4.8.3 CAMT: Uniqueness check

🌐 Language: Deutsch · **English**

🛠️ 4.8.3 CAMT: Uniqueness check

> **Uniqueness check – part of the CAMT Nuclet documentation (import and processing of CAMT.053/CAMT.054 bank statements (Cash Management)).**

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

On this page

⚠️ **Machine-translated – under review**

This page was machine-translated from German as a first draft and is currently under review. The authoritative source remains the **German original** (see the language switch above).

The check of whether an incoming record from the CAMT.054 import already exists in the database is done in the method **doesBankTransactionAlready Exist()** is implemented in the abstract class org.nuclet.camt.logic.AbstractCAMT054Logic. By default, the following attributes are considered in this comparison:

- Message ID (message id)
- Recipient (message recipient)
- Reference number (notification id)
- Bank statement number (electronic sequence number)
- Bank transaction type (business transaction type code/bank transaction type)
- Entry status (entry status)
- Initiator (initiator)
- Bank (financial institution)
- IBAN (iban)
- BIC (bic)
- Amount (amount)
- Currency (currency code)
- Credit/debit indicator (credit debit indicator)
- Booking date (booking date)
- Value date (value date)
- End-to-end ID (end-to-end id)
- Name, counterparty (name, counterparty)
- IBAN, counterparty (iban, counterparty)
- BIC, counterparty (bic, counterparty)
- Remittance information (remittance information)

If this rule is to be modified (e.g. so that the check is less strict), this is the place to start. If possible, the change should not be made directly in the AbstractCAMT054Logic class. The source-code location intended for this in the concrete implementation CAMT054Logic is marked accordingly with an @replace! tag.

⚠️ It is recommended to make the changes in the concrete implementation CAMT054Logic or in your own class that inherits from the abstract class AbstractCAMT054Logic.

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);
    }
```

Related pages

📖 Nuclet: CAMT (available in the trade Nuclet)

CAMT Nuclet data sheet

Open

📖 Interfaces

All integrations

Open