Summary:
The main reason for this document is to provide the support organization with a few arguments against the use of macros – especially since members of the support organization will encounter some (or all) of these arguments when discussing these issues with representatives from R&D. It is important to note that this is not an attempt at an outright ban of macros, but rather some reflections you should be aware of the minute you include a single macro statement in your application. From that point on, you are responsible for migrating these macros with each new version and so on, and it is a fact that most of our customers frustrations have to do with poorly designed macros and in some cases well-designed macros in inappropriate situations.

Conclusion:
What we are striving for is a heightened awareness when it comes to macros and what may work with a few thousand records does not necessarily scale very well when macros are involved and the problems tend to manifest themselves and become more serious when larger datasets are involved. It is also important to note that certain events may only be captured through the use of macros and for this reason it may be difficult to avoid macros altogether. The R&D department always strives to incorporate as much of this functionality as possible as basic QlikView functionality, thus limiting the use of macros in the long run – however, as previously stated, certain events are difficult to catch except from an outside macro…

Detail - Guidance on macros:
1) Running a macro automatically deletes all caches, undo-layout buffers and undo-logical operation buffers and this in general has a very large negative impact on performance as experienced by the clients. The reason for deleting the caches etc is that it is possible to modify properties and selections from the macros, thus opening up possible conflicts between the cached state and the modified state from a macro, and these conflicts will almost always crash or hang the clients (worst case; hang or crash the server as well).

2) The macros themselves are executed at VBScript level while QlikView, in general, is executed at assembler level which is thousands of times faster by default. Furthermore, the macros are single threaded, synchronous processing as opposed to QlikView which is asynchronous and heavily threaded, and this causes the macros to effectively interrupt all calculations in QlikView until finished. Thereafter, QlikView has to resume all interrupted calculations, which is a delicate process and very much a source (at least historically) for deadlocks (i.e. QlikView freezes while the macro is still running, without any possibility that the macro will be finished).

3) While QlikView is increasingly optimized in terms of performance and stability, the macros will always maintain their poor performance, and the gap between genuine QlikView functionality and the macros will continue to increase, making macros less and less desirable from a performance point of view. This fact, combined with the above fact that the macros tend to undermine all optimizations made in QlikView, calls for severe negative tradeoffs as soon as macros become an integral part of any larger application.

4) Macros are of secondary nature when it comes to QlikView functionality - first all internal basic QlikView functions are run and tested and thereafter the macros are run and tested which effectively means that macros will never have the same status or priority as basic QlikView functionality - always consider macros as a last resort but nothing much else. Since the automation API reflects the basic QlikView in terms of object properties etc, the macro content may actually change between versions making this a very common area for migration issues. Once a macro is incorporated in an application, this application has to be revisited with each new version in order to make sure that the macros were not affected by any structural changes in QlikView, and this makes macros extremely burdensome in terms of maintenance.

5) Only a subset of macros will work in a server environment with thin clients (Java and Ajax) since local operations (copy to clipboard, export, print etc.) are not supported, though some of these have a server-side equivalent (e.g. ServerSideExport etc.) that is very expensive in terms of performance with each client effectively affecting the server performance in a negative way.

Given all of the above, macros should not be part of any ‘recommended’ QlikView design pattern!