4.8.3 CAMT: Eindeutigkeitsprüfung

Sprache: **Deutsch** · English

4.8.3 CAMT: Eindeutigkeitsprüfung

> **Die Prüfung, ob ein eingehender Datensatz aus dem CAMT.054-Import bereits in der Datenbank vorhanden ist, wird in der Methode doesBankTransactionAlreadyExist() der abstrakten Klasse org.nuclet.camt.logic.AbstractCAMT054Logic realisiert.**

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

Auf dieser Seite

Die Prüfung, ob ein eingehender Datensatz aus dem CAMT.054-Import bereits in der Datenbank vorhanden ist, wird in der Methode **doesBankTransactionAlreadyExist()** der abstrakten Klasse org.nuclet.camt.logic.AbstractCAMT054Logic realisiert. Standardmäßig werden bei diesem Abgleich die folgenden Attribute berücksichtigt:

- Message-ID (message id)
- Empfänger (message recipient)
- Referenznummer (notification id)
- Kontoauszugsnummer (electronic sequence number)
- Bankgeschäftsvorfall (business transaction type code/bank transaction type)
- Buchungsstatus (entry status)
- Auftraggeber (initiator)
- Kreditinstitut (financial institution)
- IBAN (iban)
- BIC (bic)
- Betrag (amount)
- Währung (currency code)
- Soll-Haben-Kennzeichen (credit debit indicator)
- Buchungsdatum (booking date)
- Valuta-Datum (value date)
- End-to-End-ID (end-to-end id)
- Name, Gegenkonto (name, counterparty)
- IBAN, Gegenkonto (iban, counterparty)
- BIC, Gegenkonto (bic, counterparty)
- Verwendungszweck (remittance information)

Soll diese Regelung modifiziert werden (bspw. dahingehend, dass die Prüfung weniger strikt erfolgt), ist hier anzusetzen. Nach Möglichkeit sollte die Änderung nicht direkt in der Klasse AbstractCAMT054Logic vorgenommen werden. Die dafür vorgesehene Sourcecode-Stelle in der konkreten Implementierung CAMT054Logic ist entsprechend mit einem @replace!-Tag markiert.

> ⚠ Es wird empfohlen, die Änderungen in der konkreten Implementierung CAMT054Logic oder in einer eigenen Klasse vorzunehmen, die von der abstrakten Klasse AbstractCAMT054Logic erbt.

