

Converting Legacy Models (sct)

Since KIELER SCCharts version 0.13 textual SCCharts are no longer written in sct files but in **sctx**. These file extensions allow separation between the two syntaxes, as well as conversion of the legacy models. KIELER still has rudimentary support for this legacy syntax. When opening an sct file the editor and diagram view still work but shows warnings. None of the KIELER features, such as Compilation or Simulation support sct files. However, in the context menu (right click) of an sct file there is an *Convert to SCTx* option to convert the file into the new sctx syntax.

In some cases the produced files still need manual adjustment to match the new syntax.

Known issues:

- References to other SCCharts files are no longer added implicitly. Imports of other SCCharts must be added manually.
- Line comments (//) are lost during the conversion. Semantic comments (/**) will remain.

In general, we advice to convert all sct into sctx. To ease this process the *Convert to SCTx* action can also be invoked on folders, resulting in a conversion of all contained (recursively) sct files.

The screenshot displays the KIELER IDE interface for a project named 'product-kieker-tutorial - Example/abro.sct - KIELER'. The left pane shows the source code of a VHDL layout for a statechart named 'ABO'. The code includes input/output declarations for boolean signals A, B, O1, and O2, and a state machine structure with 'initial', 'WaitA', 'DoneA', 'WaitB', 'DoneB', and 'GotAB' states. A context menu is open over the code, with the 'Convert to SCTx' option highlighted. The right pane shows the corresponding state machine diagram, which is a legacy SCChart diagram. A warning banner at the top of the diagram view states: 'Legacy SCCharts diagram. You are using a deprecated textual version of SCCharts! This language version (.sct) is no longer supported by the KIELER tool. Please convert the file into the new '.sctx' format by performing the 'Convert into SCTx' action in the file's context menu.' The diagram shows a sequence of states: 'Init' leads to 'WaitAB' (with guard '/ O1 = false; O2 = false'), which leads to 'WaitA' (with guard 'A / B = true; O1 = true'), which leads to 'DoneA', which leads to 'WaitB' (with guard 'B / O1 = true'), which leads to 'DoneB', which leads to 'GotAB' (with guard '/ O1 = false; O2 = true'). The bottom pane shows the 'KIELER Compiler' and 'Simulation' tabs, with the 'Identity' transformation system selected.