Class generation and rule compilation

Language: Deutsch · **English**

Class generation and rule compilation

> **How Nuclos generates BOs, state models and more as Java classes – types, JAR order and timing.**

REFERENZ | APPLICATION DEVELOPER | UPDATED: JUL 2026 | APPLIES TO NUCLOS 4.2026.X

On this page

Why generated classes?

For rule programming Nuclos provides business objects, state models and other units **objectified as Java classes**. This keeps access from rules simple and type-safe. After a successful save, changes are immediately available in the classloader.

> **Naming convention**
>
> The **package name** matches the nuclet's (otherwise a default, e.g. `org.nuclet.businessentity`). Special characters, spaces or leading digits are replaced to be Java-compliant – class/method names may therefore differ from the metadata.

Generated class types

Type	Suffix	Purpose
Business object	–	objectified BO with getters/setters and query constants; QueryProvider.
State model	SM	states as constants `State_<n>` for the StatemodelProvider.
Work step	GEN	type-safe generation class for the GenerationProvider.
Report datasource	DS	objectified data source for the DatasourceProvider.
Report	RE	report with output formats for the report/printout (PDF only).
Printout (form)	PO	form with output formats for the PrintoutProvider (PDF only).
Import structure definition	ISD	structure definition for the ImportProvider.
System/nuclet parameter	–	parameter constants for the ParameterProvider.

Example of a business object incl. query constants:

```
public class Adresse extends AbstractBusinessObject implements Modifiable {

        // Konstanten für den QueryProvider
        public static final Attribute<Boolean> Standard =
                new Attribute<Boolean>("Standard", "org.nuclet.basis", "Adresse", new Long(40005905),
"standard", new Long(40006275), Boolean.class );

        public static final StringAttribute<String> Postfach =
                  new StringAttribute<String>("Postfach", "org.nuclet.basis", "Adresse", new Long(40005905),
"postfach", new Long(40006271), String.class );

        public java.lang.Boolean getStandard() {
                return getField("standard", java.lang.Boolean.class);
        }

        public void setStandard(java.lang.Boolean pStandard) {
                setField("standard", pStandard);
        }

        public java.lang.String getPostfach() {
                return getField("postfach", java.lang.String.class);
        }

        public void setPostfach(java.lang.String pPostfach) {
                setField("postfach", pPostfach);
        }
}
```

Example of a state model class:

```
public class AuftragSM{
        public static State State_10 =
                    new StateImpl(IdUtils.toLongId(40006496), "Entwurf", "Geplant", 10, IdUtils.toLongId
(40006477));
        public static State State_90 =
                     new StateImpl(IdUtils.toLongId(40006497), "Abgeschlossen", "Abgeschlossen", 90, IdUtils.
toLongId(40006477));
        public static State State_50 =
            new StateImpl(IdUtils.toLongId(40006498), "Offen", "Offen", 50, IdUtils.toLongId(40006477));
        public static State State_99 =
            new StateImpl(IdUtils.toLongId(40006499), "Storniert", "Storniert", 99, IdUtils.toLongId(40006477));
}
```

Compilation & order

The classes are compiled grouped by type into separate JARs in the CodeGenerator directory and added to the classloader:

Type	JAR	Order
Business objects	BOEntities.jar	first – only access BO metadata.
State model classes	statemodels.jar	independent.
Report datasource classes	ReportDSEntities.jar	independent.
Report classes	ReportEntities.jar	independent.
Printout classes	PrintoutEntities.jar	independent.
Import structure definition	ImportStructDefsEntities.jar	independent.
Parameters	Parameter.jar	independent.
Work step classes	Generation.jar	after the business objects (source/target objects).
Rules & libraries	Nuclet.jar	last – uses all other classes; fails on errors or invalid references.

When generation happens

- **System start:** all classes are rebuilt (directory is cleared); on errors the server still starts.
- **Nuclet import:** all classes are deleted and regenerated.
- **Runtime:** on changes to BO/state model/report/work step, affected classes and the `Nuclet.jar` are rebuilt.

> **Keep references consistent**
>
> If, say, a status numeral changes (40 50), a rule still addressing status 40 no longer finds it – the `Nuclet.jar` then cannot be built. All references must be **updated manually**.

Related pages

Rule sets

Rules.

Open

Supporting classes

Providers.

Open

Nuclet

Building block.

Open