SmartIDs

Direct use of recognition criteria

Concept

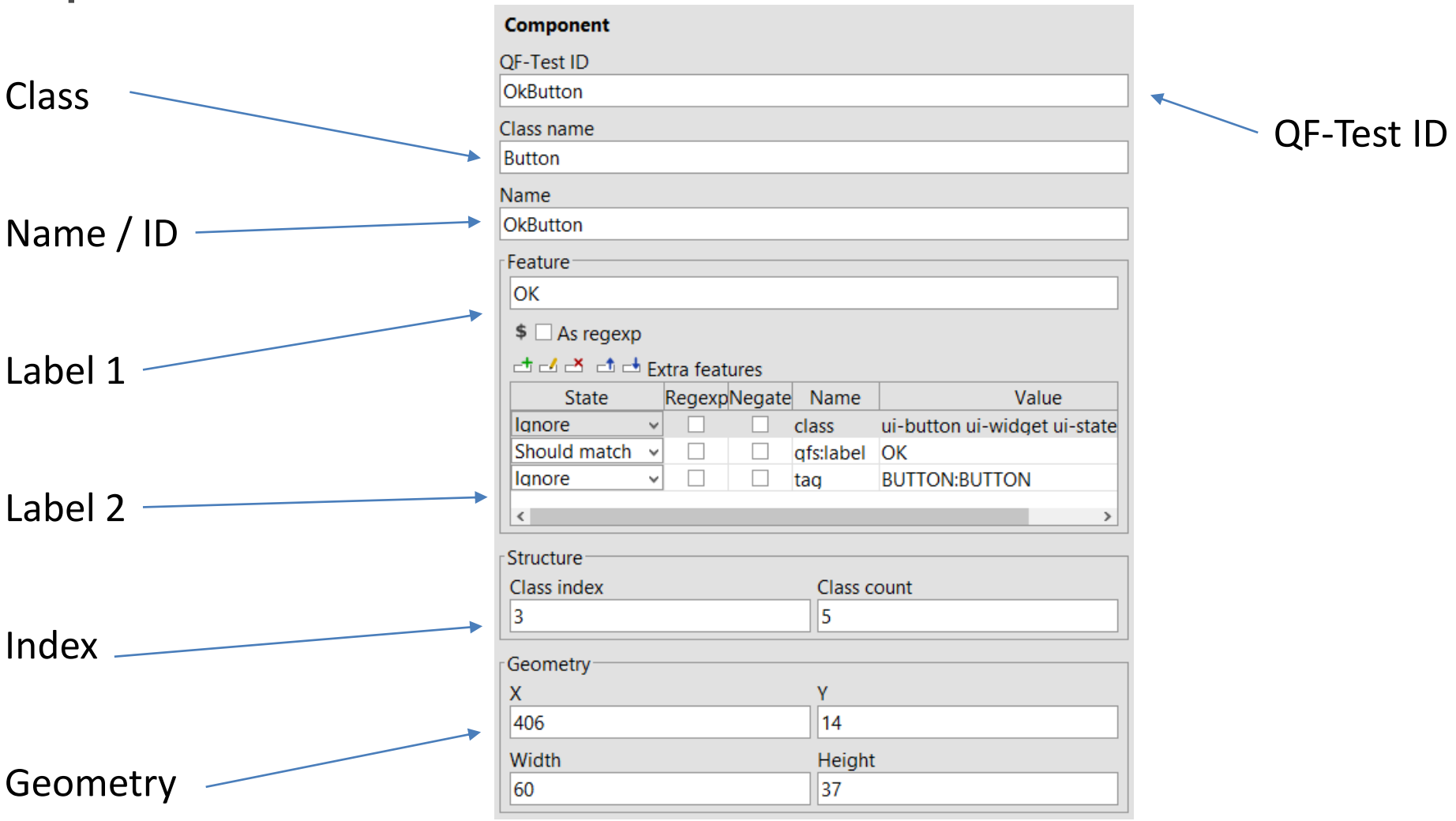
QF-TEST

Flexible, stable component recognition
without nodes saving the
recognition criteria.

The idea is to keep simple things simple.

