4.8.3 SEPA: Prozesslogik

Sprache: **Deutsch** · English

4.8.3 SEPA: Prozesslogik

> **Das SEPA-Nuclet stellt in der abstrakten Klasse AbstractSEPALogic die grundlegende Prozesslogik für die Erstellung von SEPA-Exporten bereit.**

REFERENZ | INTEGRATOREN | STAND: JUL 2026 | GILT FUER NUCLOS 4.2026.X

Auf dieser Seite

Das SEPA-Nuclet stellt in der abstrakten Klasse AbstractSEPALogic die grundlegende Prozesslogik für die Erstellung von SEPA-Exporten bereit.

Jedoch ist diese Prozesslogik an drei Stellen anwendungsspezifisch für Ihre Geschäftspraxis anzupassen.

Die betrifft

- die Generierung einer anwendungsspezifischen SEPA-Kreditoren-ID (auf Basis der tatsächlichen Daten Ihrer Firma)
- die Gestaltung des zu übermittelnden Verwendungszweck für SEPA-Lastschriften
- die Ermittellung aller in einem SEPA-Export zu berücksichtigenden Zahlungsreferenzen

Bei der Anbindung des SEPA-Nuclets sind hierfür die Klasse SEPALogic und SEPALogicFactory anzupassen.

> Es wird empfohlen, die Klasse SEPALogic nur als Vorlage für eine eigene Implementierung zu verwenden. Näheres dazu finden Sie im nächsten Abschnitt 4.8.3.1.

Außerdem ist die Initialisierung der tatsächlich eingesetzten Prozesslogik in zwei Regeln einzusetzen.

Klasse	Funktion	Java-Package	Anzupassende Methoden
SEPALogic	Template für anwendungspezifische Prozesslogik	org.nuclet.sepa.logic	• createCreditorIdentification(SEPAExport) • createDebitorIdentification(SEPAExport) • createRemittanceInformation(AbstractCreditorWrapper, AbstractReferenceWrapper, BigDecimal) • createRemittanceInformation(AbstractDebitorWrapper, AbstractReferenceWrapper, BigDecimal) • fetchCreditTansferReferences(SEPAExport) • fetchDirectDebitReferences(SEPAExport)
ExportSEPAMessage	erstellt eine Vorschau der zu exportierenden SEPA-Zahlungen	org.nuclet.sepa.rule	custom(CustomContext)
ExportSEPAMessageAndProcessReferences	erzeugt einen SEPA-Export und markiert die exportierten Objekte	org.nuclet.sepa.rule	changeState(StateChangeContext)

Tabelle 4.8.3: Übersicht, Anpassungen in Prozesslogik

4.8.3.1 SEPALogic

Bitte verwenden Sie die Klasse SEPALogic nur als Template für Ihre eigene Impementierung.

Dazu führen Sie bitte die folgenden Schritte durch:

1. Klonen Sie Klasse SEPALogic.
2. Beim Klonen geben Sie in der ersten Code-Zeile bitte Ihren eigenen Package-Namen an (z.B. *de.ihrefirma.sepa.logic*) an.
3. Optional können Sie der geklonten Klasse einen eigenen Klassennamen vergeben. Bitte berücksichtigen Sie dabei auch die Konstruktoren.

Die Klasse SEPALogic dient als Vorlage zur Ergänzung der grundlegenden Abläufe in der Prozesslogik.

Diese Logik ist an drei Stellen für den Anwendungsfall "Lastschriften" und an drei Stellen für den Anwendungsfall "Überweisungen" von Ihnen anwendungsspezifisch zu ergänzen:

Lastschriften

- Die Methode createCreditorIdentification(SEPAExport) generiert die SEPA-Kreditoren-ID. Diese ID dient der Identifizierung Ihrer Firma im SEPA-Zahlungverkehr.
- Mit der Methode createRemittanceInformation(AbstractDebitorWrapper,...) lässt sich der Verwendungszweck für SEPA-Lastschriften individuell gestalten.
- Die Methode fetchDirectDebitReferences() sollte alle Zahlungsreferenzen ermitteln, die in einem SEPA-Export zu berücksichtigen sind.

