin`	`Number of Lines Within Body`
`fxln`	`Number of Executable Lines`
`ncalls`	`Number of Call Occurrences`