# SqlSkim Developer’s Reference

## Authoring New Rules

To author a new SqlSkim rule, follow this sequence of steps:

1. Identify a suitable check to author, by finding a rule specification that’s ready to go in the SqlSkim github.com issue or by proposing a new check as an issue.
2. Allocate a rule id for the check and author other rule metadata.
3. Create a shell rule file for the check, following the code as shown in ExampleRule.cs (located in the Rules project directory).
4. [Optional] Identify one or more configurable options required for the check.
5. Create one or more tests for the check as TSQL code. These tests should provoke complete code coverage for the rule Analyze method, including both passing and failing test cases.
6. Implement the rule. Test. Win.

### Identifying a Rule to Author

Linters provide light-weight code inspection, generally using fairly simple analysis techniques. A good check is performant and low noise (in the sense that if a user enables a rule, she generally takes fixes for its results). It is fine to implement style checks. These rules should be configurable, however, to allow users to tune the analysis to their team’s guidelines. Because linters are typically integrated into build and code authoring environments, they are not suitable for authoring ‘review’ rules (which flag patterns that do not perform real detection but mark code that should be inspected by a developer for a possible problem).

Review the SqlSkim issues for open rule ideas. These are marked with the ‘rule-idea’ label. By convention, rule idea issue titles are prefixed with ‘Rule Idea:’ as well.

Alternately, you can open a new issue with your own rule idea. A finished rule idea will contain descriptive information for the rule, a thorough description of its behavior, and sample TSQL patterns for pass/fail cases.

Author Rule Metadata

SqlSkim rule metadata conforms to the [SARIF](https://github.com/sarif-standard/sarif-spec) standard. Sarif (the Static Analysis Results Interchange Format) is a public standard originally proposed by Microsoft for persisting and exchanging static analysis tool results and rules metadata. Rule metadata mostly includes user-facing strings that document the check or issues that it flags. Producing good guidance (that is consistent in style with the rest of the tool) is important for usability. Before implementing a check, author and review the following rule metadata:

* Rule id: an opaque identifier for a check that, once assigned, never changes. SqlSkim ids are allocated in RuleIds.cs (in the SqlSkim.Rules project). The member names of this const string class should exactly match the next piece of rule metadata, the short name.
* Name: a rule name is a concise rule identifier that summarizes the check. The rule type name itself is the SqlSkim rule name (and therefore can’t include characters that can’t be compiled in class names). Rule names provide a concise abstract of a check’s prescriptive guidance. Rules that advise against certain practices are often prefixed with ‘DoNot’ or ‘Avoid’ (depending on whether developers must or should avoid a certain pattern, respectively).
* Full description: each check contains a full description. By convention, this description is