Naming Conventions

- **Symbolic Name in XML Configuration**: checkNamingConventions
- **Dependant Tags in XML Configuration**: namingConventionsConfig (the whole subsection)

This quality check analyzes the identifiers for different entities (also in different contexts) and checks whether they comply to the supplied naming schemes. The naming schemes are defined as regular expressions in the "namingConventionsConfig" subsection in the T3Q configuration file. The basis for the naming conventions as well as the default values are taken from ?http://www.ttcn-3.org/NamingConventions.htm. The default settings in the "namingConventionsConfig" subsection are listed below. The fields are self-explanatory - they are formed following the schema "{languageElement}RegExp", where the language elements are also taken from ?http://www.ttcn-3.org/NamingConventions.htm and ordered in the same way as on the web page. Blank naming rules will be ignored.

```
<namingConventionsConfig>
  <moduleRegExp>[A-Z].*</moduleRegExp>
  <groupRegExp>[a-z].*</groupRegExp>
  <dataTypeRegExp>[A-Z].*</dataTypeRegExp>
  <messageTemplateRegExp>m_[a-z].*</messageTemplateRegExp>
  <messageTemplateWithWildcardsRegExp>mw_[a-z].*</messageTemplateWithWildcardsRegExp>
  <derivedMessageTemplateRegExp>md_[a-z].*</derivedMessageTemplateRegExp>
  <derivedMessageTemplateWithWildcardsRegExp>mdw_[a-z].*</derivedMessageTemplateWithWild
  <stf160sendTemplateRegExp>cs_[a-z].*</stf160sendTemplateRegExp>
  <stf160receiveTemplateRegExp>cr_[a-z].*</stf160receiveTemplateRegExp>
  <signatureTemplateRegExp>s_[a-z].*</signatureTemplateRegExp>
  <portInstanceRegExp>[a-z].*</portInstanceRegExp>
  <componentInstanceRegExp>[a-z].*</componentInstanceRegExp>
  <constantRegExp>c_[a-z].*</constantRegExp>
  <localConstantRegExp>cl_[a-z].*</localConstantRegExp>
  <extConstantRegExp>cx_[a-z].*</extConstantRegExp>
  <functionRegExp>f_[a-z].*</functionRegExp>
  <extFunctionRegExp>fx_[a-z].*</extFunctionRegExp>
  <altstepRegExp>a_[a-z].*</altstepRegExp>
  <testcaseRegExp>TC_.*</testcaseRegExp>
  <variableRegExp>v_[a-z].*</variableRegExp>
  <componentVariableRegExp>vc_[a-z].*</componentVariableRegExp>
  <timerRegExp>t_[a-z].*</timerRegExp>
  <componentTimerRegExp>tc_[a-z].*</componentTimerRegExp>
  <moduleParameterRegExp>[A-Z][A-Z_1-9]*</moduleParameterRegExp>
  <formalParameterRegExp>p_[a-z].*</formalParameterRegExp>
  <enumeratedValueRegExp>e_[a-z].*</enumeratedValueRegExp>
</namingConventionsConfig>
```

Note that component and port type definitions fall under the generic category "data type" definitions and are

therefore required to follow the same naming schema (start with an uppercase letter). This may be subject to changes in the future. Note also that variables in "for" statements are also required to follow the general naming convention for all variables. Additionally, note that only message templates that directly include wildcards or matching expressions are classified as "message templates with wildcards / matching expressions". This will be subject to change in the future, in that a deep search will be performed, resolving all referenced templates and probing them for wildcards / matching expressions.

Note also that there are additional template restriction based naming conventions checks for templates. These reflect the STF160's notions of send and receive templates. Although the names may sound misleading, for these naming conventions the actual usage context (i.e. send or receive statements) for the templates in question is **not** taken into consideration. These rules are based on template restrictions in the definition of the templates, thus are also content-based. The rules according to which templates are classified as send or receive templates are as follows:

1. Templates defined with an **omit** or **value** restriction are considered send templates and shall conform to the naming convention. If such templates have formal template parameters, then these formal template parameters shall also be defined with the same (*omit* or *value*) template restrictions. If the formal template parameters are not defined with the appropriate template restrictions for this type of template (i.e. the template definition is defined inconsistently or ambiguously), an *INFORMATION* message will be provided informing of the occurrence, and the naming convention will not be checked for such occurrences.
2. Templates defined without a restriction or with a *present* restriction are considered receive templates and shall conform to the naming convention. There is no restriction on the formal template parameters for such templates.

With careful selection of the naming conventions rules, it is possible to combine the template naming conventions for wildcards / no wildcards and STF160's send / receive templates if desired. If only STF160's send / receive distinction is desired, then the naming convention rules for wildcards / no wildcards templates shall be left blank.