Überweisungen

- Die Methode createDebitorIdentification(SEPAExport) generiert die SEPA-Debitoren-ID. Diese ID dient der Identifizierung Ihrer Firma im SEPA-Zahlungverkehr.
- Mit der Methode createRemittanceInformation(AbstractCreditorWrapper,...) lässt sich der Verwendungszweck für SEPA-Überweisungen individuell gestalten.
- Die Methode fetchCreditTransferReferences() sollte alle Zahlungsreferenzen ermitteln, die in einem SEPA-Export zu berücksichtigen sind.

Ein Beispiel dazu ist jeweils in einem Kommentarblock angegeben.

org.nuclet.sepa.logic.SEPALogic

```
package org.nuclet.sepa.logic;


import java.math.BigDecimal;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Set;

import javax.xml.bind.JAXBElement;

import org.nuclos.api.businessobject.BusinessObject;
import org.nuclos.api.businessobject.Query;
import org.nuclos.api.context.JobContext;
import org.nuclos.api.context.RuleContext;
import org.nuclos.api.exception.BusinessException;
import org.nuclos.api.provider.BusinessObjectProvider;
import org.nuclos.api.provider.QueryProvider;

import org.nuclet.common.logging.LogLevel;
import org.nuclet.common.logic.AbstractBusinessLogic;

import org.nuclet.sepa.SEPAExport;
import org.nuclet.sepa.SEPAPaymentType;
import org.nuclet.sepa.jaxb.sdd.Document;
import org.nuclet.sepa.logic.*;
import org.nuclet.sepa.logic.PaymentType;
import org.nuclet.sepa.logic.SequenceType;
import org.nuclet.sepa.wrapper.*;

// @replace!
//
// import org.nuclet.businesstemplate.Client;
```

```
// import org.nuclet.businesstemplate.ClientBilling;
// import org.nuclet.businesstemplate.ClientBillingPosition;
// import org.nuclet.businesstemplate.Supplier;
// import org.nuclet.businesstemplate.SupplierBilling;
// import org.nuclet.businesstemplate.SupplierBillingPosition;


/**
 * Concrete class implementing the declarations found in abstract class AbstractSEPALogic
 *
 * @note This class is to be read as a dummy implementation. Its methods should be overwritten
 * and filled/adapted with application specific code.
 *
 * @usage org.nuclet.sepa.job.SEPAExportJob
 *
 * @version 2.0
 * @date 03.02.2015
 * @nuclet org.nuclet.SEPA
 * @nucletversion 2.0.0
 * @sincenucletversion 1.0.0
 * @since 21.03.2014
 *
 * @author frank.lehmann@nuclos.de
 *
 */
public class SEPALogic extends AbstractSEPALogic
{
 private static final LogLevel LOGLEVEL = LogLevel.DEBUG;

 public SEPALogic(final JobContext context)
 {
 super(context, LOGLEVEL);
 }

 public SEPALogic(final JobContext context, final LogLevel logLevel)
 {
 super(context, logLevel);
 }

 public SEPALogic(final RuleContext context, final LogLevel logLevel)
 {
 super(context, logLevel);
 }

 public SEPALogic(final RuleContext context)
 {
 super(context, LOGLEVEL);
 }


 /**
 * Builds a creditor identification, based on a given SEPA export (direct debit initiation)
 *
 * @note To be implemented in a dedicated subclass!
 * @see org.nuclet.sepa.logic.CreditorIdentification
 *
 * @param boSEPAExport A given oject of type <code>SEPAExport</code>
 *
 * @return a creditor identification, based on a given SEPA export
 *
 * @throws BusinessException
 * @throws BusinessException SPA.483.001
 */
 protected CreditorIdentification createCreditorIdentification(final SEPAExport boSEPAExport) throws
BusinessException
 {
 // @replace!
 //
 // Im folgenden Beispiel wird der Kreditorenstammdatensatz in der Tabelle
 // der Kunden ("Client") gespeichert und als "intern" ("is internal") gekennzeichnet.
 // Die Kreditorenkennung wird also aus genau jenem Stammdatensatz erzeugt.
```

```
 //
 // final Query qryCreditor = QueryProvider.create(Client.class)
 // .where(Client.IsInternal.eq(Boolean.TRUE));
 // final List<Client> lstClients = QueryProvider.execute(qryCreditor);
 //
 // logDebug("Creditor list size: " + lstClients.size());
 //
