Selectively disabling rule execution

Language: Deutsch · **English**

Selectively disabling rule execution

> **Avoid endless loops: disable individual rules per context via ThreadLocal switches.**

HOW-TO | DEVELOPER | UPDATED: JUL 2026 | APPLIES TO NUCLOS 4.2026.X

On this page

Why disable rules selectively?

Sometimes you want to prevent certain rules (or parts of them) from running – typically when updating individual fields from within a rule, to avoid **endless loops** or performance problems. This is done with cross-rule **ThreadLocal** variables.

> **Caution**
>
> Use with care – do not undermine the business logic of the rules. For simple cases prefer SaveFlags.

1. A switch in the rule to be disabled

```
public class MyUpdateRule implements UpdateRule {
    // ThreadLocal-Schalter, initial aktiv
    public static final ThreadLocal<Boolean> ACTIVE =
        ThreadLocal.withInitial(() -> Boolean.TRUE);

    public void update(UpdateContext context) throws BusinessException {
        if (ACTIVE.get()) {
            // ... eigentliche Logik ...
        }
    }
}
```

ThreadLocal variables only affect the current thread/user context – the rule is not disabled globally (unlike the rule's active flag).

2. Disabling from the updating rule

```
MyBusinessObject mbo = QueryProvider.getById(MyBusinessObject.class, id);
mbo.setMyField(value);
try {
    MyUpdateRule.ACTIVE.set(Boolean.FALSE);   // Regel deaktivieren
    BusinessObjectProvider.update(mbo);
} finally {
    MyUpdateRule.ACTIVE.set(Boolean.TRUE);    // immer wieder aktivieren!
}
```

> **Don't forget finally**
>
> Always re-activate in the `finally` block so the switch is reliably reset even on error.

Related pages

📖 Server-side rules

Rule context.

Open

⚙ SaveFlags

Alternative on save.

Open