org.nuclet.camt.AbstractCAMT054Logic

```
/**
 * Checks, if a representation of the given bank transaction has already been stored to the database.
 *
 * Identifiying criteria could be a combination of:
 *
 * - entry date
 * - value date
 * - amount
 * - debit credit mark
 * - business transaction type code / bank transaction type
 * - information to account owner
 * - remittance information
 *
 * @param transaction a given bank transaction, representated by a <code>CAMT054Transaction</code> object
```

```
     *
     * @return true, if a representation of the given bank transaction has already been stored to the database
     *
     * @throws BusinessException might be thrown in case of errors or other exceptions
     */
    public boolean doesBankTransactionAlreadyExist(CAMT054Transaction transaction) throws BusinessException
    {
        Query<BankTransaction> qryGetExistingBankTransactions = QueryProvider.create(BankTransaction.
class);
        final CreditDebitIndicator boCreditDebitIndicator = creditDebitIndicatorFacade.getCreditDebitIndicator
(transaction.getCreditDebitIndicator().getCode());
        final EntryStatus boEntryStatus = entryStatusFacade.getEntryStatus(transaction.getEntryStatus().
getCode());
        final AbstractCurrencyWrapper currencyWrapper = currencyFacade.getCurrencyByIso4217Code(transaction.
getCurrencyCode());
        BankTransactionType bankTransactionType = null;

        if (transaction.getBankTransactionTypeCode() != null) {
            bankTransactionType = bankTransactionTypeFacade.getBankTransactionType(transaction.
getBankTransactionTypeCode(),
                transaction.getBankTransactionTypeTxtComplement(),
                transaction.getSwiftTransactionCode());
        }

        if (bankTransactionType != null && bankTransactionType.getId() != null) {
            qryGetExistingBankTransactions.where(BankTransaction.MessageId.eq(transaction.getMessageId()))
                .and(BankTransaction.MessageDate.eq(transaction.getMessageDate()))
                .and(BankTransaction.MessageRecipient.eq(transaction.getMessageRecipient()))
                .and(BankTransaction.NotificationId.eq(transaction.getNotificationId()))
                .and(BankTransaction.ElectronicSequenceNumber.eq(transaction.getElectronicSequenceNumber()))
                .and(BankTransaction.TransactionTypeId.eq(bankTransactionType.getId()))
                .and(BankTransaction.EntryStatusId.eq(boEntryStatus.getId()))
                .and(BankTransaction.Initiator.eq(transaction.getInitiator()))
                .and(BankTransaction.FinancialInstitution.eq(transaction.getFinancialInstitution()))
                .and(BankTransaction.Iban.eq(transaction.getIban()))
                .and(BankTransaction.Bic.eq(transaction.getBic()))
                .and(BankTransaction.Amount.eq(transaction.getAmount()))
                .and(BankTransaction.CurrencyId.eq(currencyWrapper.getId()))
                .and(BankTransaction.CreditDebitIndId.eq(boCreditDebitIndicator.getId()))
                .and(BankTransaction.BookingDate.eq(transaction.getBookingDate()))
                .and(BankTransaction.ValueDate.eq(transaction.getValueDate()))
                .and(BankTransaction.EndToEndId.eq(transaction.getEndToEndId()))
                .and(BankTransaction.AccountServiceReference.eq(transaction.getAccountServiceReference()))
                .and(BankTransaction.NameCounterparty.eq(transaction.getNameCounterparty()))
                .and(BankTransaction.IbanCounterparty.eq(transaction.getIbanCounterparty()))
                .and(BankTransaction.BicCounterparty.eq(transaction.getBicCounterparty()))
                .and(BankTransaction.RemittanceInformation.eq(transaction.getRemittanceInformation()));
        } else {
            qryGetExistingBankTransactions.where(BankTransaction.MessageId.eq(transaction.getMessageId()))
                .and(BankTransaction.MessageDate.eq(transaction.getMessageDate()))
                .and(BankTransaction.MessageRecipient.eq(transaction.getMessageRecipient()))
                .and(BankTransaction.NotificationId.eq(transaction.getNotificationId()))
                .and(BankTransaction.ElectronicSequenceNumber.eq(transaction.getElectronicSequenceNumber()))
                .and(BankTransaction.EntryStatusId.eq(boEntryStatus.getId()))
                .and(BankTransaction.Initiator.eq(transaction.getInitiator()))
                .and(BankTransaction.FinancialInstitution.eq(transaction.getFinancialInstitution()))
                .and(BankTransaction.Iban.eq(transaction.getIban()))
                .and(BankTransaction.Bic.eq(transaction.getBic()))
                .and(BankTransaction.Amount.eq(transaction.getAmount()))
                .and(BankTransaction.CurrencyId.eq(currencyWrapper.getId()))
                .and(BankTransaction.CreditDebitIndId.eq(boCreditDebitIndicator.getId()))
                .and(BankTransaction.BookingDate.eq(transaction.getBookingDate()))
                .and(BankTransaction.ValueDate.eq(transaction.getValueDate()))
                .and(BankTransaction.EndToEndId.eq(transaction.getEndToEndId()))
                .and(BankTransaction.AccountServiceReference.eq(transaction.getAccountServiceReference()))
                .and(BankTransaction.NameCounterparty.eq(transaction.getNameCounterparty()))
                .and(BankTransaction.IbanCounterparty.eq(transaction.getIbanCounterparty()))
                .and(BankTransaction.BicCounterparty.eq(transaction.getBicCounterparty()))
                .and(BankTransaction.RemittanceInformation.eq(transaction.getRemittanceInformation()));
        }
```

```
        final List<BankTransaction> lstBankTransactions = QueryProvider.execute
(qryGetExistingBankTransactions);

        return (lstBankTransactions != null && lstBankTransactions.size() > 0);
    }
```

Verwandte Seiten

📖 Nuclet: CAMT(sind im Handelsnuclet verfügbar)

CAMT-Datenblatt

Öffnen

📖 Schnittstellen

Alle Integrationen

Öffnen