

`std::false_type`

`std::true_type`

`is_configuration_string
< ConfigurationString
 < N > >`

```
graph BT; A["is_configuration_string<br/>< ConfigurationString<br/>< N > >"] --> B["std::false_type"]; A --> C["std::true_type"];
```

The diagram illustrates a C++ template specialization. At the bottom, a gray box contains the code for a template `is_configuration_string` specialized for `ConfigurationString` with a template argument `N`. Two blue arrows point from this box to two white boxes at the top: `std::false_type` on the left and `std::true_type` on the right. This indicates that the specialization in the gray box is used to determine the value of `is_configuration_string` for the specific type `ConfigurationString` and argument `N`, resulting in either `std::false_type` or `std::true_type`.