**2 Guidelines  
2.0 General**

2.0.0 Coding Language: C#

**2.1 Modern code**2.1.0 Avoid #define -> Use constants, enums or functions instead2.1.1 Prefer exceptions over error codes for error reporting -> Good exception-safe code is easier to read and less error prone in general.

**2.2 Coding style**

2.2.0 Use clear and meaningful names  
 -> Make it clear what the thing is supposed to do, only use well-established abbreviations  
2.2.1 Prefer clarity over cleverness  
 -> Put the focus on obvious code, not on short code.  
2.2.2 Do not omit the curly brackets around loop bodies and conditional statements  
 -> Have brackets even if it’s just about one single line of code.  
 -> Exception: single line body with only “return”, “break” or “continue”  
 i.e.: if(a) continue;

2.2.3 No Egyptian brackets:

if(a){  
}

Instead:

If(a)  
{  
}  
  
2.2.4 Keep the style consistent within an existing module  
 -> Use whatever style that is already in place, as long as it is consistent.  
2.2.5 Qt style for new modules

* Pascal case: functions, class, struct, enum
* Camel case: variables

2.2.6 Setter, getter -> SetValue(value), GetValue()  
2.2.7 Prefix members and static variables  
 -> Use “m\_” for members and “s\_” for static members.  
2.2.8 Comments: Multiline comments for comments longer than 2 lines, single line for everything   
 else  
2.2.9 Keep it in English  
 -> Use English for all names, comments, log outputs etc.

**2.3 Code structure**  
2.3.0 Group related things together  
 -> Use classes or namespace to make interdependencies clear  
2.3.1 Don’t use structs, use class instead  
2.3.2 Layout your classes from public to protected to private

**2.4 API & Implementation**  
2.4.0 Think about thread-safety  
 -> Thread-safety is not a trivial task and has to be considered when designing an API.  
2.4.1 Avoid global state  
 -> Don’t use global variables, avoid static variables, avoid singletons.  
2.4.2 Avoid deep nesting, keep functions short and simple  
 -> Functions shouldn’t span more than about 100 lines of code.  
2.4.3 Limit the scope of types and variables  
 -> Keep the scope of everything as narrow as possible.  
2.4.4 assert internal assumptions and invariants  
 -> assert everything that has to be true and really should be true.  
2.4.5 Always initialize members and variables  
 -> Initialize variables right at definition, members in the constructor.