 // if (lstClients != null && lstClients.size() == 1) {
 // final Client boCreditor = lstClients.iterator().next();
 //
 // final CreditorIdentification creditorId = new CreditorIdentification(
 // boCreditor.getName(),
 // boCreditor.getBic(),
 // boCreditor.getIban(),
 // boCreditor.getSEPACreditorId());
 //
 // return creditorId;
 // } else if (lstClients != null && lstClients.size() > 1) {
 // throw new BusinessException("More than one clients marked as \"internal\" exist!");
 // } else {
 // throw new BusinessException("Client marked as \"internal\" is missing!");
 // }

 return null;
 }

 /**
 * Builds a creditor identification, based on a given SEPA export (credit transfer initiation)
 *
 * @note To be implemented in a dedicated subclass!
 * @see org.nuclet.sepa.logic.CreditorIdentification
 *
 * @param boSEPAExport A given oject of type <code>SEPAExport</code>
 *
 * @return a debitor identification, based on a given SEPA export
 *
 * @throws BusinessException
 * @throws BusinessException SPA.483.001
 */
 protected DebitorIdentification createDebitorIdentification(final SEPAExport boSEPAExport) throws
BusinessException
 {
 // @replace!
 //
 // Im folgenden Beispiel wird der Debitorenstammdatensatz in der Tabelle
 // der Kunden ("Client") gespeichert und als "intern" ("is internal") gekennzeichnet.
 // Die Debitorenkennung wird also aus genau jenem Stammdatensatz erzeugt.
 //
 // final Query qryDebitor = QueryProvider.create(Client.class)
 // .where(Client.IsInternal.eq(Boolean.TRUE));
 // final List<Client> lstClients = QueryProvider.execute(qryDebitor);
 //
 // logDebug("Debitor list size: " + lstClients.size());
 //
 // if (lstClients != null && lstClients.size() == 1) {
 // final Client boDebitor = lstClients.iterator().next();
 //
 // final DebitorIdentification debitorId = new DebitorIdentification(
 // boDebitor.getName(),
 // boDebitor.getBic(),
 // boDebitor.getIban(),
 // boDebitor.getSEPACreditorId());
 //
 // return debitorId;
 // } else if (lstClients != null && lstClients.size() > 1) {
 // throw new BusinessException("More than one clients marked as \"internal\" exist!");
 // } else {
 // throw new BusinessException("Client marked as \"internal\" is missing!");
 // }

 return null;
```

```
}

/**
* Builds the remittance information, related to the given creditor and reference
*
* @param creditor An object of type <code>AbstractCreditorWrapper</code>, representing the debitor
* @param reference An object of type <code>AbstractCreditorReferenceWrapper</code>, representing the reference
* @param bgdAmount The payment amount, related to the given debitor and reference
*
* @return the remittance information, related to the given debitor and reference
*
* @throws BusinessException
*/
public String createRemittanceInformation(final AbstractCreditorWrapper debitor,
final AbstractCreditorReferenceWrapper reference,
final BigDecimal bgdAmount) throws BusinessException
{
// @replace!
//
// final SupplierBilling boSupplierBilling = (SupplierBilling)reference.getBusinessObject();
// final StringBuffer sbRemittanceInfo = new StringBuffer();
//
// sbRemittanceInfo.append("ACCT. ");
// sbRemittanceInfo.append(creditor.getSEPAMandateIdentification());
// sbRemittanceInfo.append(" DATE ");
// sbRemittanceInfo.append(reference.getReferenceDate());
// sbRemittanceInfo.append(" AMNT. ");
// sbRemittanceInfo.append(bgdAmount);
// sbRemittanceInfo.append(" BGNR ");
// sbRemittanceInfo.append(reference.getCreditTransferReference());
//
// return sbRemittanceInfo.toString();

return null;
}

/**
* Builds the remittance information, related to the given debitor and reference
*
* @param debitor An object of type <code>AbstractDebitorWrapper</code>, representing the debitor
* @param reference An object of type <code>AbstractReferenceWrapper</code>, representing the reference