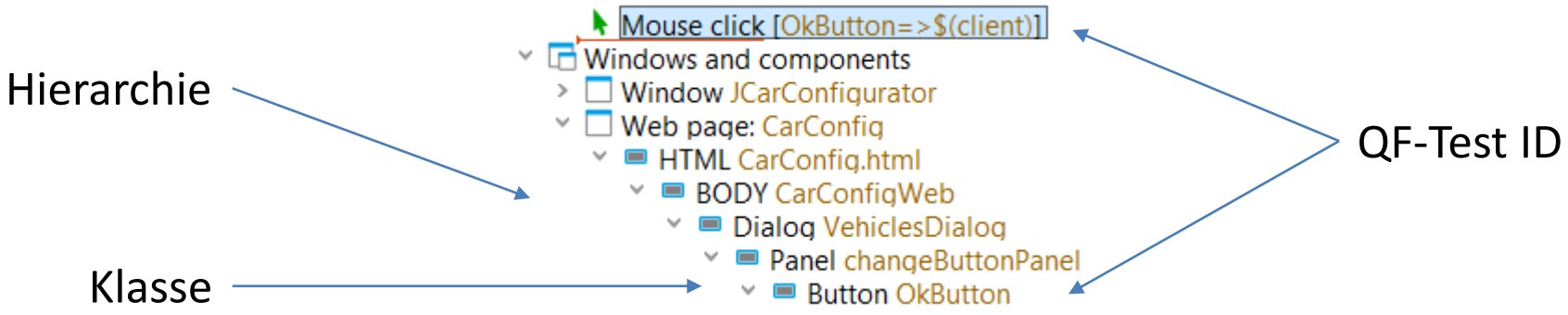
Component node

Component node:



Component node

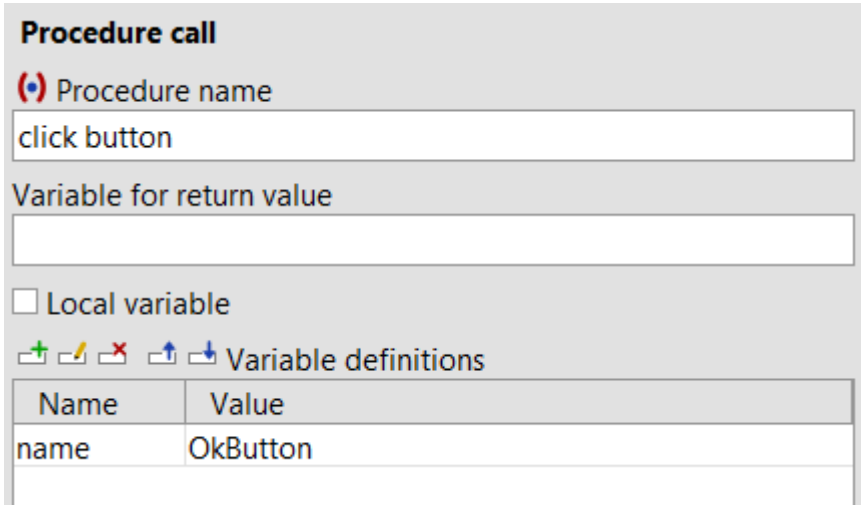
Component tree:

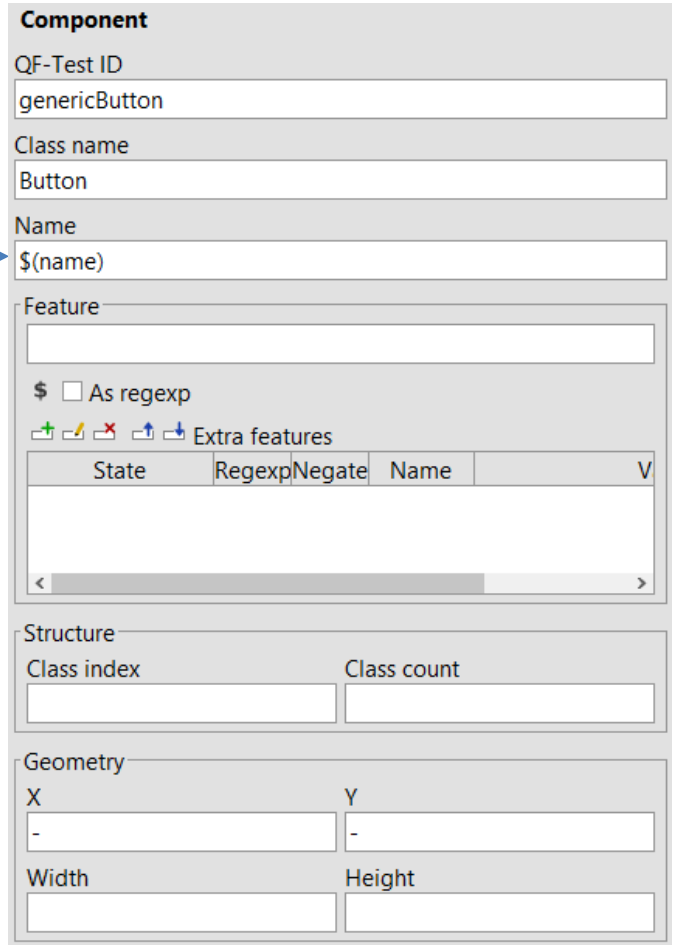


Using generic components:

1) Component with name

Name as variable

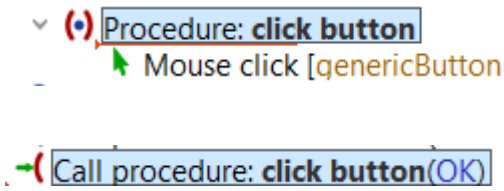




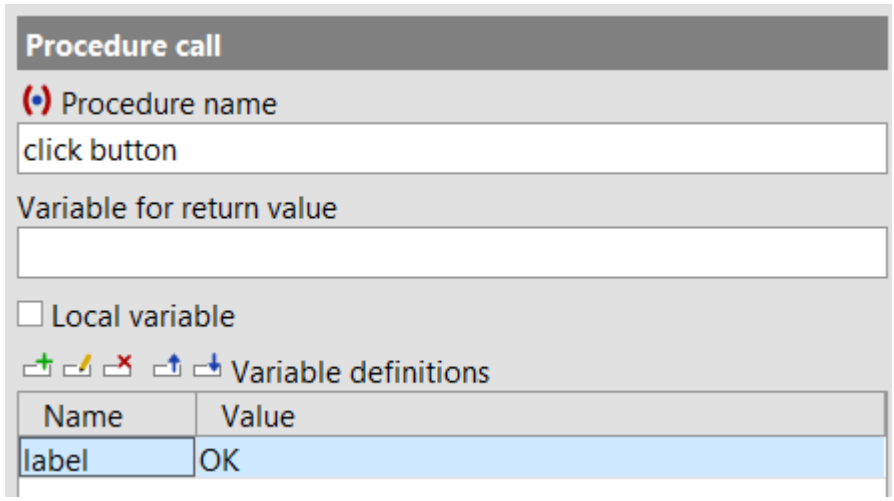
Generic components

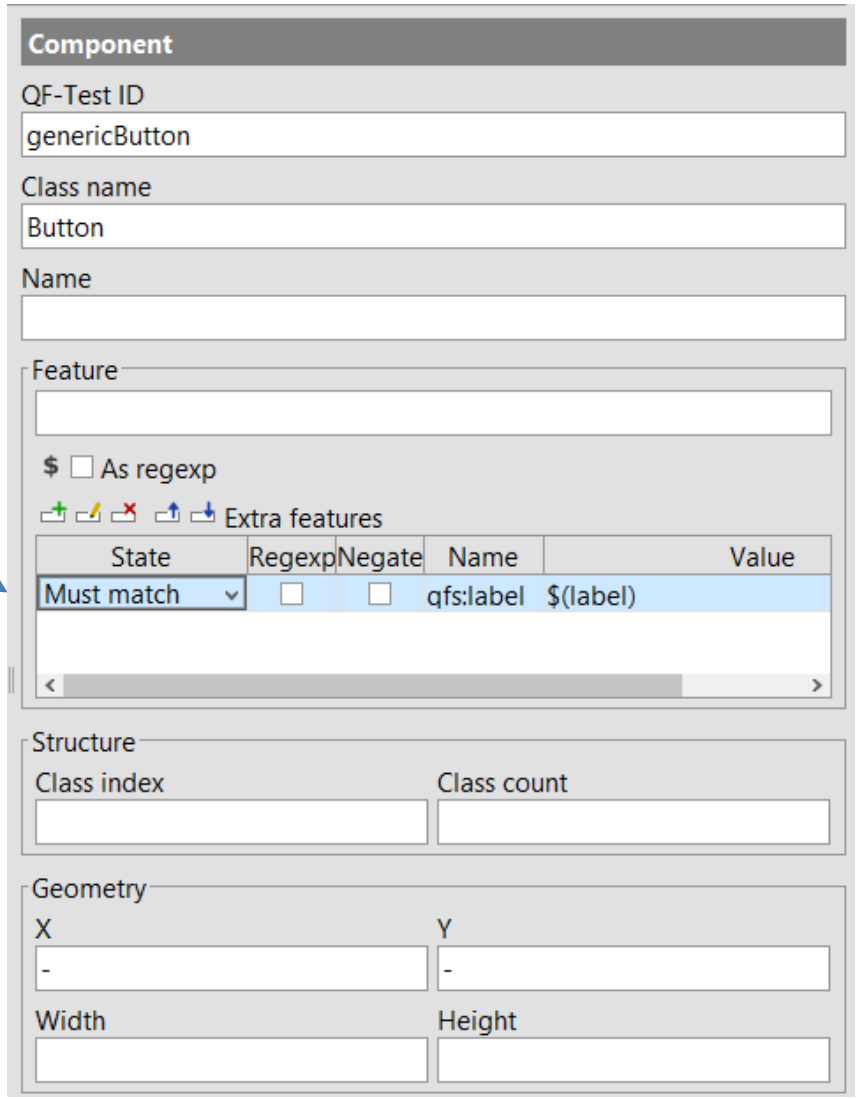
Using generic components:

2) Component with label

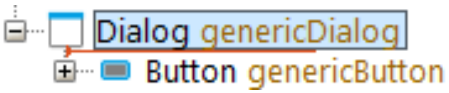


Label as variable

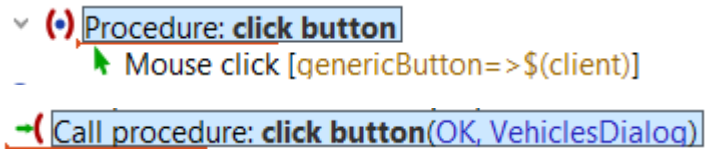


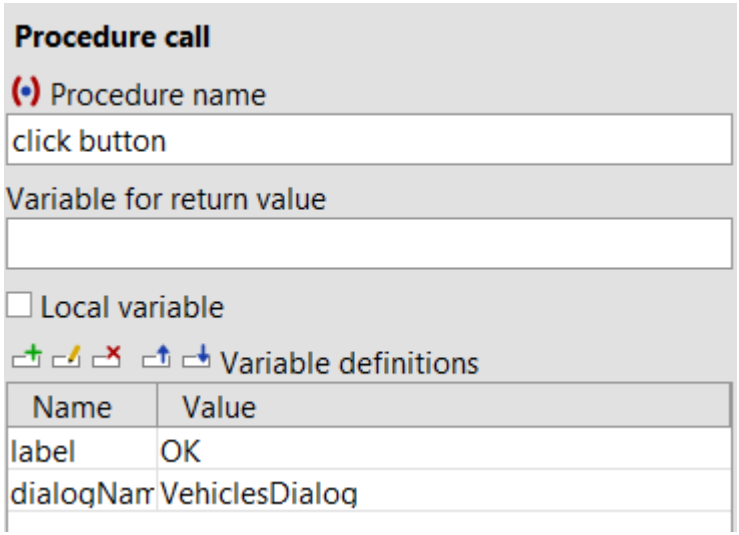


Using generic components:

