Client-side rules

Language: Deutsch · **English**

Client-side rules

> **Groovy right in the form: calculated values, dynamic field/button states and functions via the context.**

REFERENZ | LOW-CODE | UPDATED: JUL 2026 | APPLIES TO NUCLOS 4.2026.X

On this page

- What are client-side rules?
- Namespaces and packages
- Access via context
- Dynamic properties (active/inactive)
- Functions via library rules
- Best practices & debugging
- Related pages

What are client-side rules?

Client-side rules are **Groovy** expressions that react directly in the form – for calculated values, dynamic properties (e.g. field active/inactive, background colour) and buttons. They are defined on the layout or the business object.

Namespaces and packages

A namespace is defined per **Nuclet** (local identifier). Business objects without a Nuclet use the default namespace `DEF`. Packages enable Spring-managed library rules (callable functions).

Access via `context`

In code you access data via the `context` variable and expressions. `[entity]` is the *internal name* of the business object:

```
#{[namespace].[entity]}          // aktuelles BO / Unterformular-Datensätze als Liste
#{[namespace].[entity].[field]}  // Wert eines Attributs
#{[namespace].[entity].[field].id}      // Id eines Referenzfelds (java.lang.Long)
#{[namespace].[entity].[field].context} // Context eines referenzierten Objekts
```

The script runs when a new record is created or a field it reads changes. Example – sum a total across a subform:

```
def brutto = new java.math.BigDecimal(0.000)
context."#{WAR.auftrag_position}".each { item ->
    brutto = brutto.add(java.math.BigDecimal.valueOf(
        item."#{WAR.auftrag_position.gesamtpreisrechnung}"))
}
return brutto.setScale(4, java.math.RoundingMode.HALF_UP).doubleValue()
```

Dynamic properties (active/inactive)

A boolean return value controls „edit/new/delete/clone active“ and buttons: `return true` = active, `return false` = inactive.

> **Mind the context**
>
> The base context for *new active* is the parent BO of the subform, but for *edit/delete/clone active* it is the subform BO itself. To read the order status from a position, first go one context up (`.context`).

Functions via library rules

A library rule annotated with `@Component` (Spring) and a method annotated with `@Function("name")` can be called via `context."#FUNCTION {name}"(...)`. The package namespace and package name must match; parameter/return types must fit.

Best practices & debugging

Read a field from the parent object (the field must exist in the layout, may be disabled):

```
stateNumeral = context."#{NUC.Rechnungsposition.rechnung.context}"."#{NUC.Rechnung.nuclosStateNumber}"
```

- Log output with `log.info("...")` (Window Output/Scripting).
- Current user via the `username` variable (Nuclos 4.0.15+).
- Groovy exceptions are not passed to the user – catch them and show a `JOptionPane` message box for debugging.

Related pages

🧩 Business object

Calculation expression.

Open

🖼️ Layout

Dynamic field states.

Open

📖 Server-side rules

Server logic.

Open