* @param bgdAmount The payment amount, related to the given debitor and reference
*
* @return the remittance information, related to the given debitor and reference
*
* @throws BusinessException
*/
public String createRemittanceInformation(final AbstractDebitorWrapper debitor,
final AbstractDebitorReferenceWrapper reference,
final BigDecimal bgdAmount) throws BusinessException
{
// @replace!
//
// final ClientBilling boClientBilling = (ClientBilling)reference.getBusinessObject();
// final StringBuffer sbRemittanceInfo = new StringBuffer();
//
// sbRemittanceInfo.append("ACCT. ");
// sbRemittanceInfo.append(debitor.getSEPAMandateIdentification());
// sbRemittanceInfo.append(" DATE ");
// sbRemittanceInfo.append(reference.getReferenceDate());
// sbRemittanceInfo.append(" AMNT. ");
// sbRemittanceInfo.append(bgdAmount);
// sbRemittanceInfo.append(" BGNR ");
// sbRemittanceInfo.append(reference.getDirectDebitReference());
//
// return sbRemittanceInfo.toString();

return null;
}

/**
```

```
 * Fetches all credit transfer references that are to be included in a given SEPA export (credit transfer
initiation)
 *
 * @param boSEPAExport A given SEPA export
 *
 * @throws BusinessException
 */
 protected List<AbstractCreditTransferReferenceWrapper> fetchCreditTransferReferences(final SEPAExport
boSEPAExport)
 throws
 BusinessException
 {
 // @replace!
 //
 // final List<AbstractCreditTransferReferenceWrapper> lstCreditTransferReferences = new
ArrayList<AbstractCreditTransferReferenceWrapper>();
 //
 // final Query qrySupplierBillingPositions = QueryProvider.create(SupplierBillingPosition.class)
 // .where(SupplierBillingPosition.DueDate.Gte(boSEPAExport.getDateFrom()))
 // .and(SupplierBillingPosition.Faelligkeitsdatum.Lte(boSEPAExport.getDateUntil()))
 // .and(SupplierBillingPosition.SEPAExportDate.isNull())
 // .and(SupplierBillingPosition.IsOpen.eq(Boolean.TRUE))
 // .orderBy(SupplierBillingPosition.SupplierBilling, Boolean.TRUE)
 // .orderBy(SupplierBillingPosition.DueDate, Boolean.TRUE);
 // final List<SupplierBillingPosition> lstSupplierBillingPositions = QueryProvider.execute
(qrySupplierBillingPositions);
 //
 // for (final SupplierBillingPosition boSupplierBillingPosition : lstSupplierBillingPositions) {
 // lstCreditTransferReferences.add(creditTransferReferenceFacade.getWrapper(boSupplierBillingPosition));
 // }
 //
 // return lstCreditTransferReferences;

 return null;
 }

 /**
 * Fetches all direct debit references that are to be included in a given SEPA export (direct debit initiation)
 *
 * @param boSEPAExport A given SEPA export
 *
 * @throws BusinessException
 */
 protected List<AbstractDirectDebitReferenceWrapper> fetchDirectDebitReferences(final SEPAExport boSEPAExport)
 throws
 BusinessException
 {
 // @replace!
 //
 // final List<AbstractDirectDebitReferenceWrapper> lstDirectDebitReferences = new
ArrayList<AbstractDirectDebitReferenceWrapper>();
 //
 // final Query qryClientBillingPositions = QueryProvider.create(ClientBillingPosition.class)
 // .where(ClientBillingPosition.DueDate.Gte(boSEPAExport.getDateFrom()))
 // .and(ClientBillingPosition.Faelligkeitsdatum.Lte(boSEPAExport.getDateUntil()))
 // .and(ClientBillingPosition.SEPAExportDate.isNull())
 // .and(ClientBillingPosition.IsOpen.eq(Boolean.TRUE))
 // .orderBy(ClientBillingPosition.ClientBilling, Boolean.TRUE)
 // .orderBy(ClientBillingPosition.DueDate, Boolean.TRUE);
 // final List<ClientBillingPosition> lstClientBillingPositions = QueryProvider.execute
(qryClientBillingPositions);
 //
 // for (final ClientBillingPosition boClientBillingPosition : lstClientBillingPositions) {
 // lstDirectDebitReferences.add(directDebitReferenceFacade.getWrapper(boClientBillingPosition));
 // }
 //
 // return lstDirectDebitReferences;

 return null;
 }
```

```
}
```