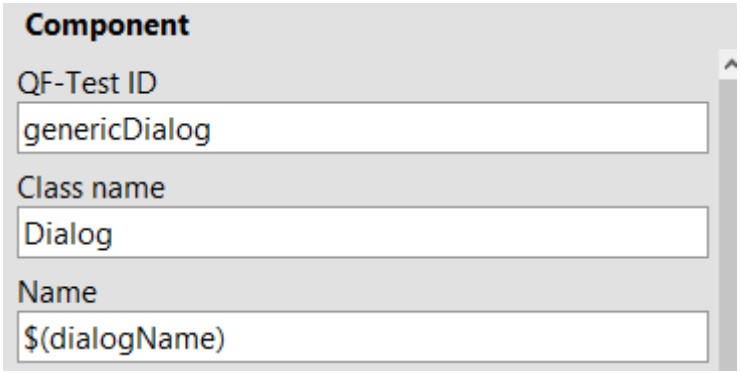


3) Component in a modal dialog

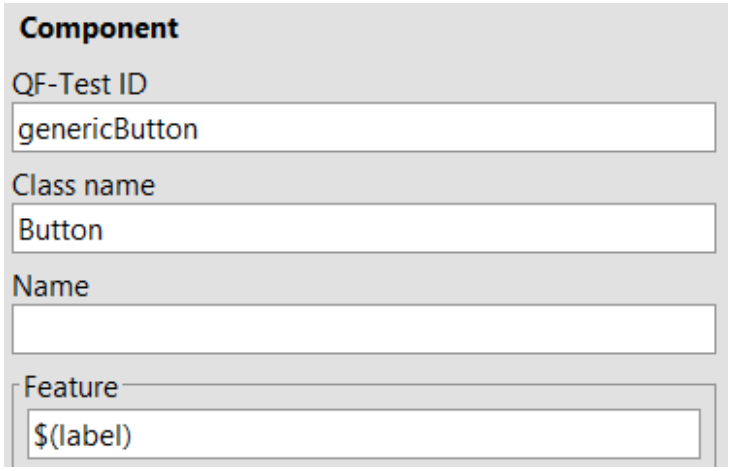




Generic dialog



Generic button



Generic components

QF-TEST

Pros:

- Lean, clear component tree
- Test implementation without component recording
- Good readability of procedure calls

Cons:

- Generic components for
  - Each class
  - Each criteria (name, label)
  - Structure/hierarchy (dialog, panel…)
- Maintenance via Search/Replace
- Performance when using labels
- Reduced number of recognition criteria

SmartIDs

QF-TEST

- A SmartID consists of one or more recognition criteria, chosen from the available criteria.
- The recognition criteria are entered in the space of the QF-Test ID of the component.
- No component nodes required.
- Available criteria for a SmartID:
    - Generic class
    - Name (ID)
    - Feature
    - Extra features (qfs:label)
    - Index
    - component hierarchy

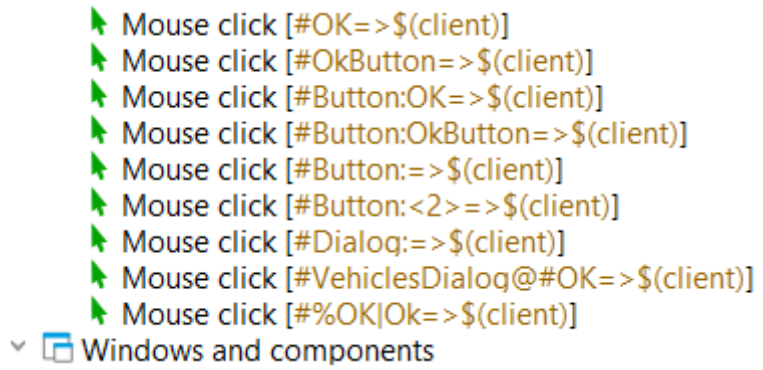
SmartIDs - Syntax

QF-TEST

Form
#Label
#Name
#Klasse:Label
#Klasse:Name
#Klasse:
#Klasse:<2>
#Parent@#Child
#%Regexp
#TabPanel:@Tab1
#Tab:Tab1
Other combinations

General syntax:

#[%][noscope:][ENGINE:][CLASS:][VALUE][<INDEX>]

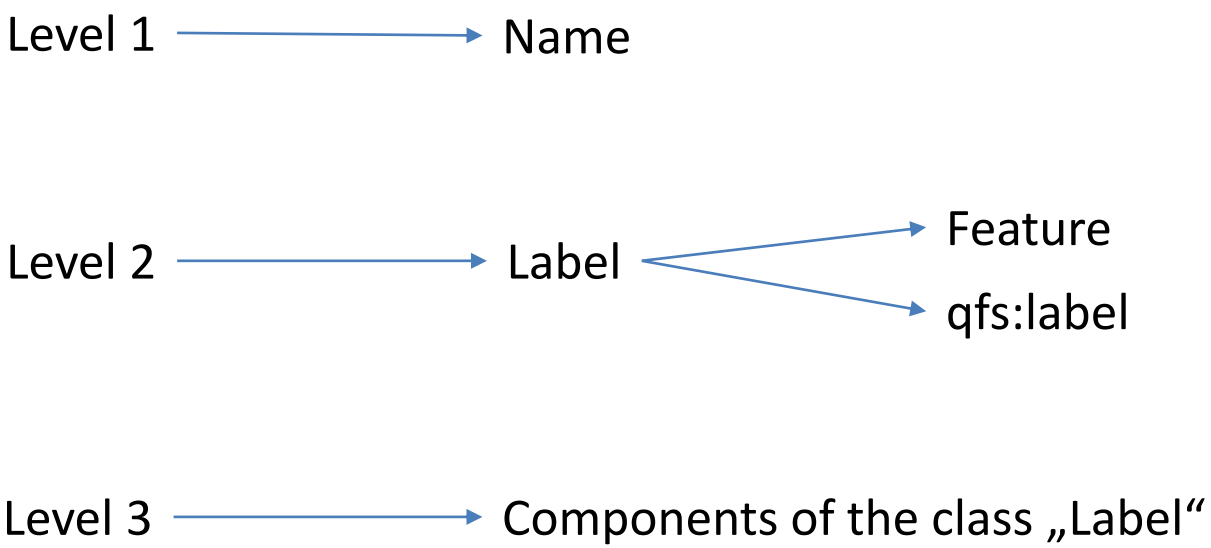


The recognition criteria will not be saved
-> no component tree.

„%“ shows VALUE is a regular expression.

SmartIDs - Criteria

When a SmartID addresses severals components the ranking is as follows*:



-> Same stability and performance as a component node: #Name

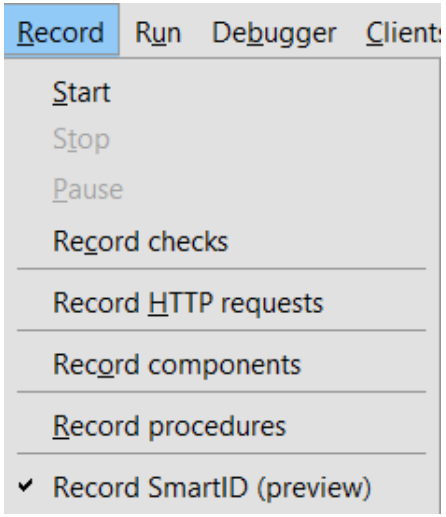
*) Can be changed via option

SmartIDs – Pros and cons

QF-TEST

- Pros
  - No component tree
  - Test implementation without component recording
  - Good readability of nodes
  - Explicit recognition criteria
- Cons
  - Maintenance via Search/Replace
  - Performance when using labels
  - Reduced number of recognition criteria

QF-Test 6.0 supports SmartIDs as a preview feature, however fully implemented



The menu „Record“ has a new entry:
„Record SmartID preview)“

Prefer recording of labels:

```
rc.setOption(Options.OPT_RECORD_SMARTID_PRIORITIES, "label,name")
```

SmartIDs – Usage

QF-TEST

- Ignore the component hierarchy
- Readability of tests
- Test driven development (TDD)
- Keyword based tests (KDT, BDT)
- Integration with other test tools
  for example Robot Framework
- Cross technology tests

Scope

QF-TEST

- Container components can be set as „Scope“
  (Window, Dialog, Panel…)
- The scope limits the range in which QF-Test will search for the subsequently addressed components
- Faster component recognition
- Improved readabilty of the tests

Scope

- Setting the scope:
  - "@scope <SmartID>" as doc tag in the comment of a node
  - "@scope <QF-Test ID of the component>" as a doc tag in the comment of a node
  - Implicitly with dialogs and popups

Scopes can be nested.

Scope

- Leaving a scope:
  - When exiting the node it was set for
  - When the next node has no (@noscope) or a different Scope
  - Implicitly when closing the dialogs or popup

SmartIDs – Tips

QF-TEST

Make sure the SmartID addresses the correct component:

- Add the class

  #TextField:First name
- Use a scope

  @scope #Customer address
- Use an index

  #Table:&0&5@#Button:<1>

In case a recorded SmartID would become too complicated QF-Test will record a component node.

SmartIDs – Customer reaction

QF-TEST

The introduction of **Scopes** and **SmartIDs** was a critical leap for QF-Test. It has opened the gates wide for generic solutions. Scope and SmartIDs are very easy to explain, understand and use. They are also super powerful, easy to maintain, parameterize, centralize, and bring a lot of accuracy and stability.

I switched all the test automation projects I was managing at the time to SmartIDs three months after my introduction to Scopes and SmartIDs. I also implemented a model-based software solution “automated test automation tool” that generates test code for model-based applications and uses QF-Test to automatically populate and validate the UI. This significantly reduced the implementation and maintenance efforts in the projects. After this, QF-Test and me became obviously unstoppable at mgm.

Lilia Gargouri, mgm, in the interview with Thomas Max, QFS