4.8.3.2 ExportSEPAMessage

In der Klasse ExportSEPAMessage ist die Klasse SEPALogic gegen die von Ihnen im Abschnitt 4.8.3.1 erstellte Prozesslogik auszutauschen.

org.nuclet.sepa.rule.ExportSEPAMessage

```
package org.nuclet.sepa.rule;

import org.nuclos.api.annotation.Rule;
import org.nuclos.api.context.CustomContext;
import org.nuclos.api.exception.BusinessException;
import org.nuclos.api.rule.CustomRule;

import org.nuclet.sepa.SEPAExport;
import org.nuclet.sepa.logic.SEPALogic;


/**
 * @name ExportSEPAMesssage
 * @description Export SEPA message without processing the references
 * @usage
 * @change
 *
 * @version 2.0
 * @date 30.01.2015
 * @nuclet org.nuclet.SEPA
 * @nucletversion 2.0.0
 * @sincenucletversion 1.0.0
 * @since 14.03.2013
 *
 * @author frank.lehmann@nuclos.de
 */
@Rule(name="ExportSEPAMesssage", description="Export SEPA Messsage")
public class ExportSEPAMesssage implements CustomRule
{

 public void custom(CustomContext context) throws BusinessException
 {
 final SEPAExport boSEPAExport = context.getBusinessObject(SEPAExport.class);

 SEPALogic logic = new SEPALogic(context);
 logic.initialize(context, boSEPAExport);
 logic.export();
 }
}
```

4.8.3.3 ExportSEPAMessageAndProcessReferences

In der Klasse ExportSEPAMessageAndProcessReferences ist die Klasse SEPALogic gegen die von Ihnen im Abschnitt 4.8.3.1 erstellte Prozesslogik auszutauschen.

org.nuclet.sepa.rule.ExportSEPAMessageAndProcessReferences

```
package org.nuclet.sepa.rule;

import org.nuclos.api.annotation.Rule;
import org.nuclos.api.context.StateChangeContext;
import org.nuclos.api.exception.BusinessException;
import org.nuclos.api.rule.StateChangeRule;

import org.nuclet.sepa.SEPAExport;
import org.nuclet.sepa.logic.SEPALogic;


/**
 * @name ExportSEPAMesssageAndProcessReferences
 * @description Export SEPA messsage and process references
 * @usage
 * @change
 *
 * @version 2.0
 * @date 30.01.2015
 * @nuclet org.nuclet.SEPA
 * @nucletversion 2.0.0
 * @sincenucletversion 1.0.0
 * @since 21.03.2014
 *
 * @author frank.lehmann@nuclos.de
 */
@Rule(name="ExportSEPAMesssageAndProcessReferences", description="Export SEPA messsage and process references")
public class ExportSEPAMesssageAndProcessReferences implements StateChangeRule
{

 public void changeState(StateChangeContext context) throws BusinessException
 {
 final SEPAExport boSEPAExport = context.getBusinessObject(SEPAExport.class);

 SEPALogic logic = new SEPALogic(context);
 logic.initialize(context, boSEPAExport);
 logic.export();
 logic.processReferences(boSEPAExport);
 }
}
```

Verwandte Seiten

📖 Nuclet: SEPA (sind im Handelsnuclet verfügbar)

SEPA-Datenblatt

Öffnen

📖 Schnittstellen

Alle Integrationen

Öffnen